Assignment 3A

David Boerema (s3683869)
Marios Souroulla (s4765125)
Marius Captari (s4865928)
Max Valk (s3246922)

Group 3

October 12, 2021

3A1.1

3A1.1a

The entropy of the collection of training examples is:

$$E = -p(+)\log_2(p(+)) - p(-)\log_2(p(-)) = -\frac{4}{9}\log_2(\frac{4}{9}) - \frac{5}{9}\log_2(\frac{5}{9}) \approx 0.9911 \tag{1}$$

3A1.1b

Split on a_1

Entropy of the resulting node for $a_1 = F$:

$$E(a_1 = F) = -\frac{1}{5}\log_2(\frac{1}{5}) - \frac{4}{5}\log_2(\frac{4}{5}) \approx 0.7219 \tag{2}$$

Entropy of the resulting node for $a_1 = T$:

$$E(a_1 = T) = -\frac{3}{4}\log_2(\frac{3}{4}) - \frac{1}{4}\log_2(\frac{1}{4}) \approx 0.8113 \tag{3}$$

Then the information gain IG is given as:

$$IG = E - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) = 0.9911 - \left(\frac{5}{9} * 0.7219 + \frac{4}{9} * 0.8113\right) \approx 0.2294 \tag{4}$$

Split on a_2

Entropy of the resulting node for $a_2 = F$:

$$E(a_2 = F) = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1 \tag{5}$$

Entropy of the resulting node for $a_2 = T$:

$$E(a_2 = T) = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) \approx 0.9710 \tag{6}$$

Then the information gain IG is given as:

$$IG = E - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) = 0.9911 - \left(\frac{4}{9} + \frac{5}{9} * 0.9710\right) \approx 0.007 \tag{7}$$

3A1.1c

We can evaluate all possible splits by splitting on a $>$ threshold on 1 up to 8. For the splits using >1 and >8, all data points will be put in the same group, so therefore the entropy remains unchanged (an information gain of 0). For the remaining splits, we will need to compute the resulting information gain. Table 1 gives the resulting distribution of the classes over the positive and negative node for a threshold condition. There are in total 12 node entropies that have to be calculated, but these fall within 6 categories (For instance, both the False node for threshold >2 and the True node for condition >7 have a distribution of 1 of one class and 0 of the other, and thus equal entropy). We will calculate those and then the resulting information gain for each split.

Threshold:	T:+	T:-	F:+	F:-
>2	3	5	1	0
>3	3	4	1	1
>4	2	4	2	1
>5	2	2	2	3
>6	1	2	3	3
>7	0	1	4	4

Table 1: Class distribution for True and False nodes if we split on a certain thresholds on a_3

Node types

Note: E-node$_{i-j}$ denotes the entropy of a node such that the ratio of one class to the other is $i:j$.

$$\text{E-node}_{1-0} = -\frac{1}{1}\log_2(\frac{1}{1}) - \frac{0}{1}\log_2(\frac{0}{1}) = 0 \tag{8}$$

$$\text{E-node}_{1-1} = -\frac{1}{2}\log_2(\frac{1}{2}) - \frac{1}{2}\log_2(\frac{1}{2}) = 1 \tag{9}$$

$$\text{E-node}_{1-2} = -\frac{1}{3}\log_2(\frac{1}{3}) - \frac{2}{3}\log_2(\frac{2}{3}) \approx 0.9183 \tag{10}$$

$$\text{E-node}_{2-3} = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) \approx 0.9710 \tag{11}$$

$$\text{E-node}_{3-4} = -\frac{3}{7}\log_2(\frac{3}{7}) - \frac{4}{7}\log_2(\frac{4}{7}) \approx 0.9852 \tag{12}$$

$$\text{E-node}_{3-5} = -\frac{3}{8}\log_2(\frac{3}{8}) - \frac{5}{8}\log_2(\frac{5}{8}) \approx 0.9544 \tag{13}$$

Information gains of each split

Let IG$_{>i}$ denote the information gain after using $>i$ as a threshold in order to split the dataset in two.

$$\text{IG}_{>2} = 0.9911 - \left(\frac{1}{9}\left(\text{E-node}_{1-0}\right) + \frac{8}{9}\left(\text{E-node}_{3-5}\right)\right) \approx 0.1427 \tag{14}$$

$$\text{IG}_{>3} = 0.9911 - \left(\frac{2}{9}\left(\text{E-node}_{1-1}\right) + \frac{7}{9}\left(\text{E-node}_{3-4}\right)\right) \approx 0.0026 \tag{15}$$

$$\text{IG}_{>4} = 0.9911 - \left(\frac{3}{9}\left(\text{E-node}_{1-2}\right) + \frac{6}{9}\left(\text{E-node}_{1-2}\right)\right) \approx 0.0728 \tag{16}$$

$$\text{IG}_{>5} = 0.9911 - \left(\frac{5}{9}\left(\text{E-node}_{2-3}\right) + \frac{4}{9}\left(\text{E-node}_{1-1}\right)\right) \approx 0.0072 \tag{17}$$

$$\text{IG}_{>6} = 0.9911 - \left(\frac{6}{9}\left(\text{E-node}_{1-1}\right) + \frac{3}{9}\left(\text{E-node}_{1-2}\right)\right) \approx 0.0183 \tag{18}$$

$$\text{IG}_{>7} = 0.9911 - \left(\frac{8}{9}\left(\text{E-node}_{1-1}\right) + \frac{1}{9}\left(\text{E-node}_{1-0}\right)\right) \approx 0.1022 \tag{19}$$

3A1.1d

The error classification error is given as:

$$\text{Error}(t) = 1 - \max_i p(i|t) \tag{20}$$

Split on a_1

The classification error for splitting on a_1 is:

$$\frac{5}{9}\left(1 - \frac{4}{5}\right) + \frac{4}{9}\left(1 - \frac{3}{4}\right) = \frac{2}{9} \tag{21}$$

Split on a_2

The classification error for splitting on a_1 is:

$$\frac{4}{9}\left(1 - \frac{1}{2}\right) + \frac{5}{9}\left(1 - \frac{3}{5}\right) = \frac{4}{9} \tag{22}$$

A lower classification error is better, so therefore a split on a_1 is the best split.

3A1.1e

The Gini index is defined as:

$$\text{Gini}(t) = 1 - \sum_{i=0}^{c-1} p(i|t)^2 \tag{23}$$

Split on a_1

The Gini index for splitting on a_1 is given as:

$$\frac{5}{9}\left(1 - \left(\left(\frac{1}{5}\right)^2 + \left(\frac{4}{5}\right)^2\right)\right) + \frac{4}{9}\left(1 - \left(\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right)\right) \approx 0.2611 \tag{24}$$

Split on a_2

The Gini index for splitting on a_2 is given as:

$$\frac{4}{9}\left(1 - \left(\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2\right)\right) + \frac{5}{9}\left(1 - \left(\left(\frac{2}{5}\right)^2 + \left(\frac{3}{5}\right)^2\right)\right) \approx 0.4888 \tag{25}$$

A lower Gini index is better, so therefore a split on a_1 is the best split.

3A2

3A2.1

The interested reader should note that a notebook performing the exploratory analysis is available on gitlab.

3A2.1a - Feature investigation

First, we observed the range of each feature, ranked from largest to smallest. As we can see in Figure 1, there is quite a large difference in range between the various features. This indicates that prior to using distance-based methods, a form of normalisation is desired.

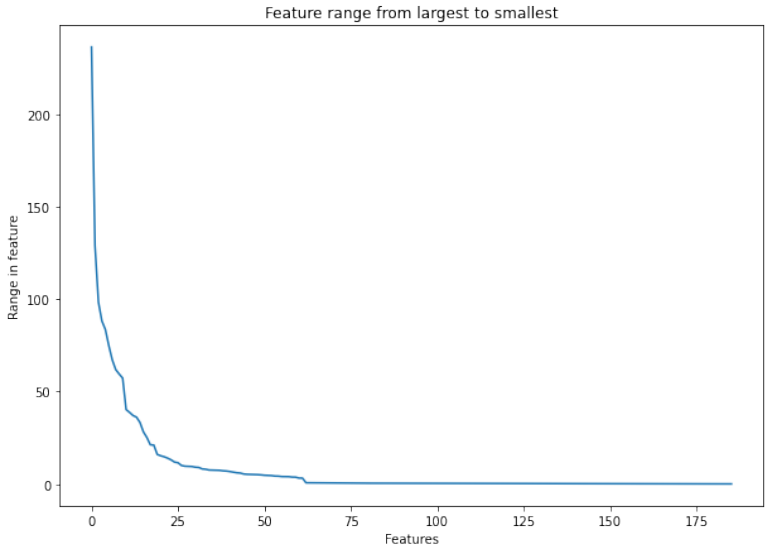


Figure 1: The ranges found on each feature, sorted from largest to smallest. The x-axis represents the rank with respect to how large the range is, not the feature numbers themselves. The minimum range found is approximately 0.059, the maximum range is approximately 236.18 and the median range found was approximately 0.392.

Next, we perform a PCA and take a look at how well the different principal components can represent the data. This can be seen in Figure 2. We found that at least 90% of the variance is explained by the top 6 components, at least 95% is explained by the top 9 components and 99% of the variance is explained by the top 19 components.

Finally, we perform an ANOVA to compare the means of each feature between the classes. We found that 114 of the features differed significantly at a significance level of $p < 0.05$, and 73 at a level of $p < 0.001$. The 10 most significant features found can be seen in Figure 3. However, it should be noted that of all features, only 9 meet the ANOVA criterion of coming from a normally distributed population (tested with Shapiro-Wilk, thresholded against $p < 0.05$). Of those 9, 8 meet the homogeneity of variance assumption (Bartlett, thresholded against $p < 0.05$). Of those 8, 2 features differ significantly with $p < 0.001$ (features 22 and 89), and 4 others with $p < 0.05$ (features 18, 60, 64, 92). These are visualized in Figure 4. While we do note that it is improper with regards to the assumptions of ANOVA, we continue with the 73 features that reached the significance threshold of $p < 0.001$ as experimentally, it was found that those features combined lead to clearer separation while using t-SNE than when using the 8 features for which the result of the ANOVA can be said to be sound. Finally, for contrast, the worst features with the highest p-values can be seen in Figure 5.

3A2.1a - Holistic view of the data

We embed the data using the first 3 principal components, as can be seen in Figure 6. We do see that there is a subset of the data for class 2 that seems to be separated clearly from class 1 in these 3 dimensions, but there also exists a group of points of class 2 that overlap in space with class 1. The unlabelled data largely is distributed in the same way as the labelled data. There is one point that is in a region of space that is not close to any labelled point, which may be more difficult to classify.

Next, we will embed the labelled data using t-SNE, as can be seen in Figure 7. The classes do not seem to be very separated in this representation.

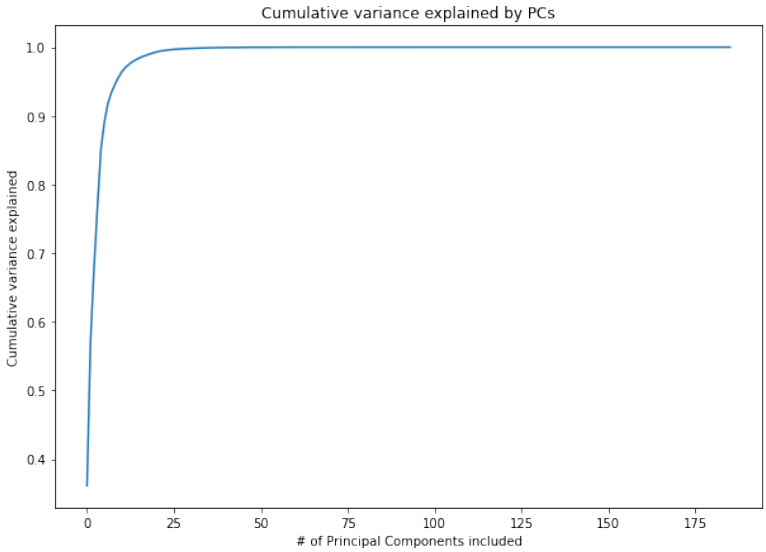


Figure 2: The variance explained by including the top x leading components.

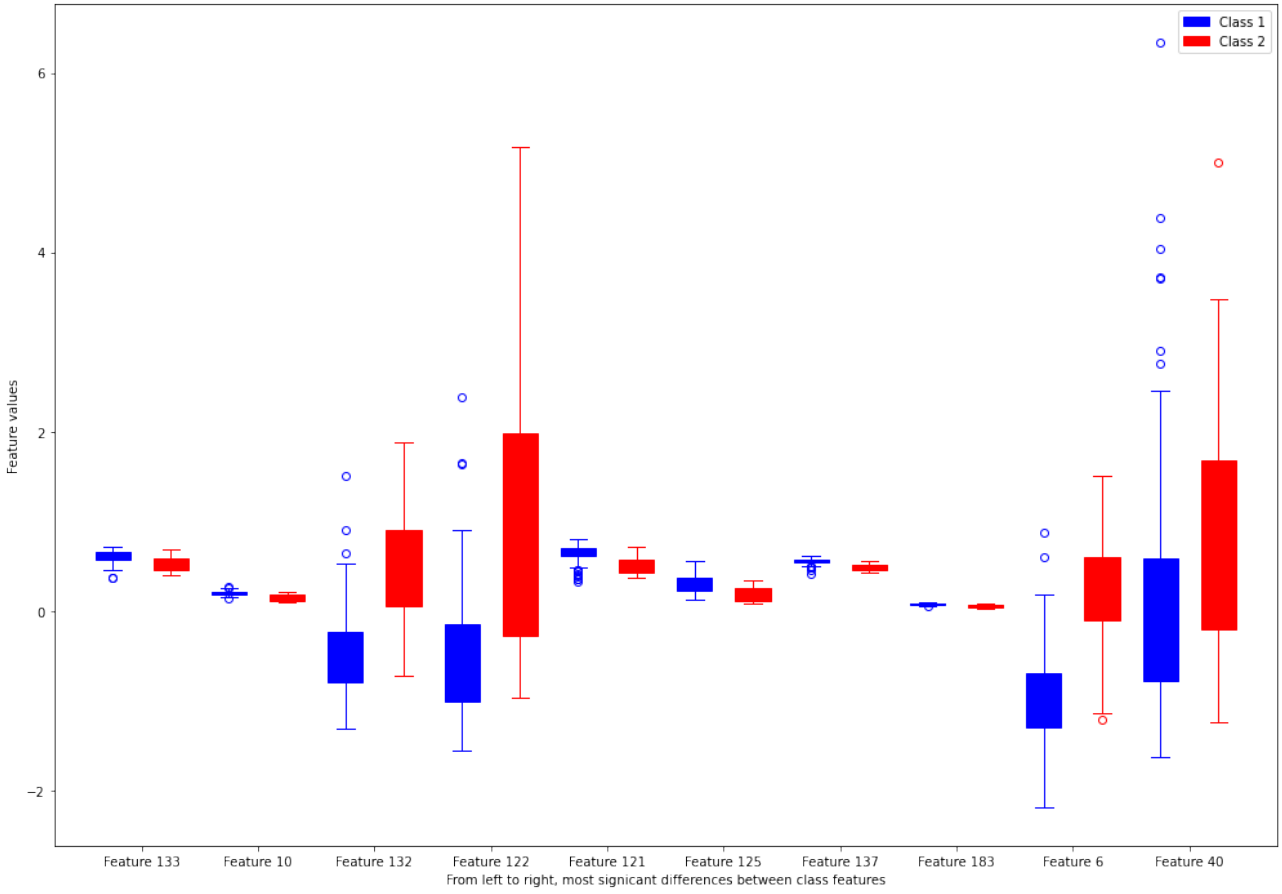


Figure 3: The feature values of the 10 most significant features according to an ANOVA.

3A2.1a - Impact of pre-processing

We will observe how different pre-processing steps affect the ability of t-SNE to create a clearer separation in a 3 dimensional embedding. First, we will perform these steps on the original data, and then we will perform the same steps on a new dataset which has been created using SMOTE.

Pre-upsampling: In Figure 8 we see the effect of various pre-processing methods on the resulting t-SNE embedding. The Relief feature selection seems to be the most effective in creating two somewhat separate clusters, but they still overlap. ANOVA creates three overlapping clusters. Performing a z-transform on all features or

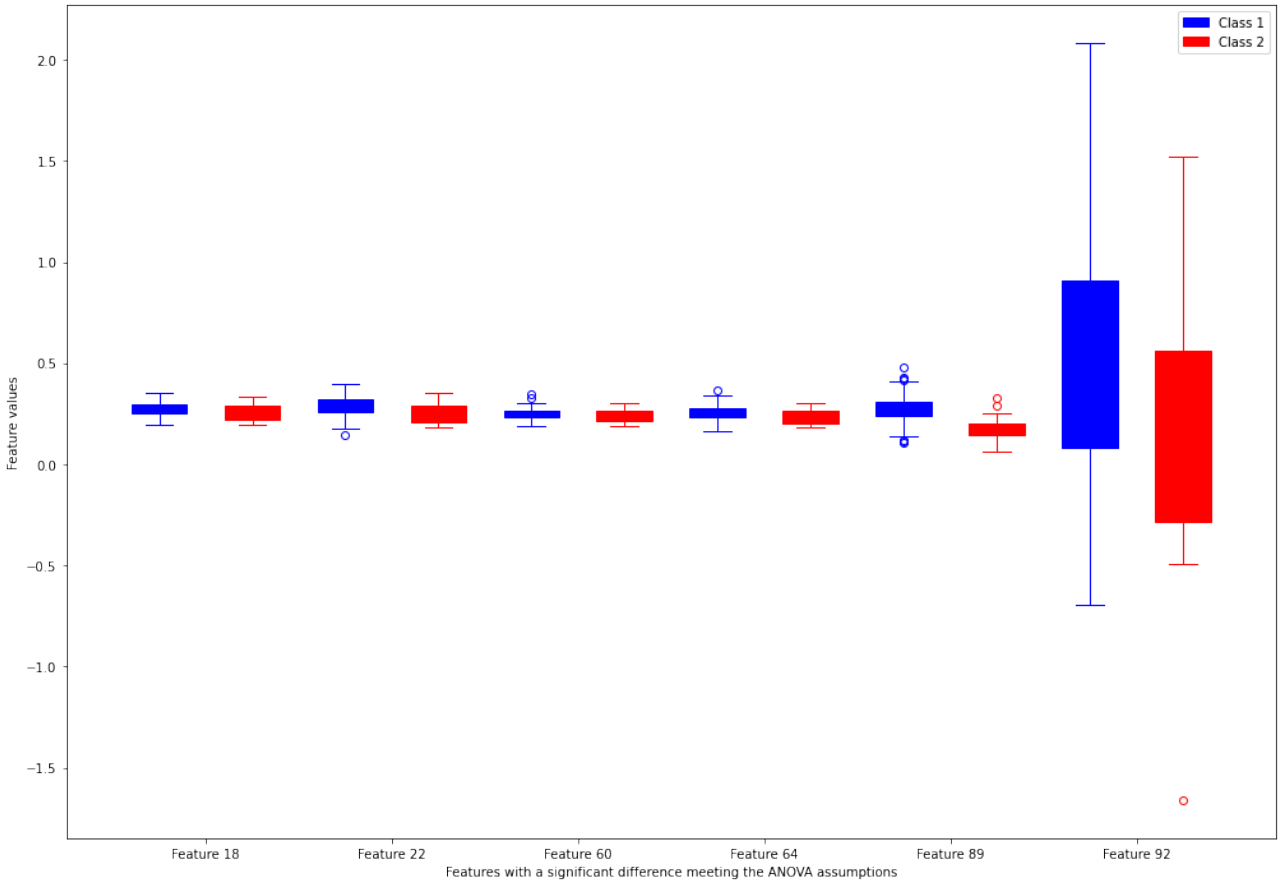


Figure 4: The feature values of the 6 features that were significant and met the assumptions of the ANOVA.

projecting to the top 9 PCs does not seem to aid in a clear separation of the class datapoints. Combining the features found with Relief and ANOVA did not help with the separation either.

Upsampling using SMOTE: We apply t-SNE after upsampling in order to obtain a baseline performance prior to feature selection. The result can be seen in Figure 9. This already seems like a much better embedding, with 3 clear clusters of the classes. We then applied the same transforms as before to the new dataset, the results of which can be seen in Figure 10. The PC-embedding and z-transform do not seem to increase the separability in the t-SNE embedding, but the use of ANOVA and relief (as well as their combination) results in a possible embedding where the datapoints are nearly perfectly separable with a plane.

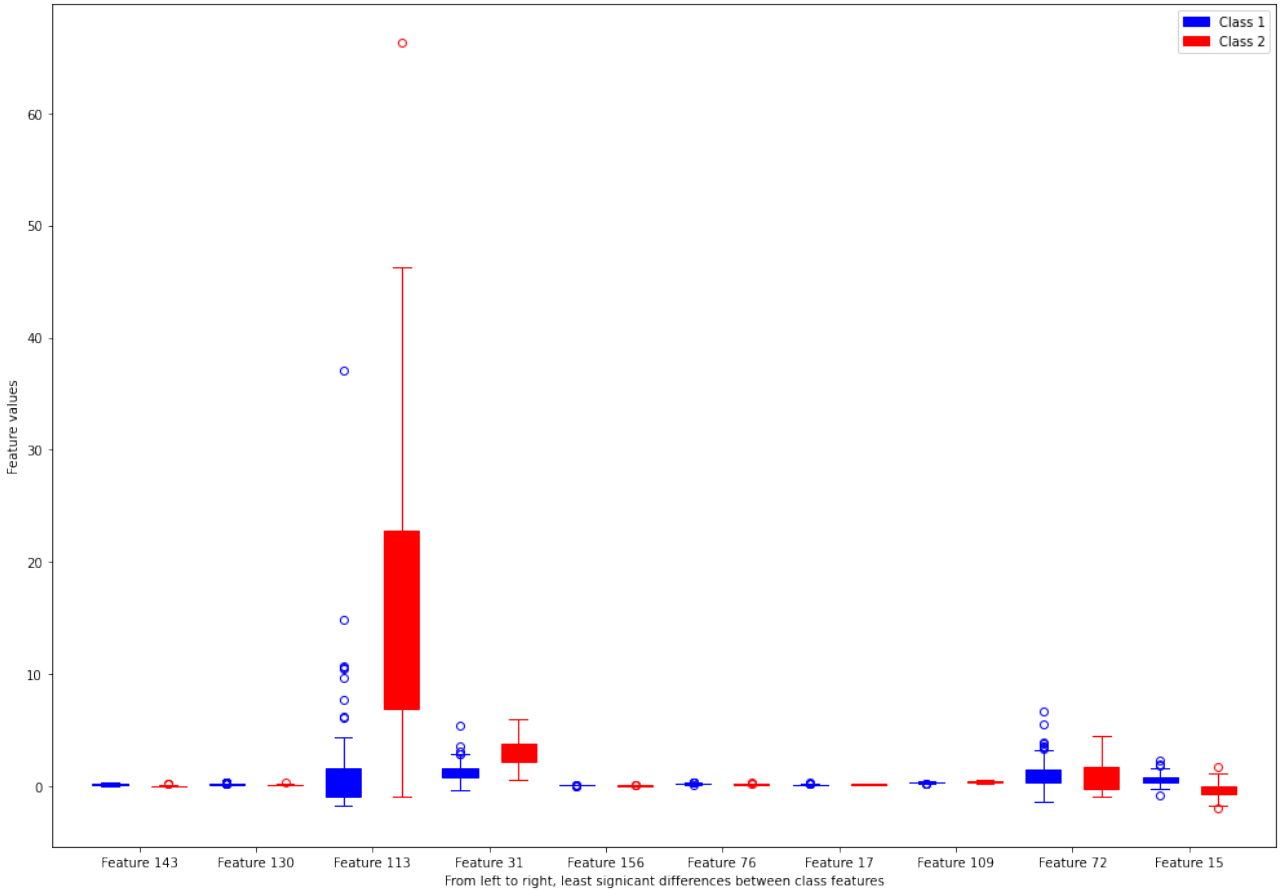


Figure 5: The feature values of the 10 features that were found the least significant.

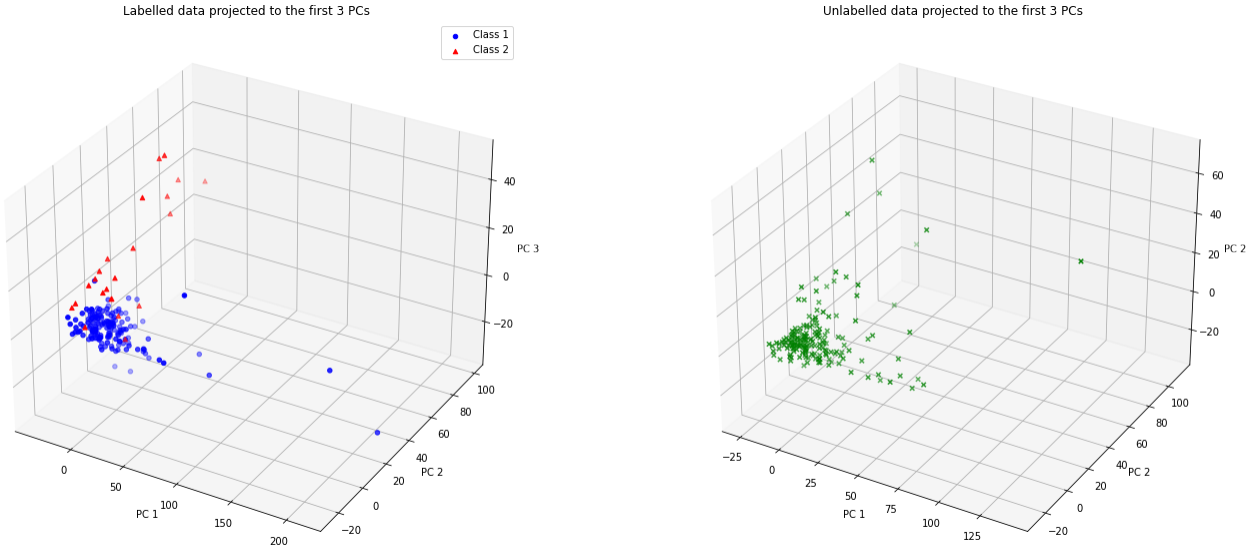


Figure 6: The labelled data (left) and unlabelled data (right) embedded in the first three principal components.

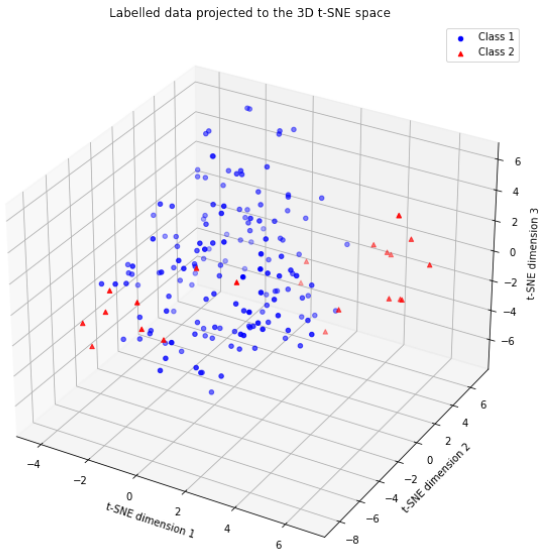
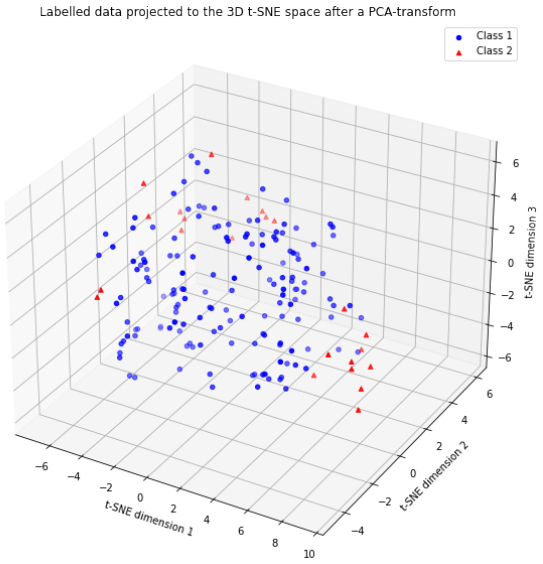
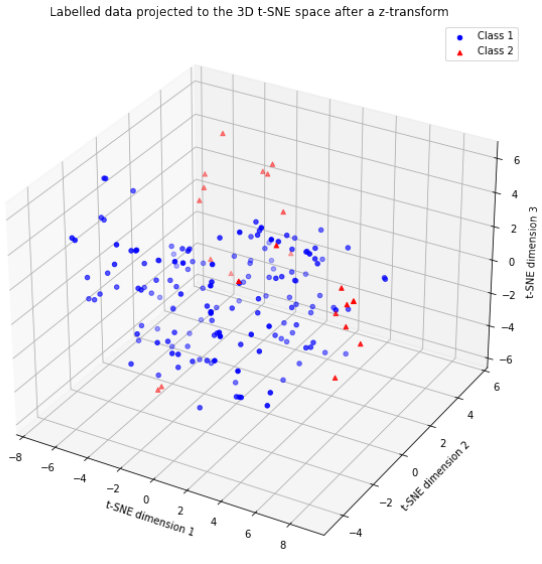


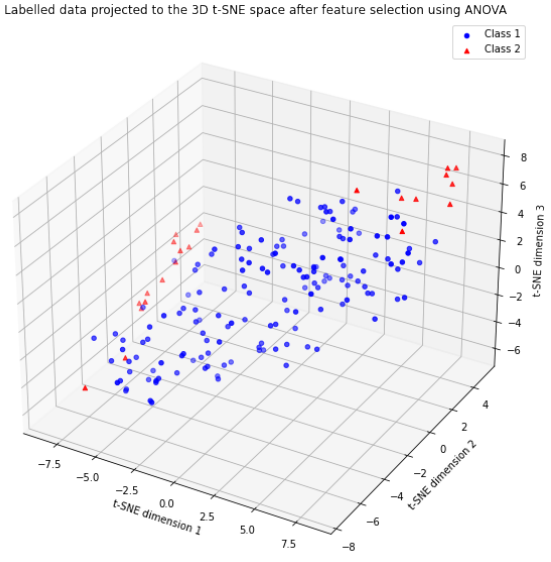
Figure 7: The data embedded in 3 dimensions using t-SNE using all features.



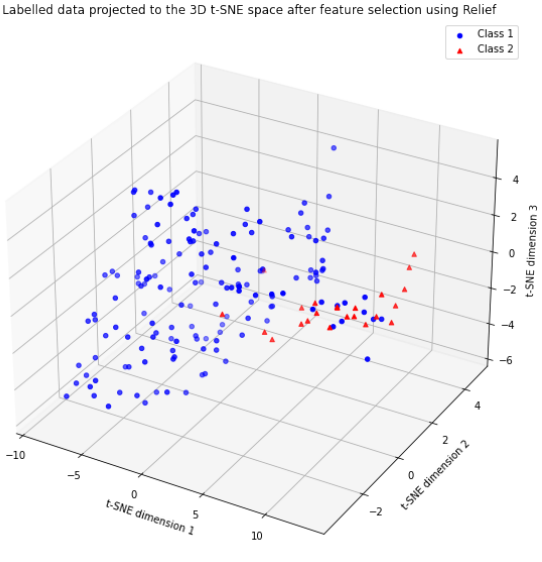
(a) Using top 9 PCs



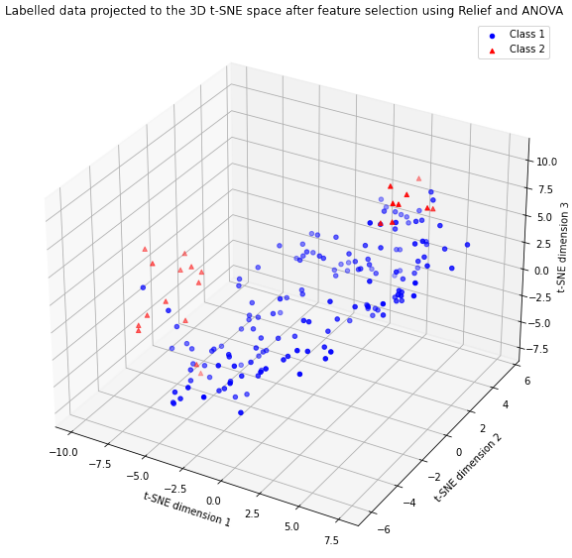
(b) After z-transform



(c) Using 73 significant features from ANOVA



(d) Top 10 features based on Relief



(e) Using the relief features and ANOVA combined.

Figure 8: The labelled data embedded in 3 dimensions using t-SNE after having applied a variety of transformations/selections.

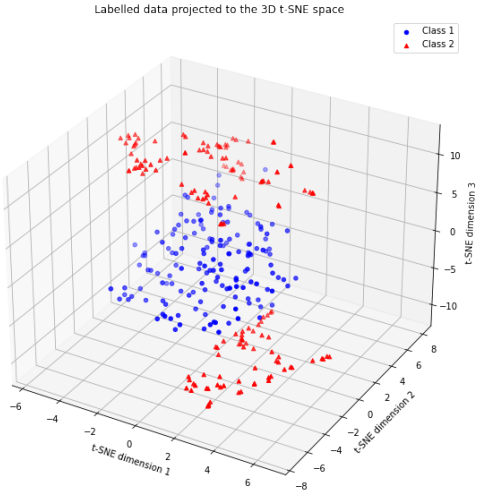
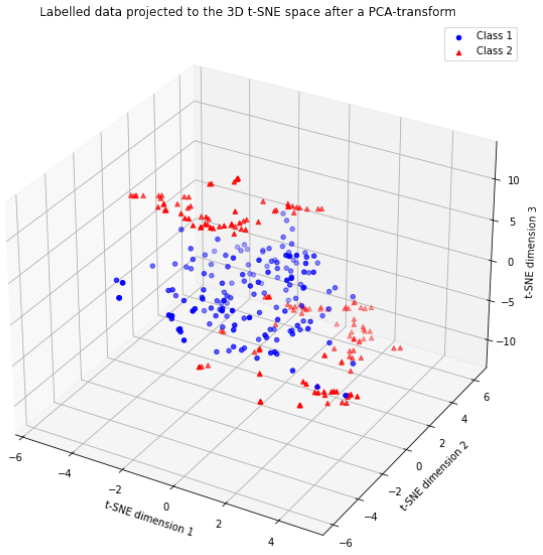
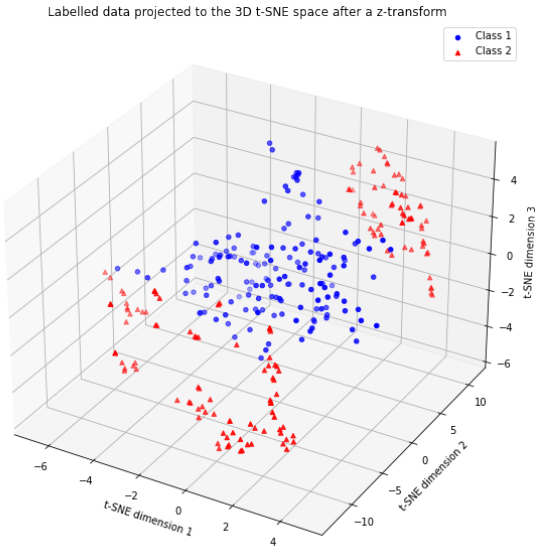


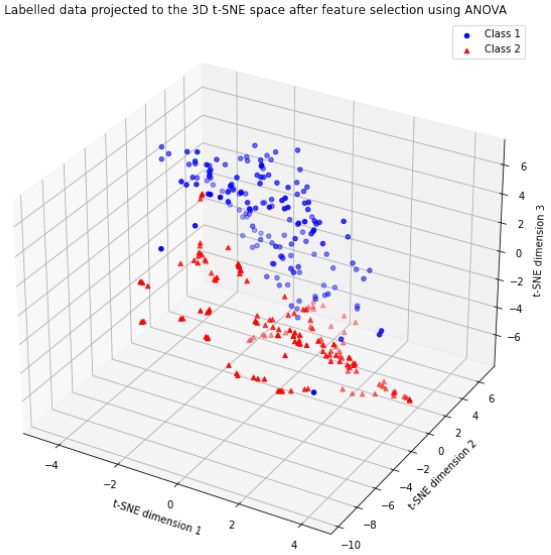
Figure 9: The t-SNE projection to 3 dimensions after upsampling.



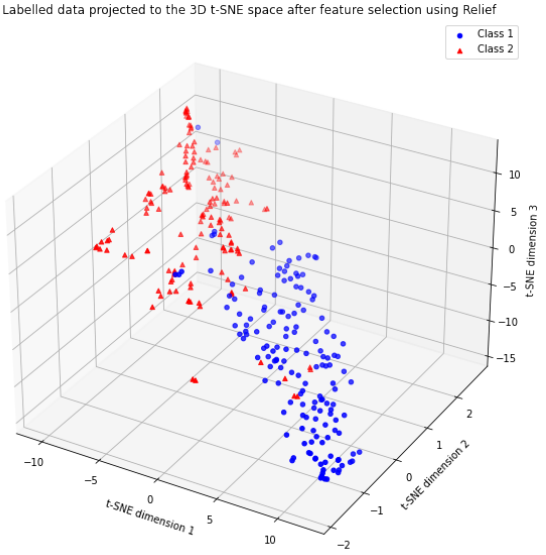
(a) Using top 9 PCs



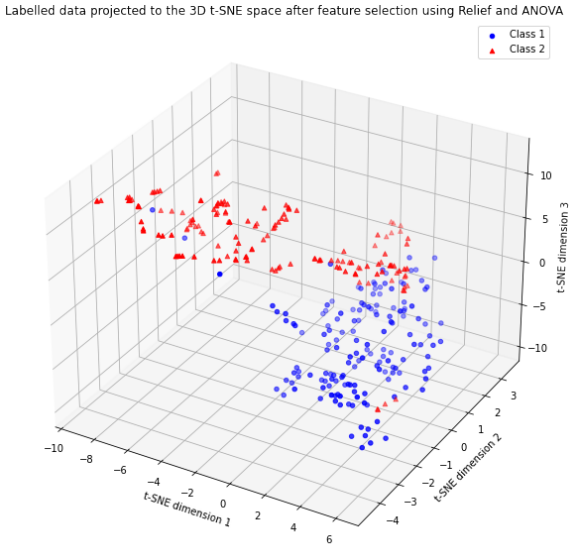
(b) After z-transform



(c) Using 73 significant features from ANOVA



(d) Top 10 features based on Relief



(e) Using the relief features and ANOVA combined.

Figure 10: The labelled data embedded in 3 dimensions using t-SNE after having applied a variety of transformations/selections to the upsampled dataset.

3A2.2

As seen from the analysis done in section 3A2.1, the provided dataset contains a heavy imbalance in the target category. The labeled dataset contains 179 samples of patients, of which only 15% have AML. Fortunately, there are no null values to be found in any of the 186 features for all the 359 subjects provided.

Splitting the data Train data will be used to train the models using a 5 fold cross-validation while test data will be used for validating how well the trained model behaves when provided with previously unseen data and to prevent over-fitting. The train/test split ratio used was 80/20. To ensure that we maintain a similar ratio between class labels in both the train and test datasets a stratify option was used during the split.

Performance metrics In order to compares our different models we need to define which performance metric(s) shall be used. Given that we have an imbalanced dataset and we want to optimize for overall model performance we opted to use F_1-score as the main metric. To help with visualizing performance we also calculate the confusion matrix of the predictions on the test dataset. These matrices can be found in the notebooks related to 3A2-2.

Test and train framework To conduct all of these experiments we will use the GridSearchCV object from the scikit-learn python library. This object allows us, as the name suggests, to perform grid search while doing cross validation. In addition, making use of the imbalanced-learn library, the pipeline object will be used as an input to each GridSearchCV object. This pipeline objects contains not only the desired model to use but also all the prepossessing to be done beforehand. For an overview of all performed experiments please refer to Figure 11.

3A2.2a - Base-line experiments

Decision Trees The first model we've tested in our experiments was decision trees. To start off, a simple model was created without any prepossessing or any parameter tuning to have a baseline performance metric. From section 3A2.1a - Impact of pre-processing it was clear that using an upsampling technique like SMOTE, followed by feature selection based on ANOVA could increase model performance. Both ANOVA and SMOTE seemed to improve performance, however, when parameters were more finely tuned it performed worse than the baseline model with SMOTE when tested on the unseen data. This could be due to the natural tendency that decision trees have of over-fitting once their parameters have been fine tuned. Even with max depth of the tree set to 3 it seems that it wasn't enough on the validation dataset. The results of the best runs is presented in Table 2.

Model	Criterion	Max Depth	ANOVA	CV(F_1-score)	Test(F_1-score)
Baseline	Entropy	None	NA	0.834	0.5
SMOTE	Entropy	None	NA	0.73	0.8
ANOVA + SMOTE	Gini	3	Top 50	0.849	0.6

Table 2: Best parameters and F_1-scores for each experiment over 5 fold CV. Best value for min_samples_split always 2 and min_samples_leaf equal to 1.

K-Nearest Neighbors In contrast to decision trees, the second model we tested was k-NN, a non-parametric classification method. Since it is a distance-based method and there is a large difference in range between the various features (Figure 1), it will be wise to apply a scaling method beforehand. Yet, as for the first model, a base model was run without any pre processing, just as a benchmark, followed by k-NN with both a standard scaler and ANOVA feature selection. It is clear from Table 3 that the fine tuned k-NN provides better results all than previous decision trees. By always selecting the closest neighbour (k=1) using an uniform euclidean distance it manages to achieve a 0.971 F_1-score in cross validation and 0.889 in the validation dataset (by miss-classifying one patient as having AML).

Gaussian Naive Baynes and Support Vector Classifier We decided to test two more models to have a more varied set of classifiers to choose from for our ensemble model. Gaussian Naive Baynes is as the name suggests, based on Baynes theorem, however in this case the likelihood of the features is assumed to be Gaussian. The choice for using it were mainly based on its simplicity and robustness. Support Vector machines are another popular classifier that finds an optimal boundary between the possible outputs. We choose to use a SVC for its good performance on datasets where the number of features is very high, specially when compared to the number of samples. A standard scaler, followed by ANOVA feature selection and then SMOTE was always

Model	K	Distance	ANOVA	CV(F_1-score)	Test(F_1-score)
Baseline	1	Euclidean	NA	0.615	0.571
SMOTE	17	Euclidean	NA	0.835	0.833
Scaler + ANOVA + SMOTE	1	Euclidean	Top 50	0.971	0.889

Table 3: Best parameters and F_1-score for each experiment over 5 fold CV. Best value for min_samples_split always set to 2 and min_samples_leaf equal to one.

applied to both models. In the table below we present the best results of each one of the classifiers. Both models performed well on cross validation and both models correctly labeled all patients in the test dataset.

3A2.2b - Ensemble

Our ensemble classifier By combining the best aspects of different classifiers we can end up with an overall better model. To make use of this we've combined three of our best models: k-NN, SVC and Gaussian Naive Baines. By combining all three equally well performing models and using a voting classifier to get a majority vote we can balance out each classifier individual weaknesses and have an overall more accurate final model. The voting scheme was set to *hard*, meaning that the predicted label is the majority of the class labels predicted by each individual classifier. The ensemble classifier performed very well on the test dataset (36 data points), with a F_1-score of 1.0, all the labels were predicted correctly.

Random Forest To compare the performance of our own ensemble classifier we used another ensemble method. Random forests fit a number of decision tree classifiers on various sub-samples of the dataset and by using their average, it improves the predictive accuracy and reduces over-fitting. The best performing random forest we tested had an average F_1-score of 0.876 during cross-validation and just as our ensemble classifier, it got all instances of the testing dataset correct with an F_1-score of 1.0.

Gradient Boosting Classifier Just Like Random Forests, Gradient Boosting is also a set of decision trees. But while Random Forests build each tree independently, this type of classifier builds one tree at a time. The results are then compiled along the way in an additive way (ensemble), instead of being only combined at the end of the process. Gradient Boosting tends to perform better on unbalanced data while Random Forest excel at problems with noise in the data. By running an extensive grid search we got a final model that has an average F_1-score of 0.905 during cross-validation, and perfectly predicted all the labels in the test dataset.

3A2.2c - Summarization of all experiments

After running and measuring the performance of all of the models mentioned in the report its clear that ensemble models outperform single classifiers. Through the combination of similar performing models we can create a ensemble classifier that balances out each individual weaknesses. Another important conclusion is the importance of pre-processing. From the baseline experiments it was clear that SMOTE plays a critical role in the performance of the model, given the high imbalance and small amounts of data from the problem at hand. Scaling is also of extreme importance in methods based on distance functions like k-NN. The best performing model we got was a a Gradient Boosting Classifier, which is an ensemble method that has a recent history of being a great performer, specially in cases with limited samples. The final predictions CSV file (using the Gradient Boosting Classifier) can be found in the repository as **Team_3_prediction**.
All the detailed strategies and results can be visualized in Figure 1 as well as the best settings found for each model using grid-search.

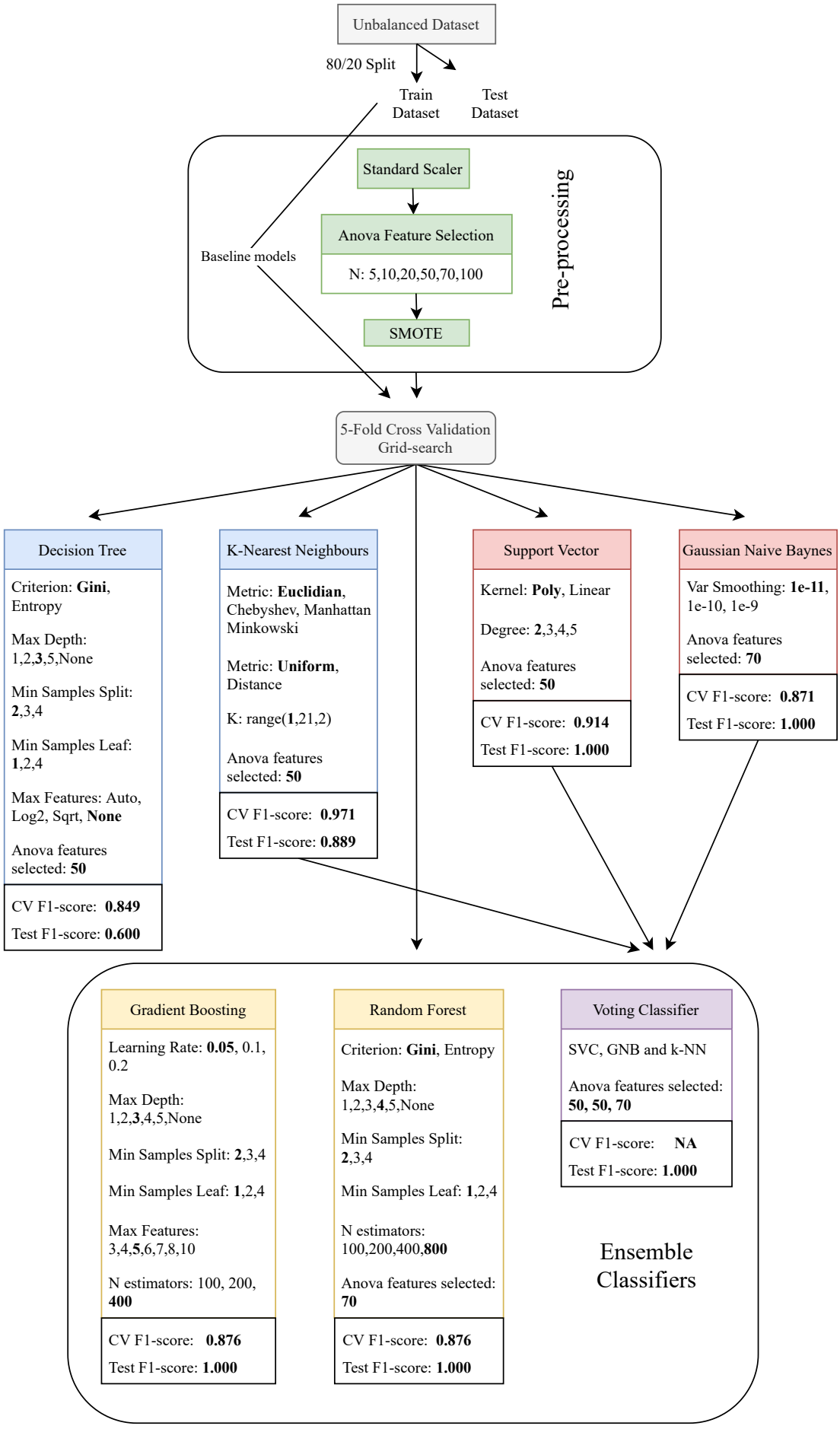


Figure 11: Schematic of experiments performed. The best parameter values are represented in **bold** as well as their respective F_1-scores