AML Patient Prediction

Georgios Skoulidis

December 23, 2024

4A1. Classification in Practice

4A1.1. Descriptive Analysis

(a) Out of all the labeled instances, over 87% belong to the healthy class, representing most of the dataset. Due to this skewed distribution, we must carefully select the evaluation metrics for this dataset. Accuracy alone can be misleading, because it simply measures the proportion of correctly predicted instances out of the total, without considering the distribution of the classes. Thus, we will also compute precision, recall, and F1-score; the harmonic mean of precision and recall, which will also serve as our primary metric for the system's performance, later when performing cross-validation.

Patients	Instances
Healthy	156
AML	23

Table 1: Instances of Classes

(b)

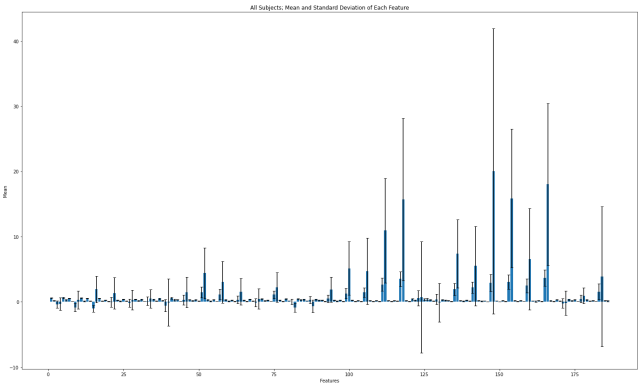

Figure 1: Error bars showing the means and standard deviations for the features of the entire dataset.

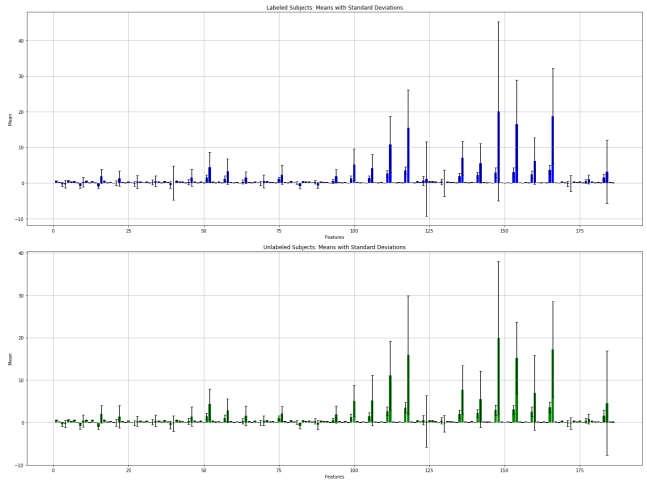

Figure 2: Error bars showing the means and standard deviations for the features of the labeled and unlabeled datasets separately.

Let us examine the mean values of each feature for the entire dataset. Most features have positive mean values, with only a few having negative ones. In addition, certain features exhibit very high mean values with also high standard deviations, indicating significant variability. In general, the features are on different scales, with much variation between them.
This variability can impact subsequent analysis, especially for distance-based methods, since they are sensitive to feature scales and often require normalization or standardization, due to features with high means and standard deviations dominating distance calculations. In contrast, tree-based methods are less affected by feature scales because they rely on threshold-based tree splits.
We should note that a high mean does not imply a feature's importance for classification and a high standard deviation does not necessarily correlate with the feature's ability to distinguish between classes.

4A1.2. Exploratory Analysis

(a) The entire dataset variance can only be fully explained by all principal components. However, the first components capture most of the variance, with each subsequent one contributing progressively less until the last ones have almost zero contribution.
In Figure (3), we observe the percentage of variance explained by each principal component. In Figure (4), we observe the cumulative variance explained by the components, where the curve gradually approaches 100%. This

figure shows that 5 components explain 85% of the variance, 7 explain 90%, and 10 explain 95% of total variance.

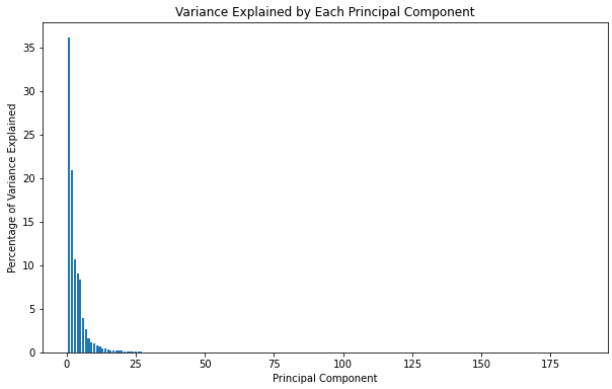

Figure 3: Variance explained by individual principal components.

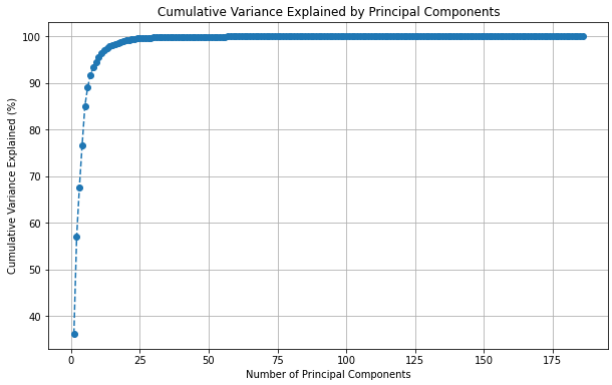

Figure 4: Cumulative variance explained by the principal components.

For the dimensionality reduction, a sufficient proportion of the total variance should be retained so that the distribution remains representative. In our case, using only a few components should be sufficient. For example, with 5 components, we would lose 15% of the total variance, and with 10 components, the loss would drop to only 5%, meaning that the most important information is still there. Distance-based methods are likely to benefit from this dimensionality reduction, as reducing the number of features simplifies distance calculations, potentially improving performance. Tree-based methods could also benefit by concentrating on the more useful components. However, trees are generally more robust to irrelevant features.
Dimensionality reduction offers the advantage of visualization, too. Reducing the dataset to 2 or 3 components could provide insightful visual representations while also retaining significant information.

(b) By performing ANOVA, we assess the statistical significance of the distribution of each feature with respect to the class label in the labeled dataset (179 patients). The most important features are those with the highest F-scores and the lowest p-values.
In distance-based methods, the most relevant features are likely to have greater distances from their class means, contributing more to higher classification performance. Therefore, reducing dimensionality using ANOVA would likely improve performance. It could also prove particularly beneficial for tree-based methods, as it helps identify the most relevant features for making splits, improving the interpretability of the model, and clarifying decision paths.

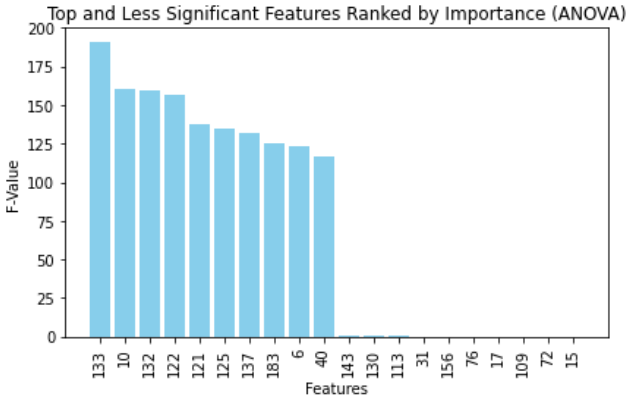

Figure 5: ANOVA F-scores for most and least important features.

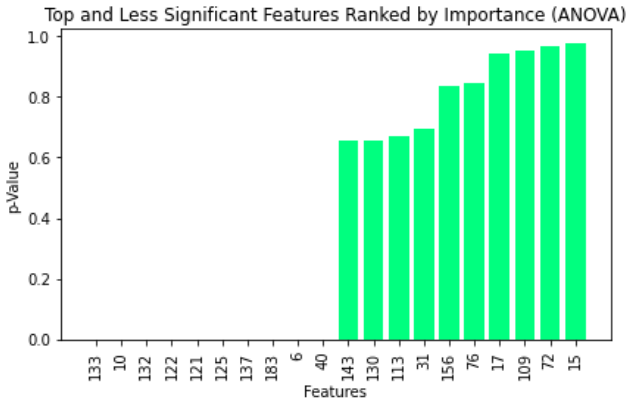

Figure 6: ANOVA p-values for most and least important features.

(c) The first thing we observe when visualizing the data with t-SNE and PCA is the class overlap. Since PCA is designed for linear relationships between features, we notice more overlap in the clusters of the classes, when compared to t-SNE, which provides a better separation. This also means that there is non-linearity in our data. Outliers are also present in the PCA representation and may negatively impact the classifier's performance. Visualizing the unlabeled data helps us understand how challenging it is for a 2D classifier to create a decision boundary between the two classes. We should also note that class imbalance is now visualized and needs to be addressed effectively.

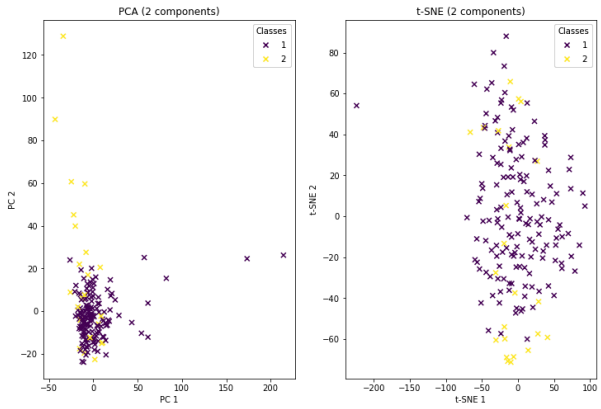

Figure 7: Visualization for 2 PCA and 2 t-SNE components, of the labeled dataset.

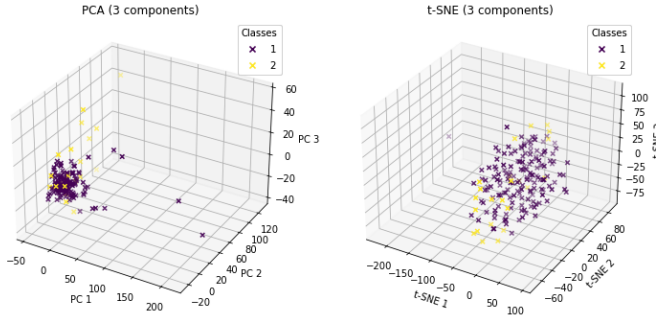

Figure 8: Visualization for 3 PCA and 3 t-SNE components, of the labeled dataset.

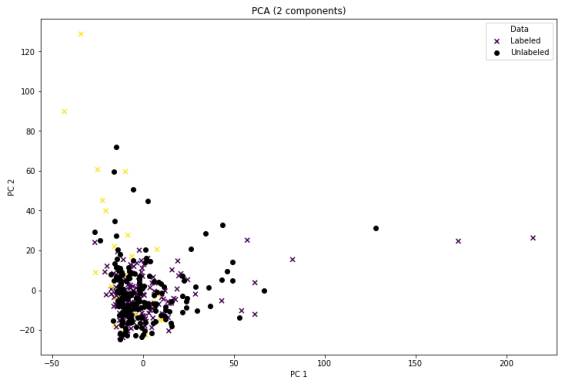

Figure 9: Visualization for 2 PCA components, of both labeled and unlabeled datasets.

4A1.3. Experimentation with training and validation

a Base-line experiments on one fold:

For the Stratified K-fold cross-validation, we used K=5 since the limited amount of labeled subjects prevents us from using higher K values, as the cross-validation set in each fold would be extremely small and non-representative. We also compute F1-score, accuracy, precision, and recall, while using F1 as our main comparison metric.

a.1) kNN:

KNN is a distance-based method, and thus we do not expect it to perform as well as tree-based methods, as the data has not been normalized or standardized. In this initial phase of validation, we select three reasonable values of k: 3, 5, and 6. These values are chosen based on hypothetical cluster areas that could potentially separate classes in the earlier visualizations.

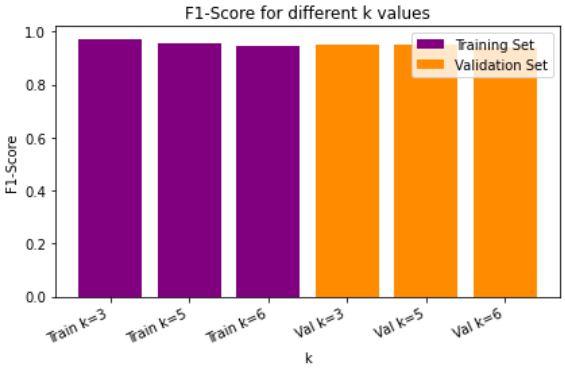

Figure 10: F1 Scores for KNN with varying values of k.

In our case, the training F1-score reaches its highest value when K=3. However, we focus on the validation F1-score, which shows nearly equal performance for k=3, and k=5. If we had to choose a model at this stage, we would aim for the simplest option between the two, which is KNN with k=3.

a.2) DT:

Tree-based methods are less sensitive to feature scales, which makes us expect slightly higher performance compared to distance-based methods. For hyper-parameter tuning, we focus on three main hyper-parameters: `max depth`, `class weights` and `max features`. It is important to note that we utilize `sklearn.tree.DecisionTreeClassifier` [1], and its other default parameters will remain fixed during cross-validation.

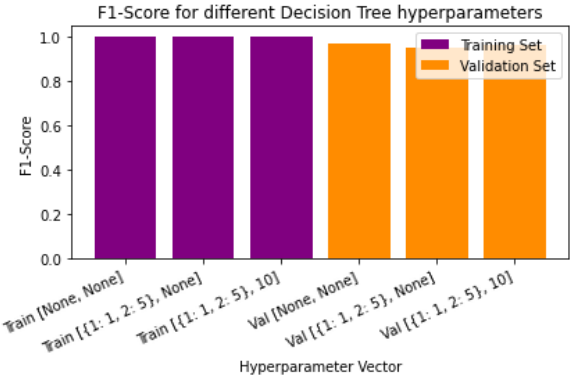

Figure 11: F1 Scores for Decision Tree with different class weights and max features.

What we observe here is that precision, recall, and F1-score all decrease when using `class weights` of 1:5 for the majority:minority class ratio, which gives a boost to the minority class. While we might argue that prioritizing the minority class is important, the classifier appears to underperform, likely due to its failure to accurately identify instances from the majority class, leading to overfitting the minority class. Additionally, it is very possible that the minority class contains outliers, which negatively affected the classification performance after their weights were boosted.

Setting `max_features` to 10, had us also observe another decline in performance. This is likely because 10 features may be insufficient to capture the critical information needed for the classifier to reach an effective decision boundary. We might need to utilize more features to estimate a better real-world function. Regarding `max_depth`, we observe the same exact performance metrics between `max_depth=None` and `max_depth=5`, across all parameter configurations. This suggests a few possibilities. First, it indicates that the natural complexity of the dataset is likely to be fully captured within 5 splits, meaning deeper trees (such as with `max_depth=None`) do not insert overfitting. The tree may not need more depth to separate data. Both configurations are equally sufficient, and further examination could give us more insight regarding different hyperparameter vectors.

It's important to note that without pruning or reducing depth, decision trees can reach a perfect training accuracy of 100%. However, achieving perfect accuracy on validation or evaluation data is highly unlikely. Overfitting is also very likely to occur in such cases, as the model would capture specific patterns in the training set that would not generalize sufficiently to new unseen data.

Table 2: Average F1 Scores for Different Models

Model	Avg F1/Training	Avg F1/Validation
KNN (k=3)	0.9720	0.9513
KNN (k=5)	0.9556	0.9514
KNN (k=6)	0.9469	0.9400
DT (1:1, max_feat=None, max_depth=None)	1.0000	0.9711
DT (1:5, max_feat=None, max_depth=None)	1.0000	0.9629
DT (1:5, max_feat=10, max_depth=None)	1.0000	0.9522
DT (1:1, max_feat=None, max_depth=5)	1.0000	0.9711
DT (1:5, max_feat=None, max_depth=5)	1.0000	0.9629
DT (1:5, max_feat=10, max_depth=5)	1.0000	0.9522

b Including preprocessing:

b.1) kNN:

The benefits of each given pre-processing method are briefly outlined below:

- **Z-Score Transformation:** ensures all features contribute equally to the distance metric.
- **Principal Component Analysis:** opposes the curse of dimensionality by retaining the principal components explaining most of the variance.
- **Feature Selection with ANOVA:** opposes the curse of dimensionality by simply retaining the most important features

We consider both Z-score transformation and feature selection to be the appropriate pre-processing methods, as the first standardizes the data across different features and the second opposes the curse of dimensionality issue. We believe that especially ANOVA feature selection would sufficiently improve the model's performance. We also think that the previously tested values of K will perform well. To take one step further towards our claim, we have decided to explore various configurations involving pre-processing parameters. This approach will help us back up our hypothesis and potentially identify a highly effective model.

We perform 5-Fold Stratified Cross-Validation and at each split, we compute the mean value of the training set and subtract it from both the training set and the validation set, in order to apply PCA to the training set and transform the validation set using the principal components derived from the training data. We use several numbers of principal components for tuning, before we apply KNN, also testing several values of K. We plot F1-scores in Figure (12).

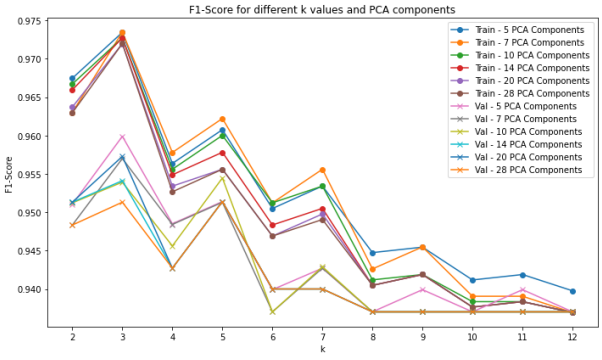

Figure 12: F1-Score of KNN with PCA features

We observe that for K=3, and with 5 principal components, we achieve a validation F1-score of 0.9600, which is the highest obtained so far with the KNN algorithm.
Then we apply ANOVA to each training set to get the n most important features. Again, we explore a grid of various values to optimize the number of features retained. We tune for different values of K, and plot the F1-score results in Figure (13).

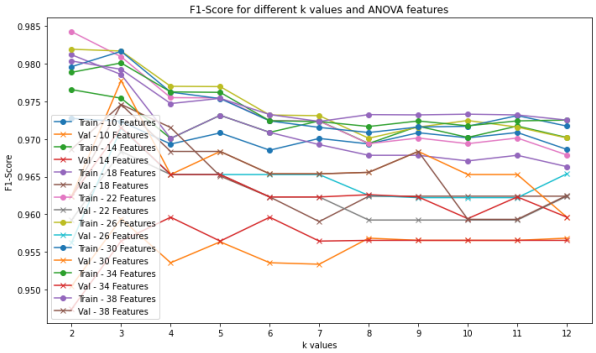

Figure 13: F1-Score of KNN with ANOVA selected features

We now observe that for 30 selected features with K=3, we get an F1-score of 0.9778. This is by far the best validation score we have obtained so far with the KNN algorithm.

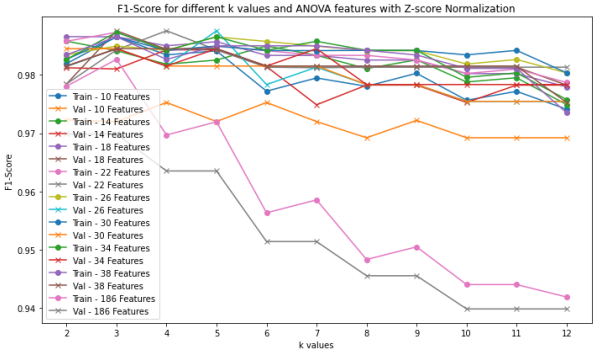

Figure 14: F1-Score of KNN with Z-Score and ANOVA selected features

Lastly, we apply a Z-score transformation to the training set following the same pre-processing logic as before. However, we could also consider applying the transformation after selecting different numbers of features with ANOVA. Therefore, we create a grid that includes various values for the number of features, same as before, but now also includes the total number of features (186) to simulate the scenario where ANOVA is not applied. We then plot the F1-scores for the different configurations in Figure (14). Z-score transformation without feature selection yields a maximum validation F1-score of 0.9699 for both K=2 and K=3. While it still yields better results than what PCA did, selecting certain features with ANOVA clearly outperforms the simple transformation. Nevertheless, when these two techniques are combined, they produce exceptionally high-performing KNN models.

We identified three different configurations with an F1-score of 0.9876, marking the highest performance among every KNN model. These configurations correspond to: (K=3, 18 selected features), (K=4, 22 selected features), and (K=5, 26 selected features). These models demonstrate equal capability, and thus if we had to select one, we would opt for the simplest configuration: K=3 with the selection of the 18 most important ANOVA features, scaled using Z-score transformation.
For our findings, we should not be surprised that a distance-based method like KNN performs better when only important features are selected and scaling is applied to them.

b.2) DT:

Decision trees may not benefit from preprocessing the same way that KNN does, but it is still quite possible. We are highly convinced that standardization will not enhance classification as it did with KNN, mainly due to trees making decisions based on thresholds, thus having no sensitivity over feature scales. Moreover, PCA is challenging to assess in terms of its potential benefit. While dimensionality reduction could help simplify the model, transforming data into principal components might cause features to lose their original meaning. This could negatively impact the decision tree's ability to classify.
Finally, ANOVA looks more promising as it retains the most relevant features in their natural form.
Since decision trees contain multiple hyperparameters, it is a bit more difficult to visualize F1-scores from multiple configurations as we did before. Thus, we choose to stick with the configurations we used in the previous question and concentrate on the impact of ANOVA feature selection on them specifically. We will also tune for different numbers of selected features and provide the results in tables.
We perform a parameter tuning process similar to before, utilizing the three configurations previously used for decision trees, along with a feature selection grid of [10, 14, 18, 22, 26, 30, 34, 38, 50, 100]. The process is optimized for different numbers of selected features. All results are saved in the file `dt_with_anova_results.csv`, which we provide for further analysis.
Upon observing the obtained validation F1-scores, we observe that for all configurations utilizing ANOVA feature selection, the average F1 value is 0.9591. The average validation F1 score without feature selection is 0.9621, suggesting that feature selection does not benefit the decision tree overall. However, we managed to also achieve a maximum F1 score of 0.9748 with 26 selected features and the hyperparameters `'class_weight': {1: 1, 2: 5}, 'max_features': 10, 'max_depth': None` (same for `max_depth=5`). This leads us to conclude that while feature selection may not be advantageous overall, it can enhance results for certain configurations, emphasizing the importance of tuning for every hyperparameter.

Table 3: Top Configurations for Decision Trees with ANOVA Feature Selection (Validation F1 Score Only)

Hyperparameters	Selected Features	Validation F1
'class_w': -, 'max_feat': -, 'depth': -	30	0.9648
'class_w': -, 'max_feat': -, 'depth': -	38	0.9648
'class_w': -, 'max_feat': -, 'depth': 5	30	0.9648
'class_w': -, 'max_feat': -, 'depth': 5	38	0.9648
'class_w': -, 'max_feat': -, 'depth': 5	18	0.9992
'class_w': -, 'max_feat': -, 'depth': -	22	0.9650
'class_w': -, 'max_feat': -, 'depth': -	26	0.9653
'class_w': -, 'max_feat': -, 'depth': 5	26	0.9653
'class_w': 1: 1, 2: 5, 'max_feat': 10, 'depth': -	38	0.9676
'class_w': 1: 1, 2: 5, 'max_feat': 10, 'depth': 5	38	0.9676
'class_w': -, 'max_feat': -, 'depth': -	50	0.9681
'class_w': -, 'max_feat': -, 'depth': 5	50	0.9681
'class_w': -, 'max_feat': -, 'depth': 5	22	0.9682
'class_w': -, 'max_feat': -, 'depth': -	100	0.9747
'class_w': -, 'max_feat': -, 'depth': 5	100	0.9747
'class_w': 1: 1, 2: 5, 'max_feat': 10, 'depth': -	26	0.9748
'class_w': 1: 1, 2: 5, 'max_feat': 10, 'depth': 5	26	0.9748

c Ensemble:
Ensemble methods often boost classification performance by combining multiple well-performing models. For KNN, we will employ three different classifiers to form an ensemble, majority voting classifier. Meanwhile, for decision trees, we will utilize a Random Forest, which is an often-used ensemble method.

c.1) **kNN:**
We utilize the best-performing classifiers for each preprocessing technique we used before. Those are:

- KNN with K=3, PCA with 5 PCs (yielded F1=0.9600)
- KNN with K=3, ANOVA with 30 features (yielded F1=0.9778)
- KNN with K=3, ANOVA with 30 scaled features with Z-score transformation (yielded F1=0.9876)

We get an ensemble classifier, by aggregating the majority votes from each fold, capable of predicting validation data with an F1 score of **0.9782**. This indicates a slight performance improvement, proving that ensembling effectively improves predictions for our best KNN models.

c.2) **DT:**
Random Forest is a tree-based ensemble method that makes predictions by averaging the results from multiple decision trees (`n_estimators`). We use the same decision tree criteria as before. Since both `max_depth = None` and `max_depth = 5` give us identical validation results, we focus only on `max_depth = None`. We also experiment with different numbers of estimators to determine the optimal configuration. The parameter grid we use is:

- **estimator_grid** = [5, 10, 25, 40, 60, 100]
- **hyperparameters**:
  1. {'class_weight': None, 'max_features': None, 'max_depth': None}
  2. {'class_weight': {1: 1, 2: 5}, 'max_features': None, 'max_depth': None}
  3. {'class_weight': {1: 1, 2: 5}, 'max_features': 10, 'max_depth': None}

After evaluating different configurations, the best ensemble model occurs with:

- **n_estimators**: 25
- **class_weight**: {1: 1, 2: 5}
- **max_features**: 10

Below we also provide all the validation metrics of the optimal decision tree model.

- **Accuracy** = 0.9833
- **Precision** = 0.9848
- **Recall** = 0.9833
- **F1 Score = 0.9801**

This model stands out as the top performer so far and strongly highlights the effectiveness of ensembling techniques.

c.1) **d Average and spread of performance over all cross-validation folds:**
We have already computed the means of the most promising models across all cross-validation folds. Since we have identified that the optimal model is the Random Forest, we can also calculate and report its standard deviation.

$$\text{Accuracy} : 0.9833 \pm 0.0333$$
$$\text{Precision} : 0.9848 \pm 0.0304$$
$$\text{Recall} : 0.9833 \pm 0.0333$$
$$\text{F1} : 0.9801 \pm 0.0397$$

4A1.4. Results

The model was ultimately trained on the entire training set and then used to predict whether an individual was healthy or had the disease in the provided unlabeled data. The professors, who had access to the labels, validated our results. The final prediction model achieved a sensitivity of 0.95 and specificity of 1.0 and thus the received the second best place in the class competition.

References

[1] Scikit-learn. Decisiontreeclassifier, 2024. Accessed: 2024-10-05.