Assignment 3

Rik Vegter (s3147495)
Jordi Timmermans (s3870758)
Hari Vidharth (s4031180)
Ayça Avcı (s4505972)

Group 12

October 8, 2020

1 Pen & Paper

(a) We have 4 times + and 5 time - which gives:
$Entropy = -(\frac{4}{9} * log_2(\frac{4}{9} + \frac{4}{9} * log_2(\frac{4}{9}) \approx 0.991$

(b) **Calculating entropy and information gain for a_1**
We have 4 times T (3 times +, 1 time −) and 5 times F (1 time + and 4 times −). From this follows:
$Entropy = \frac{4}{9} * -(\frac{3}{4} * log_2(\frac{3}{4}) + \frac{1}{4} * log_2(\frac{1}{4})) + \frac{5}{9} * -(\frac{1}{5} * log_2(\frac{1}{5}) + \frac{4}{5} * log_2(\frac{4}{5})) \approx 0.762$
The information gain is approximately $0.991 - 0.762 = 0.229$
Calculating entropy and information gain for a_2
We have 5 times T (2 times + and 3 times − and 4 times F (2 times + and 2 times −). From this follows:
$Entropy = \frac{5}{9} * -(\frac{2}{5} * log_2(\frac{2}{5}) + \frac{3}{5} * log_2\frac{3}{5}) + \frac{4}{9} * -(2 * \frac{2}{4} * log_2(\frac{2}{4})) \approx 0.984$
The information gain is approximately $0.991 - 0.984 = 0.007$

(c) We first sort the values in ascending order in order to see where splits can happen:

a_3	Target Class
1.0	+
3.0	−
4.0	+
5.0	−
5.0	−
6.0	+
7.0	+
7.0	−
8.0	−

Table 1: a_3 in ascending order

We split the instances based on the value of $\frac{a_{3_i} + a_{3_{i+1}}}{2}$ if a_{3_i} and $a_{3_{i+1}}$ are not equal. Entropy for split at 2: $\frac{1}{9} * -1 * 0 + \frac{8}{9} * -(\frac{3}{8} * log_2(\frac{3}{8}) + \frac{5}{8} * log_2(\frac{5}{8})) \approx 0.848$
Information gain for split at 2: $0.991 - 0.848 = 0.143$
Entropy for split at 3.5: $\frac{2}{9} * -(2 * \frac{1}{2} * log_2(\frac{1}{2}) + \frac{7}{9} * -(\frac{3}{7} * log_2(\frac{3}{7}) + \frac{4}{7} * log_2(\frac{4}{7})) \approx 0.989$
Information gain for split at 3.5: $0.991 - 0.989 = 0.002$
Entropy for split at 4.5: $\frac{3}{9} * -(\frac{2}{3} * log_2(\frac{2}{3}) + \frac{1}{3} * log_2(\frac{1}{3})) + \frac{6}{9} * -(\frac{4}{6} * log_2(\frac{4}{6}) + \frac{2}{6} * log_2(\frac{2}{6})) \approx 0.918$
Information gain for split at 4.5: $0.991 - 0.918 = 0.073$
Entropy for split at 5.5: $\frac{5}{9} * -(\frac{2}{5} * log_2(\frac{2}{5}) + \frac{3}{5} * log_2(\frac{3}{5})) + \frac{4}{9} * -(2 * \frac{2}{4} * log_2(\frac{2}{4})) \approx 0.984$
Information gain for split at 5.5: $0.991 - 0.984 = 0.007$
Entropy for split at 6.5: $\frac{6}{9} * -(2 * \frac{3}{6} * log_2(\frac{3}{6}) + \frac{3}{9} * (\frac{1}{3} * log_2(\frac{1}{3}) + \frac{2}{3} * log_2(\frac{2}{3})) \approx 0.973$
Information gain for split at 6.5: $0.991 - 0.973 = 0.018$
Entropy for split at 7.5: $\frac{8}{9} * -(2 * \frac{4}{8} * log_2(\frac{4}{8})) + \frac{1}{9} * 0 \approx 0.889$
Information gain for split at 7.5: $0.991 - 0.889 = 0.102$

(d) classification error rate $a_1 = \frac{4}{9} * (1 - max(\frac{3}{4}, \frac{1}{4}) + \frac{5}{9} * (1 - max(\frac{1}{5}, \frac{4}{5})) = 0.222$
classification error rate $a_2 = \frac{5}{9} * (1 - max(\frac{2}{5}, \frac{3}{5}) + \frac{4}{9} * (1 - max(\frac{2}{4}) = 0.444$
The error rate of a_1 is lower, so a_1 is the preferred split according to the classification error rate.

(e) gini index $a_1 = \frac{4}{9} * (1 - (\frac{3}{4}^2 + \frac{1}{4}^2)) + \frac{5}{9} * (1 - (\frac{1}{5}^2 + \frac{4}{5}^2)) \approx 0.344$
gini index $a_2 = \frac{5}{9} * (1 - (\frac{2}{5}^2 + \frac{3}{5}^2)) + \frac{4}{9} * (1 - (2 * \frac{2}{4}^2) \approx 0.436$
The lower the gini index, the better the split. a_1 thus has the most favorable split.

2 Classification in Practice

Introduction

Adult acute myeloid leukaemia (AML) is cancer which affects the blood and the bone marrow. This type of cancer is very risky for older people when not treated immediately. This is why potential AML patients must get diagnosed as soon as possible. With the rise of machine learning, new possibilities have come to light for diagnosing patients.

Many recent studies present methods to diagnose patients. Examples of these diagnoses are diagnoses of people who potentially have heart disease. Many heart diseases can be diagnosed using data mining [1]. For this type of machine learning, large data sets are necessary. In this study, we analyse the data set of DREAM6/FlowCAP2 Molecular Classification of Acute Myeloid Leukemia (AML) Challenge, 2011. A part of this data was presented to our research group.

Data set

The data consists of 359 subjects of which 186 characteristic medical features were extracted. Of these 359 subjects, from 179 subjects it is known whether they have AML or not. Of the other 180 subjects, it is not known whether they are diagnosed with AML. However, it is known that 20 of these 180 patients are diagnosed with AML. Out of the 179 labelled subjects, 23 are diagnosed with AML. That means that the data set is quite imbalanced and that preprocessing methods might be necessary. The goal of this study is to create a model that can classify unlabelled patients correctly.

Methodology

Descriptive and Exploratory Analysis

Before the classification phase, to get an idea about features of the dataset and how to proceed in the classification process, eigenvalue decomposition and ANOVA techniques are used on the dataset to see which and how many features are more promising and explains a certain percentage of variance in the dataset. In figure 1, it shows that the first 30 components explain almost all the variance. Especially, the first 6 components, explained 92.72% of the variance. This means that for example the 6 most explaining principal components might be more promising in tackling the problem than all of the data. When all data are used, the complexity of the problem goes up, while most data do not even add information.
ANOVA has been used to see which features are the most important. The 3 most important and 3 least important features are plotted in the boxplots in the appendix. Feature 11, 133 and 134 represent the top features while feature 16, 73, 110 being the less significant ones.

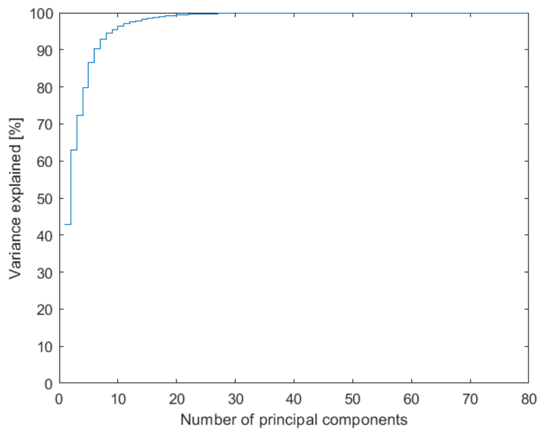

Figure 1: Variance Explained

To get a holistic view of the data, we plotted the first 2 principal components from PCA. Figure 2 reveals that dataset is imbalanced since it has more AML patients labelled than healthy people. The class distribution is thus imbalanced, which might lead to a poor classification performance with traditional machine learning models. Additionally, the data points of the different classes are not well separated when projected onto their first 2 principal components. This makes it such that linear classifiers might not perform well on the data. We should thus choose wisely which classifier we are going to use.

To see whether the distribution of the train and test data are equal, we applied the Wilcoxon rank-sum test to every feature. The test with a 5 percent significance level failed to reject the null hypothesis for every feature, Hence, there is no considerable difference in the distribution between the training and test data.

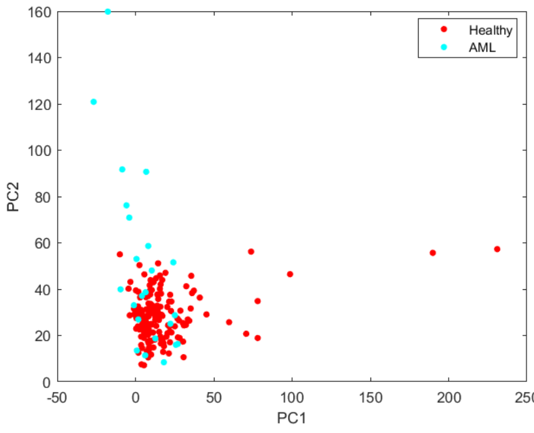

Figure 2: Data projected onto the first 2 principal components

To investigate the impact of different preprocessing on the data, we used the non-linear embedding tSNE. As preprocessing methods we used Z-score normalization, PCA with the first 20 principal components and ANOVA using the 20 best features which led to the figures 3 to 6. ANOVA seems to have positive effects on classification because data is well-separated according to their labels compared to other preprocessing techniques. PCA might have a slight positive effect since data seems decently well-separated according to their labels compared to training data itself. Applying the z-score transform will probably not have any positive effect on classification.

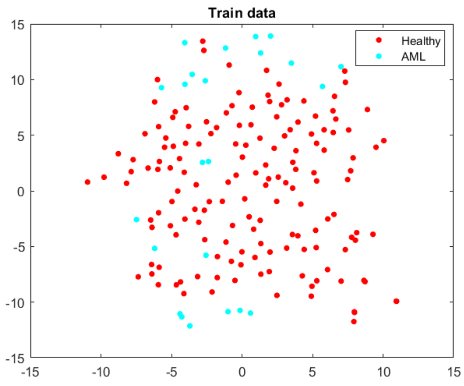

Figure 3: Train Data

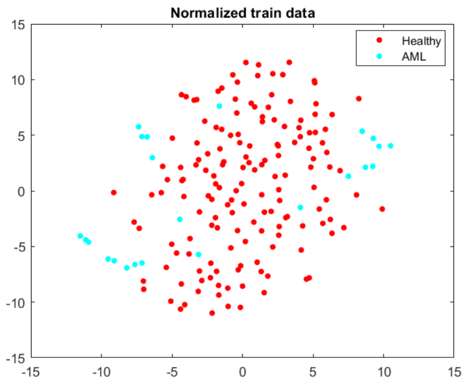

Figure 4: Normalized Train Data

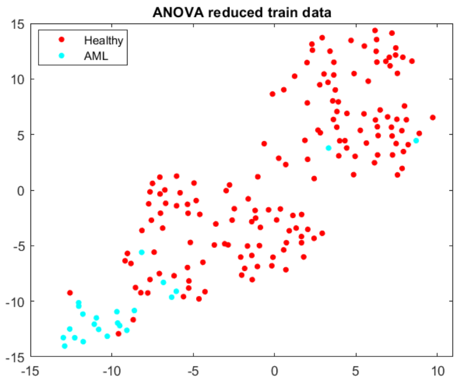

Figure 5: ANOVA Reduced Train Data

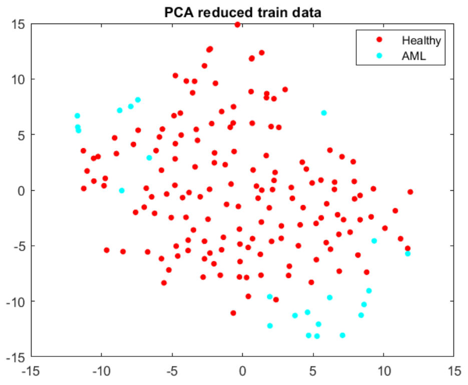

Figure 6: PCA Reduced Train Data

Preprocessing

Since the data that our classifier will be trained on is highly imbalanced, some preprocessing methods are used which are described below.

One method to balance the data set is by simply doubling the inferior subset of the training data. This means that to balance the data set we can duplicate all subjects in our data set which are labelled as an AML patient. This process can be repeated until good results of the classifiers are acquired. What should be taken into account is when duplicating data of subjects, the model might become biased towards the training data, hence not performing well on unseen data. This effect is enhanced every time the inferior data class is duplicated

Another approach to balance an imbalanced dataset is to oversample the minority class. The method mentioned above is duplicating the data from the minority class n times to create more samples. But this does not provide new information to the model. One such method that generates new information by oversampling the minority class is Synthetic Minority Oversampling Technique(SMOTE). SMOTE works by sampling data from the minority class and placing the new point within the region of the selected existing points. The number of points to be selected and the sampling strategy can be specified as a parameter. For our experiments, we have taken the K-Neighbours value in the range of 1 to 9 and the sampling strategy as ”not majority” which means re-sampling all the classes except the majority class since our data set contains binary classes the minority class will be oversampled. Figure 7 shows an example scatter plot for the dataset to show the difference between with and without preprocessing and the effect SMOTE has on the minority class.

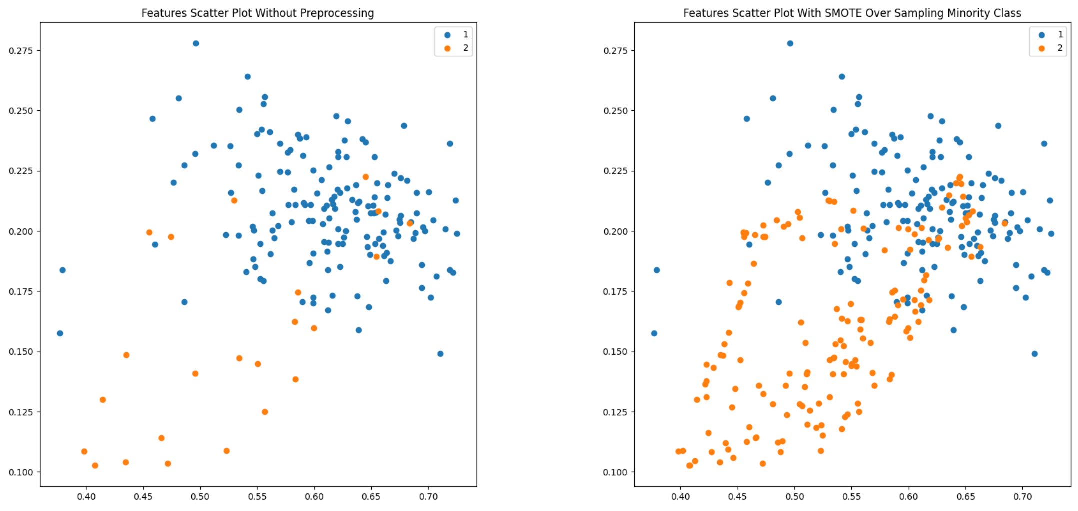

Figure 7: Scatter plot between features X1 and X2 representing the binary class 1 and 2 with and without SMOTE

Classifiers

To classify the unlabelled patients, different classifiers were used to get accuracy as high as possible while taking into account that accuracy is not high due to solely the right classification of the majority class. As a baseline model, we implemented KNN and decision tree classifiers without any preprocessing. Then we experimented with these classifiers using the above mentioned preprocessing methods.

After training models and analyzing the performance of these models, we moved on to ensemble classifiers namely Random Forests, AdaBoost and Gradient Boosting.

For all of these classifiers, hyperparameter tuning was used in a stratified K-Fold cross-validation setting with a value in the range 2 to 10, the stratified method ensures the labels are distributed equally during the train test split preventing the dominance of a class in either the train or the test set. All of the classifiers are going to be tested by evaluating the cross-fold validation accuracy, log loss and the testing confusion matrix.

All the mentioned experiments were performed using python3 programming language and the scikit learn [2] machine learning library.

K Nearest Neighbors Classifier

As mentioned before one model that we are going to train is KNN. The only parameter that was taken into account here is K. This parameter determines how many neighbours need to be considered to determine the class of an unlabelled data point. The considered values of this parameter are K $\in \{1, 3, 5, 7, 9\}$.

Decision Tree Classifier

For the decision tree classifier, the hyperparameters chosen for experimentation are the criteria to measure the quality of the split ”Gini” or ”entropy”, the strategy used to determine the split ”best” or ”random”, the maximum number of features to be considered when looking for a split If “auto”, then max_features=sqrt(n_features), If “sqrt”, then max_features=sqrt(n_features), If “log2”, then max_features=log2(n_features), If None, then max_features=n_features, and finally the class weights ”None” having one set of weights or ”balanced”.

Ensemble Classifiers

To compare the single classifier methods to ensemble classifiers, some ensemble models were implemented.

-Bagging

Random Forest Classifier
A random forest classifier will be trained with hyperparameter tuning. The parameters to be considered are the maximum depth of the forest and the number of trees that are in the forest. For the maximum depth we considered max depth$in\{1, 2...25\}$. For the number of trees, the values from 1 to 150 were considered. All possible combinations are going to be analysed and the best ones will be presented in the results.

-Boosting

AdaBoost Classifier
For the AdaBoost classifier the hyper parameters chosen for experimentation are base estimator as decision tree classifier, number of estimators $\in 10, 50, 100, 200$, learning rate $\in 0.01, 0.1, 1$, and finally the algorithm ”SAMME” or ”SAMME.R”.

Gradient Boosting Classifier
For the Gradient Boosting classifier the hyper parameters chosen for experimentation are loss to be optimized ”deviance” or ”exponential”, number of estimators $\in 10, 50, 100, 200$, learning rate $\in 0.01, 0.1, 1$, the criteria to measure the quality of the split ”friedman_mse” or ”mse” or ”mae”, and finally the maximum number of features to be considered when looking for a split If “auto”, then max_features=sqrt(n_features), If “sqrt”, then max_features=sqrt(n_features), If “log2”, then max_features=log2(n_features), If None, then max_features=n_features.

Results

Before we present the results we want to mention that besides the results shown below, we also tried other classifiers, namely, SVM, Naive Bayes and the extra trees classifier but the performance of these classifiers were not as expected compared to the ones below so the results were not presented. We also tried feature selection with ANOVA based on the results from the previous section but there was no significant improvement in the performance of the classifiers and hence those results are not presented as well.

K Nearest Neighbors Classifier

Without preprocessing 10-fold cross-validation returns an accuracy of 82.5 % when using KNN (K = 3) after parameter tuning. The confusion matrix is shown in table 2.

When considering the preprocessing method of duplicating the minority data class, the best results are obtained when octupling the minority data class. This provides us with 184 AML subjects and 156 healthy subjects. KNN found its best accuracy when using a value of K = 1. Using 10-fold cross-validation this returned an accuracy of 91.7%. The confusion matrix is shown in table 2. As we can see from this confusion matrix only two true negatives were found while no false positives were observed.

KNN	Preprocessing method	K-fold cross validation	Best Hyper parameters	Log loss	Testing accuracy	Test confusion matrix
Without preprocessing	-	10	K = 3	32.620	0.825	[[14 1] [2 0]]
With preprocessing	Octupling data	10	K = 1	13.206	0.917	[[18 2] [0 13]]

Table 2: Best results of the experiments of KNN with and without preprocessing

Decision Tree Classifier

Table 3 shows the results of the decision tree classifier with and without preprocessing. Out off all the experiment settings mentioned above the best classifier results have been presented in the table. The SMOTE prepossessing help improve the performance further of the best result without the preprocessing.

Decision Tree	SMOTE K-Neighbors	K-Fold Cross Validation	Best Hyper Parameters	Training Accuracy	Testing Accuracy	Log Loss	Testing Confusion Matrix
Without Preprocessing	-	5	{'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': None, 'splitter': 'random'}	1	0.944	30.702	[[16 0] [1 1]]
	-	2	{'class_weight': None, 'criterion': 'gini', 'max_features': None, 'splitter': 'random'}	1	0.889	30.702	[[15 1] [1 1]]
With Preprocessing SMOTE	5	6	{'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': None, 'splitter': 'random'}	1	1	17.270	[[16 0] [0 16]]

Table 3: Best results from the experiments for decision tree classifier with and without preprocessing.

Ensemble Classifiers

-Bagging

Random Forest Classifier

To compare the single classifiers against ensemble classifiers random forest classifiers were trained. The maximum depth was set to 19 where the number of trees in the forest was set to 100 because this returned the best accuracy. The accuracy was 100 %. The confusion matrix is shown in table 4. The expectation was that the accuracy was going to be low since the training data was duplicated multiple times. During the 10-fold cross-validation, all of the 10 accuracies were 100%.

	Preprocessing method	K-fold cross validation	Best Hyper parameters	Log loss	Testing accuracy	Testing confusion matrix
Random Forest Classifier	Octupling minor data class	10	max depth = 19, number of trees = 99	19.302	1.0	[[32 0] [0 36]

Table 4: Best parameter settings of random forest classifier with the results using cross fold validation and preprocessing

-Boosting

AdaBoost Classifier

Table 5 shows the results of the AdaBoost classifier with and without preprocessing. Out off all the experiment settings mentioned above the best classifier results have been presented in the table. The SMOTE prepossessing help improve the performance further of the best result without the preprocessing the other parameters remain the same as seen in the table but the n_estimators have increased, we believe the reason behind this may be due to the increase in features by the introduction of the synthetic data.

AdaBoost	SMOTE K-Neighbors	K-Fold Cross Validation	Best Hyper Parameters	Training Accuracy	Testing Accuracy	Log Loss	Testing Confusion Matrix
Without Preprocessing	-	2	{'algorithm': 'SAMME.R', 'learning_rate': 1, 'n_estimators': 10}	1	0.944	30.702	[[16 0] [1 1]]
With Preprocessing SMOTE	1	2	{'algorithm': 'SAMME.R', 'learning_rate': 1, 'n_estimators': 50}	1	1	17.270	[[16 0] [0 16]]

Table 5: Best results from the experiments for adaboost classifier with and without preprocesing.

Gradient Boosting Classifier

Table 6 shows the results of the Gradient boosting classifier with and without preprocessing. Out off all the experiment settings mentioned above the best classifier results have been presented in the table. The SMOTE prepossessing help improve the performance further of the best result without the preprocessing.

Gradient Boosting	SMOTE K-Neighbors	K-Fold Cross Validation	Best Hyper Parameters	Training Accuracy	Testing Accuracy	Log Loss	Testing Confusion Matrix
Without Preprocessing	-	2	{'criterion': 'mae', 'learning_rate': 0.1, 'loss': 'deviance', 'max_features': 'log2', 'n_estimators': 100, 'warm_start': True}	1	0.944	30.702	[[16 0] [1 1]]
With Preprocessing SMOTE	1	2	{'criterion': 'friedman_mse', 'learning_rate': 1, 'loss': 'deviance', 'max_features': 'log2', 'n_estimators': 100, 'warm_start': True}	1	1	17.270	[[16 0] [0 16]]

Table 6: Best results from the experiments for gradient boosting classifier with and without preprocesing.

Discussion

Table 7 summarizes the best performing classifiers out of all the experiments conducted ranked in the order of performance on the unlabelled data.

Classifier	Preprocessing	K-Fold Cross Validation	Best Hyper Parameters	Testing Accuracy	Log Loss
Gradient Boosting	SMOTE	2	{'criterion': 'friedman_mse', 'learning_rate': 1, 'loss': 'deviance', 'max_features': 'log2', 'n_estimators': 100, 'warm_start': True}	1.0	17.270
AdaBoost	SMOTE	2	{'algorithm': 'SAMME.R', 'learning_rate': 1, 'n_estimators': 50}	1.0	17.261
Random Forest	Octupling	10	{'max_depth': '19', 'n_estimators': 99}	0.917	19.302
Decision Trees	SMOTE	6	{'class_weight': 'balanced', 'criterion': 'entropy', 'max_features': None, 'splitter': 'random'}	1.0	17.269
KNN	Octupling	10	K=1	1.0	13.206

Table 7: Summary of top 5 classifiers with the best performance.

From all of the results presented we conclude that the ensemble classifiers perform better than the single classifiers. The best performing single classifier with SMOTE was the decision tree which yielded a testing accuracy of 1.0 which was tested with cross-fold validation. Without preprocessing due to the highly imbalanced nature of the dataset we did observe slight overfitting where the classifiers favoured the majority class. Even though the accuracy values seemed decent the confusion matrix proved this point which was a criterion taken into consideration during the final evaluation of the model performance. This overfitting was overcome by the preprocessing methods which provided a balance between the classes.

When comparing the different ensemble methods, the results were similar to a random forest classifier with an accuracy of 1.0 and boosting classifiers with the same accuracy as well. However, the random forest classifier's input was preprocessed with octupling the minor data class while the boosting methods' input was preprocessed with SMOTE. As mentioned before a downside of duplicating data points in a minor data class is that the classifier will become biased to the presented data. Since SMOTE creates synthetic new data points this problem is less significant when using SMOTE. This is why we have chosen our best model to be a boosting ensemble method combined with SMOTE. The results of AdaBoost and Gradient Boosting are the same. Both have an accuracy of 1 on the testing data. Since we have some pre-knowledge of the unlabelled subjects, namely that 20 subjects of the test set should be labelled as an AML patient, we also considered this when choosing our optimal model. Gradient boosting classifies 19 of the unlabelled subjects as AML patients while AdaBoost classifies 18 unlabeled subjects as AML patients. As Gradient boost returns 19 AML patients which is closer to 20 than AdaBoost we conclude and choose our Gradient Boosting model (parameters can be seen in table 6) with SMOTE preprocessing as our best performing model.

Appendix

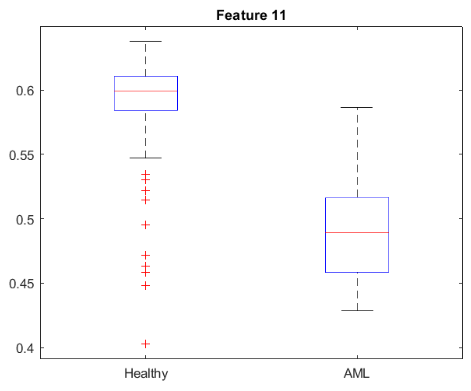

Figure 8: Data distribution of feature 11

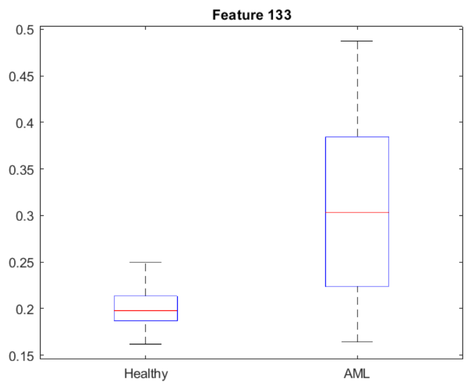

Figure 9: Data distribution of feature 133

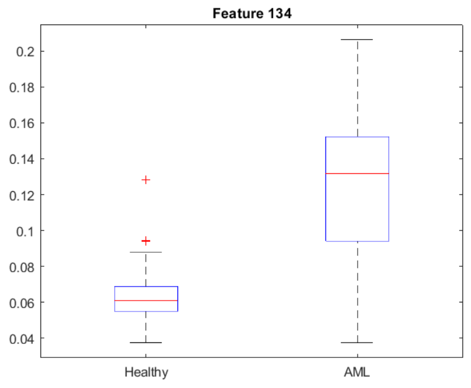

Figure 10: Data distribution of feature 134

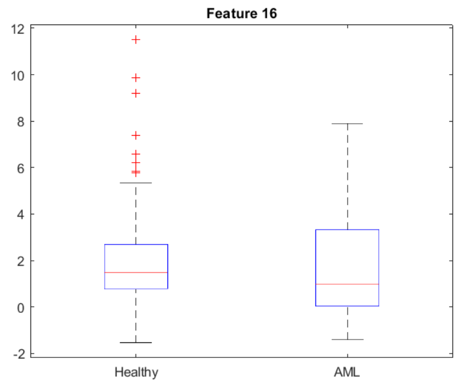

Figure 11: Data distribution of feature 16

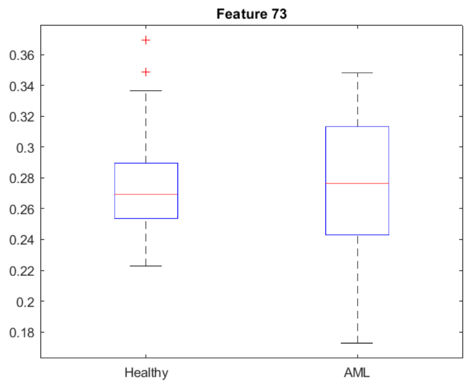

Figure 12: Data distribution of feature 73

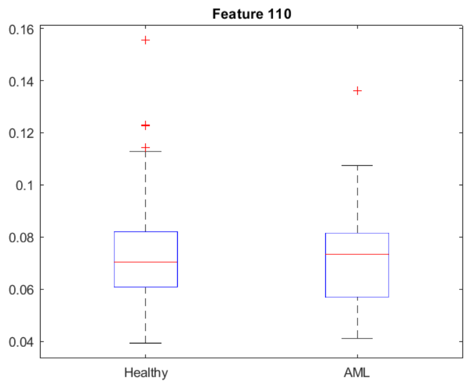

Figure 13: Data distribution of feature 110

References

[1] Chaitrali S Dangare and Sulabha S Apte. Improved study of heart disease prediction system using data mining classification techniques. *International Journal of Computer Applications*, 47(10):44–48, 2012.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

3 Genetic algorithm

a

In the generateInitialPopulation we filled in the following:

```
function pop = generateInitialPopulation(n,ndim)
    pop = [];
    for i = 1:n
        chromosome = randi([0, 1], [1, ndim]);
        pop = [pop;chromosome];
    end
end
```

b

In getOffSpring() we filled in the following:

```
function offspring = getOffSpring(parent1,parent2,mutateprob)
    parentlength = length(parent1);
    split = floor(rand(1,1)*parentlength+1);
    if split == 0
        offspring = parent1;
    elseif split == 6
        offspring = parent2;
    else
        offspring = [parent1(1:split), parent2(split+1:end)];
    end

    for i = 1:length(offspring)
        if rand(1,1) < mutateprob
            offspring(i) = abs(offspring(i) -1);
        end
    end

end
```

c

In getScore() we added in the following line in order to calculate the score:

```
    score = 1*10^4*meanacc + 0.4*sum(chromosome == 0);
```

4 Application: Feature Selection

100 runs have been done to compare the accuracy of KNN with feature selection versus KNN without feature selection. The results can be seen in figure 14. Feature selection as preprocessing thus significantly improves the accuracy of the KNN.

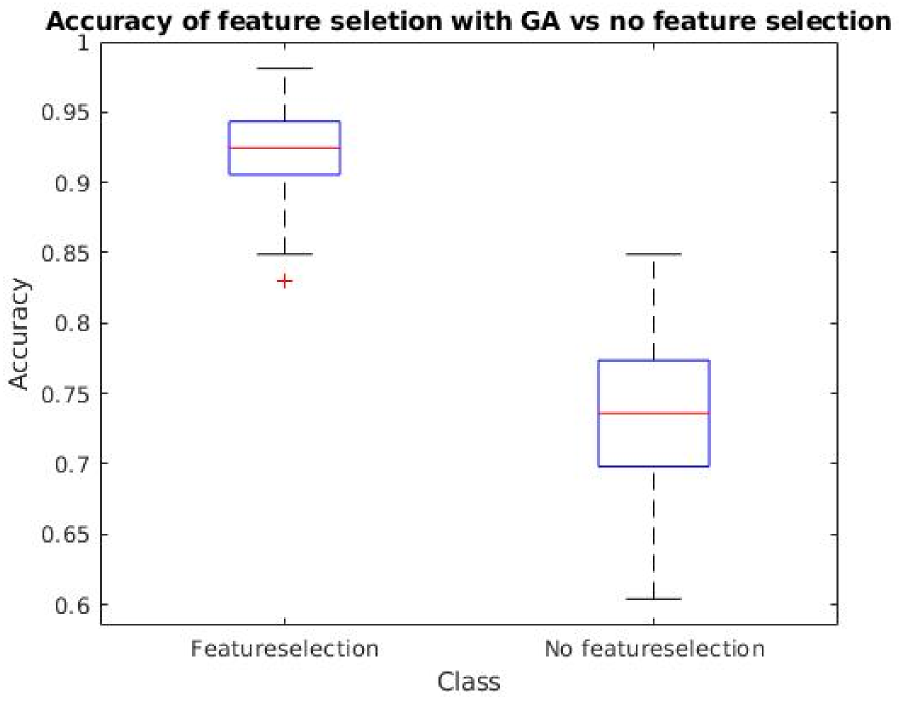

Figure 14: KNN predictions with feature selection vs without feature selection. Feature selection has a median of 0.92, no feature selection has a median of 0.74.