Assignment 5: Classification

Low, Daniel Mark (S3120155)
Petre, Bogdan (S3480941)
Xu, Teng Andrea (S3548120)

Group 7

October 9, 2017

1 Exam: Example Questions (15 P)

1.1 Sorting Answers

The order of the extracted line numbers is: **4, 8, 1, 3, 9, 6.**

1.2 Set of training examples

a) We first create the table of nodes and counts from table 1. c=0 represents the positive class and c=1 represents the negative class.

Table 1: Nodes and counts table

Nodes	Count
c=0	4
c=1	5

Then, we use the formula for entropy and obtain:

$$Entropy = -\frac{4}{9} \cdot \log_2 \frac{4}{9} - \frac{5}{9} \cdot \log_2 \frac{5}{9} = 0.991$$

b) In order to obtain the information gains of an attribute, we must split the parent node into children nodes based on that attribute, calculate the weighted entropy of the children nodes and then subtract it from the parent entropy.

1. **Attribute a1.** Child node 1 (T) has 4 instances, child node 2 (F) has 5 instances.

$$EntropyChildNode1 = -\frac{1}{4} \cdot \log_2 \frac{1}{4} - \frac{3}{4} \cdot \log_2 \frac{3}{4} = 0.811$$

$$EntropyChildNode2 = -\frac{1}{5} \cdot \log_2 \frac{1}{5} - \frac{4}{5} \cdot \log_2 \frac{4}{5} = 0.721$$

$$WeightedEntropy = \frac{4}{9} \cdot 0.811 + \frac{5}{9} \cdot 0.721 = 0.761$$

$$InformationGain = 0.991 - 0.761 = 0.23$$

2. **Attribute a2.** Child node 1 (T) has 5 instances, child node 2 (F) has 4 instances.

$$EntropyChildNode1 = -\frac{2}{5} \cdot \log_2 \frac{2}{5} - \frac{3}{5} \cdot \log_2 \frac{3}{5} = 0.970$$

$$EntropyChildNode2 = -\frac{2}{4} \cdot \log_2 \frac{2}{4} - \frac{2}{4} \cdot \log_2 \frac{2}{4} = 1$$

$$WeightedEntropy = \frac{5}{9} \cdot 0.970 + \frac{4}{9} \cdot 1 = 0.983$$

$$InformationGain = 0.991 - 0.983 = 0.008$$

c) For attribute **a3** there are 6 possible binary splits:

1. $<= 1, > 1$. Information Gain is 0.14
2. $<= 3, > 3$. Information Gain is 0.002
3. $<= 4, > 4$. Information Gain is 0.07
4. $<= 5, > 5$. Information Gain is 0.007
5. $<= 6, > 6$. Information Gain is 0.018
6. $<= 7, > 7$. Information Gain is 0.102

To compute these values we used a Python script, **12c.py**.

d) We need to compute the classification error rates for attributes a1 and a2.

Table 2: a1 error rate

a1	+	-
T	3	1
F	1	4

Table 3: a2 error rate

a2	+	-
T	2	3
F	2	2

Based on tables 2 and 3, the error rates are:

$$Error_{a1=T} = 1 - max(\frac{3}{4}, \frac{1}{4}) = 0.25$$

$$Error_{a1=F} = 1 - max(\frac{1}{5}, \frac{4}{5}) = 0.2$$

$$WeightedError_{a1} = \frac{4}{9} \cdot Error_{a1=T} + \frac{5}{9} \cdot Error_{a1=F} = 0.22$$

$$Error_{a2=T} = 1 - max(\frac{2}{5}, \frac{3}{5}) = 0.4$$

$$Error_{a2=F} = 1 - max(\frac{2}{4}, \frac{2}{4}) = 0.5$$

$$WeightedError_{a2} = \frac{5}{9} \cdot Error_{a2=T} + \frac{4}{9} \cdot Error_{a2=F} = 0.44$$

Based on these calculations, the best split is using a1 because it has the lower error rate.

e) To calculate the Gini indexes for a1 and a2, we will also use 2 and 3.

$$Gini_{a1} = \frac{4}{9} \cdot (1 - (\frac{3}{4})^2 - (\frac{1}{4})^2) + \frac{5}{9} \cdot (1 - (\frac{1}{5})^2 - (\frac{4}{5})^2) = 0.34$$

$$Gini_{a2} = \frac{5}{9} \cdot (1 - (\frac{2}{5})^2 - (\frac{3}{5})^2) + \frac{4}{9} \cdot (1 - (\frac{2}{4})^2 - (\frac{2}{4})^2) = 0.48$$

Based on these calculations, the best split is using a1 because it has the lower Gini index.

2 Classification in Practice

2.1 Descriptive and Exploratory Analysis (25 P)

2.1.1 Investigate the features (5/25P)

The code for this exercise is **2_1_a.py**.

We performed Principal Component Analysis on the training set. 2 features explained 94.1% of the variance. The first component explained 71.4% of the variance and the second component explained 22.7% of the variance.

We separated healthy participants and patients with cancer into two datasets: healthy_dataset = 156 subjects x 189 features; cancer_dataset 23 subjects x 186 features. We then took column by column (i.e., feature-vector by feature-vector) of each dataset and performed a one-way ANOVA between both feature-vectors. We sorted the results by p-value obtained after comparing healthy vs. patient for each vector, and plotted the lowest 5 and highest 5 p-values in Figure 1. The y-axis is not the p-value but the feature-value. The lowest p-values are the most statistically different pairs (i.e., features 133, 10, 132,122,121):

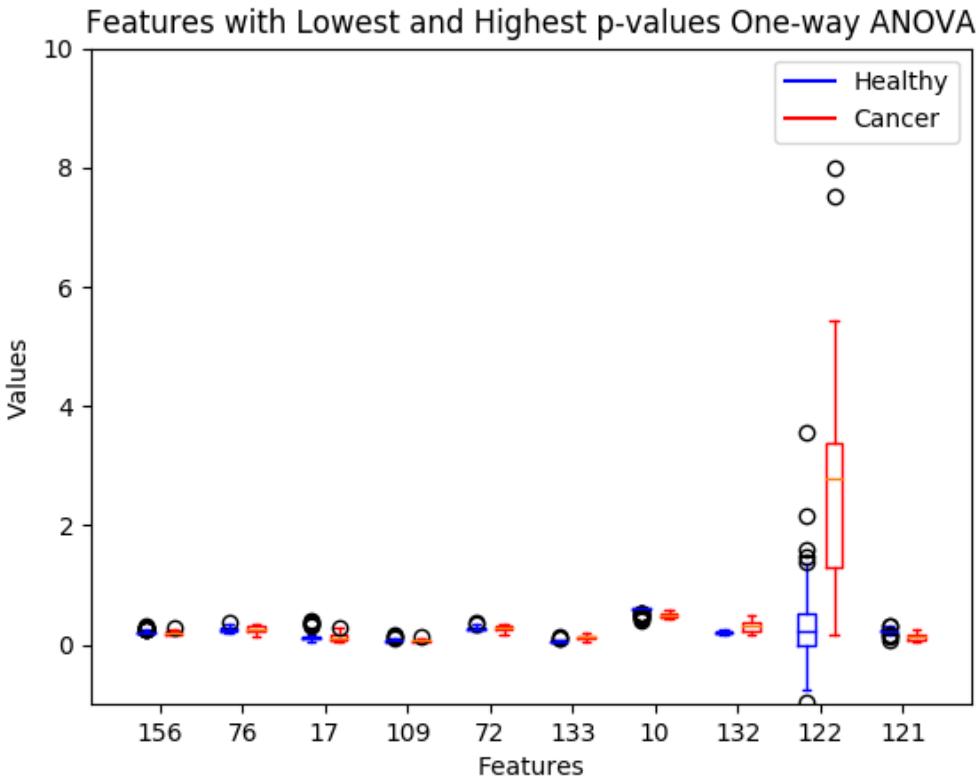

Figure 1: Lowest 5 and Highest 5 p-values.

Feature 122 seems like an extremely different distribution than the rest and may be erroneous data, which makes it difficult to visualize the other distributions. Therefore, in Figure 2, we removed feature 122 and added the next lowest p-value (i.e., the lowest 5 now p-values are features 133, 10, 132,121,125) and plotted again. Here we see that the lowest p-values seem to be able to distinguish between patients and controls: the medians between patients and controls are quite different, whereas they are quite similar for the first (high-p-value) pairs.

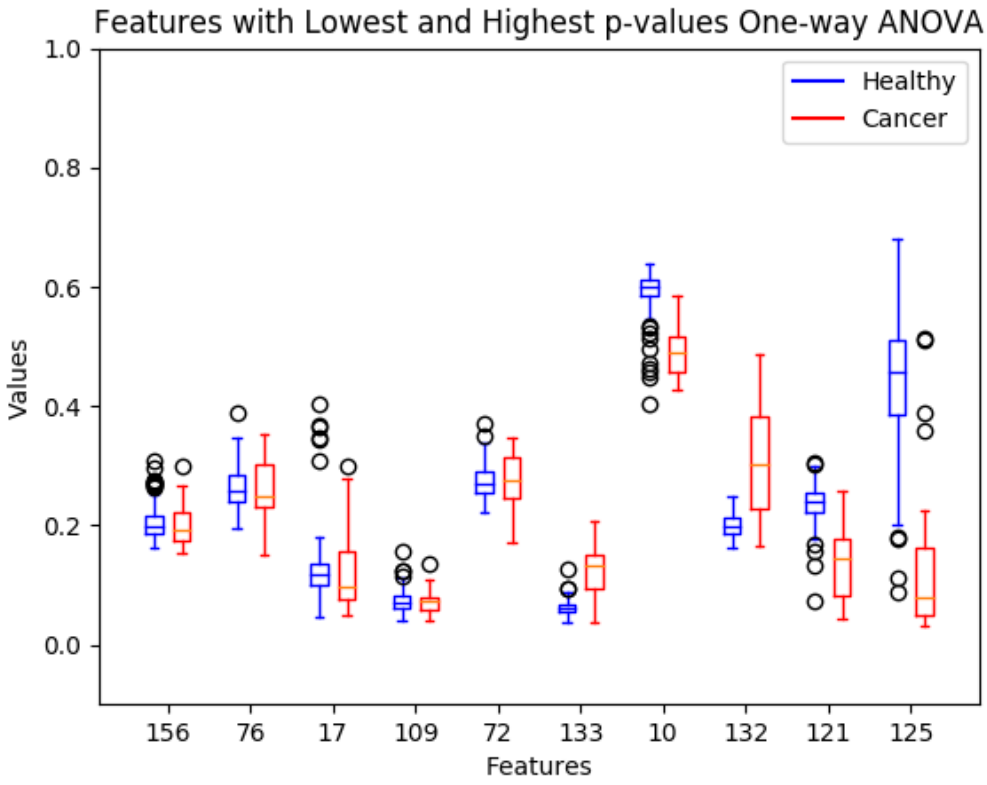

Figure 2: Lowest 5 and Highest 5 p-values.

After Bonferroni correction for multiple comparison, alpha= 0.00027, and the statistically significant features that separate patients from controls are: '133', '10', '132', '122', '121', '125', '137', '183', '6', '40', '36', '7', '38', '8', '182', '91', '120', '181', '1', '136', '90', '2', '95', '123', '53', '48', '87', '94', '176', '14', '37', '3', '174', '134', '180', '167', '52', '178', '124', '89', '81', '4', '85', '162', '129', '166', '5', '0', '135', '79', '185', '141', '39', '83', '50', '99', '93', '26', '68', '49', '168', '27', '117', '16', '57'.

2.1.2 Get a holistic view on the data (5/25P)

The code for this exercise is **2_1_b.py**. Figure 3 plots the first two principal components of the training set using Principal Component Analysis (PCA). Figure 4 plots the first two principal components of the test set.

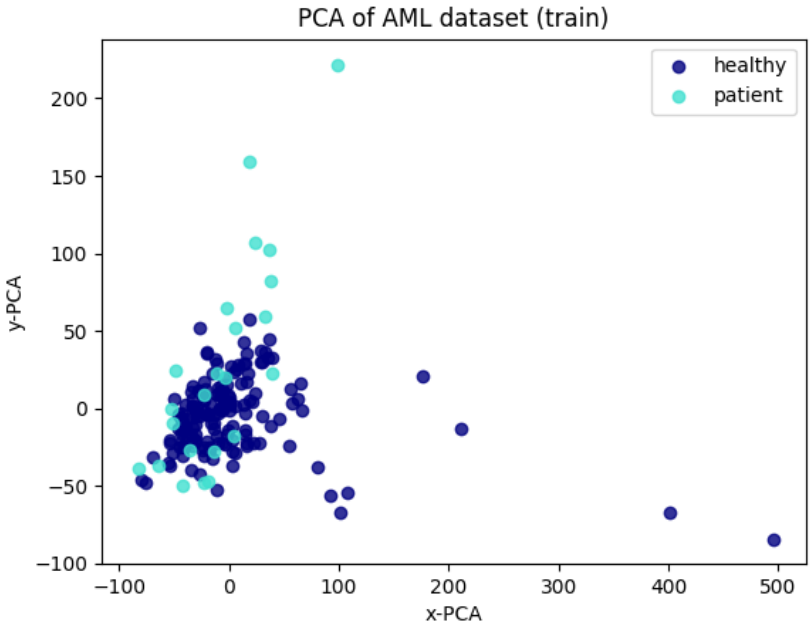

Figure 3: PCA on train dataset before preprocessing.

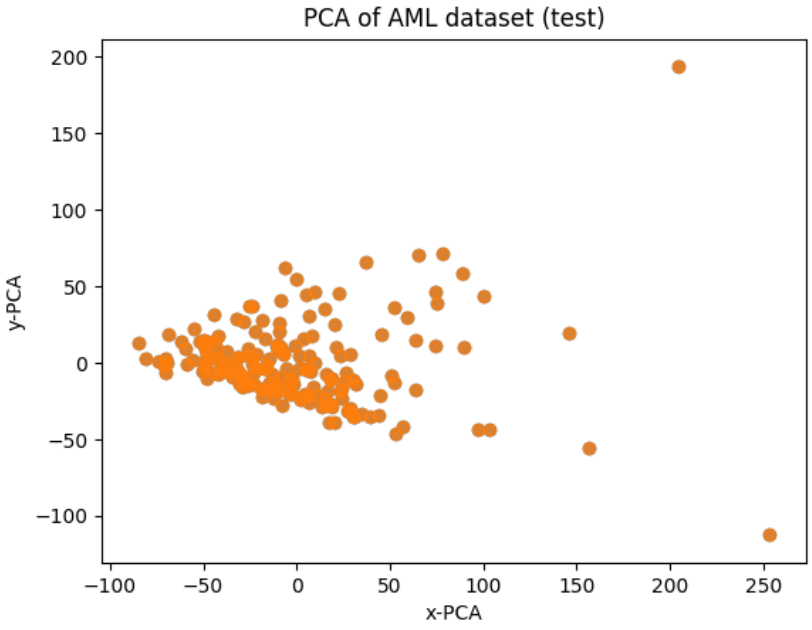

Figure 4: PCA on test dataset before preprocessing.

Figure 5 plots train set using tSNE and Figure 6 plots the test set using tSNE.

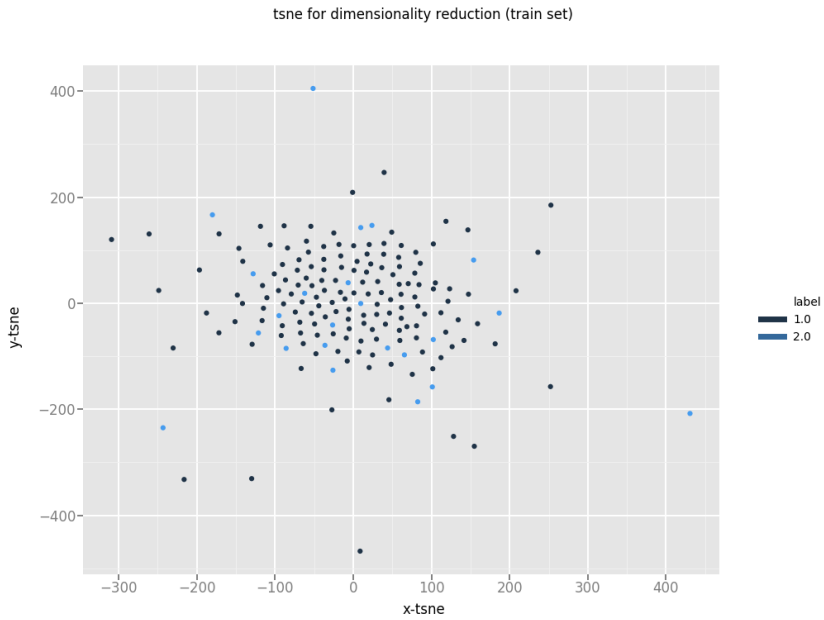

Figure 5: tSNE on train dataset before preprocessing (navy blue are healthy controls and light blue are patients)

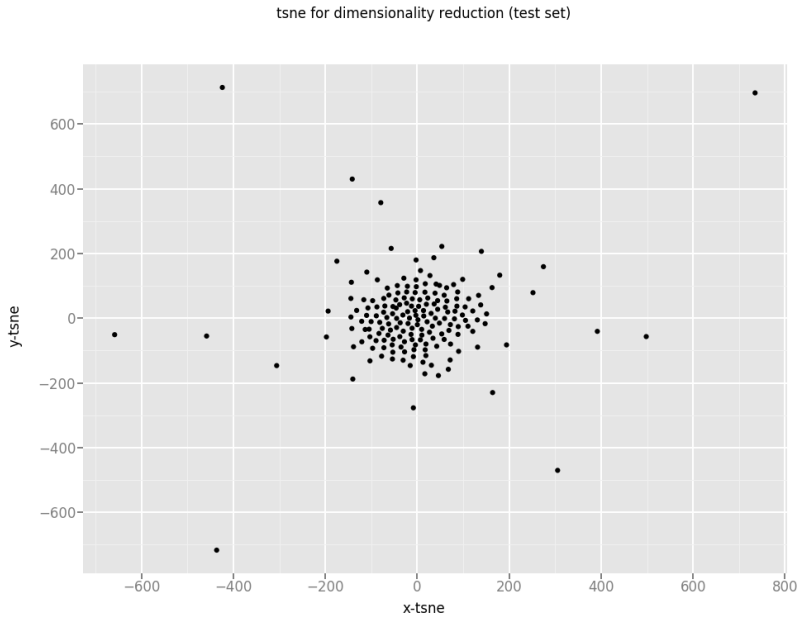

Figure 6: tSNE on test dataset before preprocessing.

These visualizations allow us to see that there are a few cases that are very different than the majority (not necessarily outliers) but these don't belong only to patients, which is a problem. These "outliers" have very different values and distort the representation of both patients and controls.
In PCA of the training set, patients seem to fall near x=0 and have more variance on y, whereas healthy controls seem to fall near y=0 and have more variance on x. This is a patterns that could make it easier to classify. However, on the PCA of the test set, the variance on y is lower, so this separation is somewhat lost.
For the tSNE, we fixed the random seed to 42. When comparing patients and controls in the train-

ing set in the tSNE visualizations, there is no clear pattern separating both populations. They both seem to belong to a normal distribution. For the tSNE visualization, as recommended by the Scitkit-learn documentation for tSNE, we first performed a dimensionality reduction method called TruncatedSVD to reduce the dimensions from 186 to 70 and then performed tSNE on these 70 dimensions.

2.1.3 General impact of preprocessing (15/25P)

The script for this exercise is **2_1_c.py**. We will perform different preprocessing to the original tSNE in Figure 5.

Figure 7 shows a tSNE after z-score transformation. It is not clear from this visualization how the z-score transformation would help classification. There is not a specific pattern to patients vs. controls. Nevertheless, we will apply this transformation because it enables us to better statistically compare two scores that may be from different normal distributions (e.g., patients and controls).

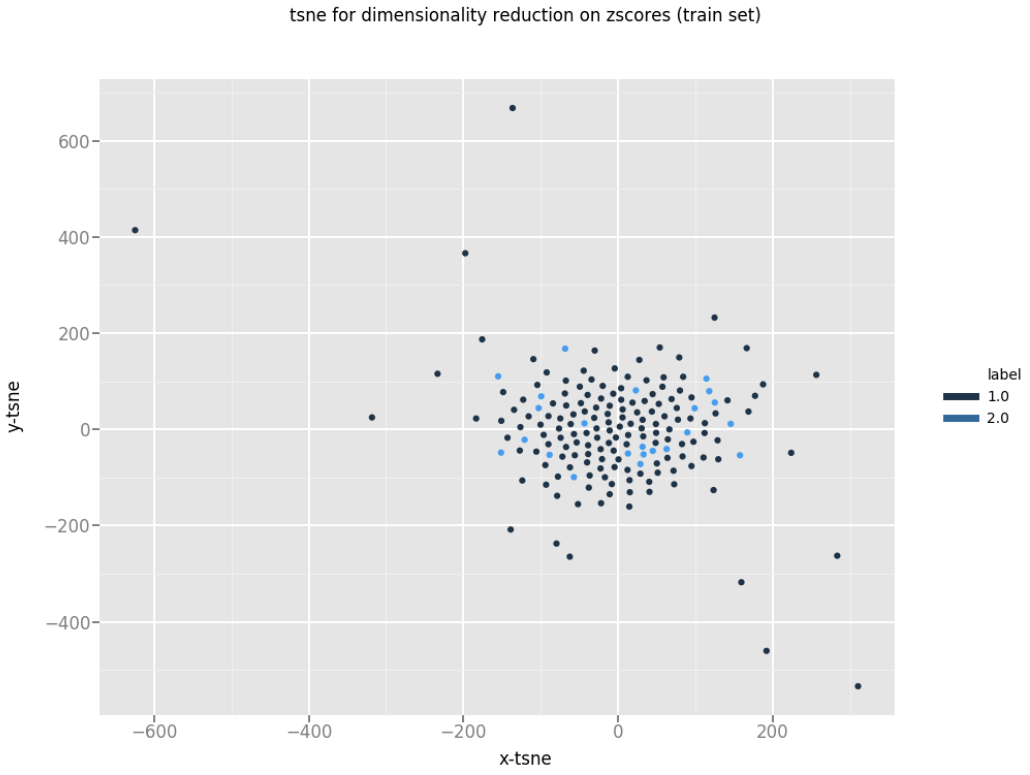

Figure 7: tSNE on train dataset afters zscore transformation (navy blue are healthy controls and light blue are patients)

From exercise 2_1_a, we obtained the features that most significantly separate patients from healthy controls. In Figure 8. we perform tSNE after leaving only the following highly significant features:

'133', '10', '132', '122', '121', '125', '137', '183', '6', '40', '36', '7', '38', '8', '182', '91', '120', '181', '1', '136', '90', '2', '95', '123', '53', '48', '87', '94', '176', '14', '37', '3', '174', '134', '180', '167',

’52’, ’178’, ’124’, ’89’, ’81’, ’4’, ’85’, ’162’, ’129’, ’166’, ’5’, ’0’, ’135’, ’79’, ’185’, ’141’, ’39’, ’83’, ’50’, ’99’, ’93’, ’26’, ’68’, ’49’, ’168’, ’27’, ’117’, ’16’, ’57’.

The data has now been centered and is more normally distributed. There were some subjects before (Figure 7) that seemed like outliers. Now that we took non-relevant features out, these subjects have become more normal. Regarding patients vs. controls, there does not seem to be a major difference visually, except perhaps there seems to be a cluster of patients around y=-75 and x=0, but there are also many healthy controls there as well and patients far from that cluster.

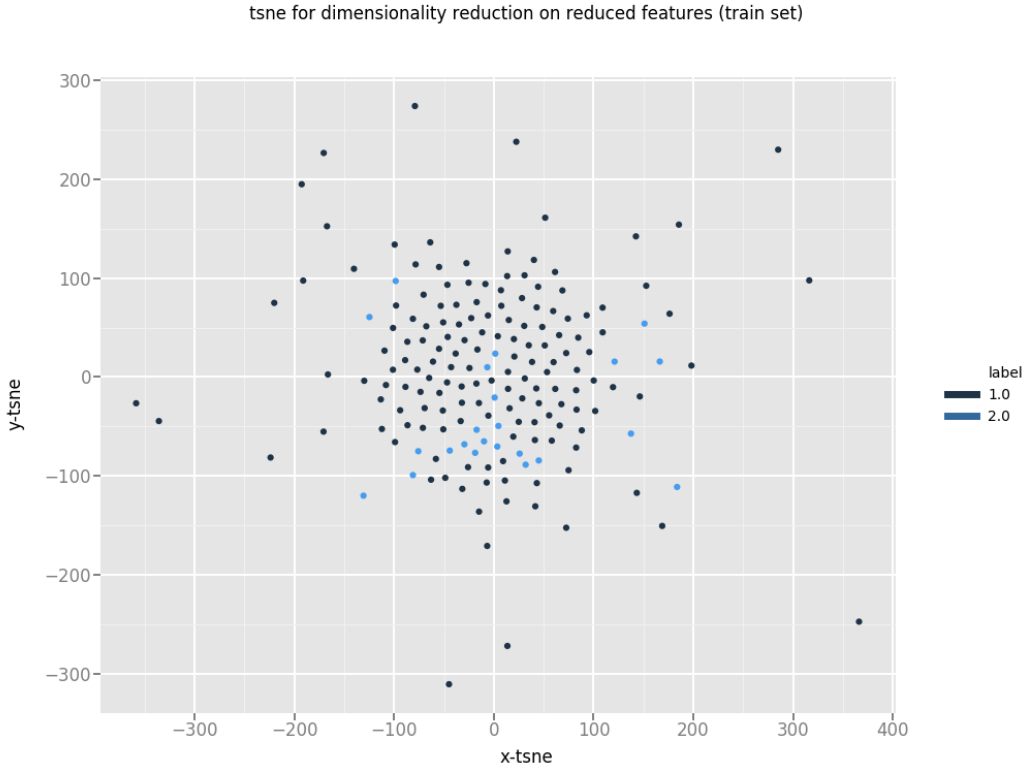

Figure 8: tSNE on train dataset after zscore transformation and using only top 65 statistically significant features (navy blue are healthy controls and light blue are patients)

We tried reducing only the top 48 features and obtained Figure 9, and it seems patients are also a bit clustered around (0,-75).

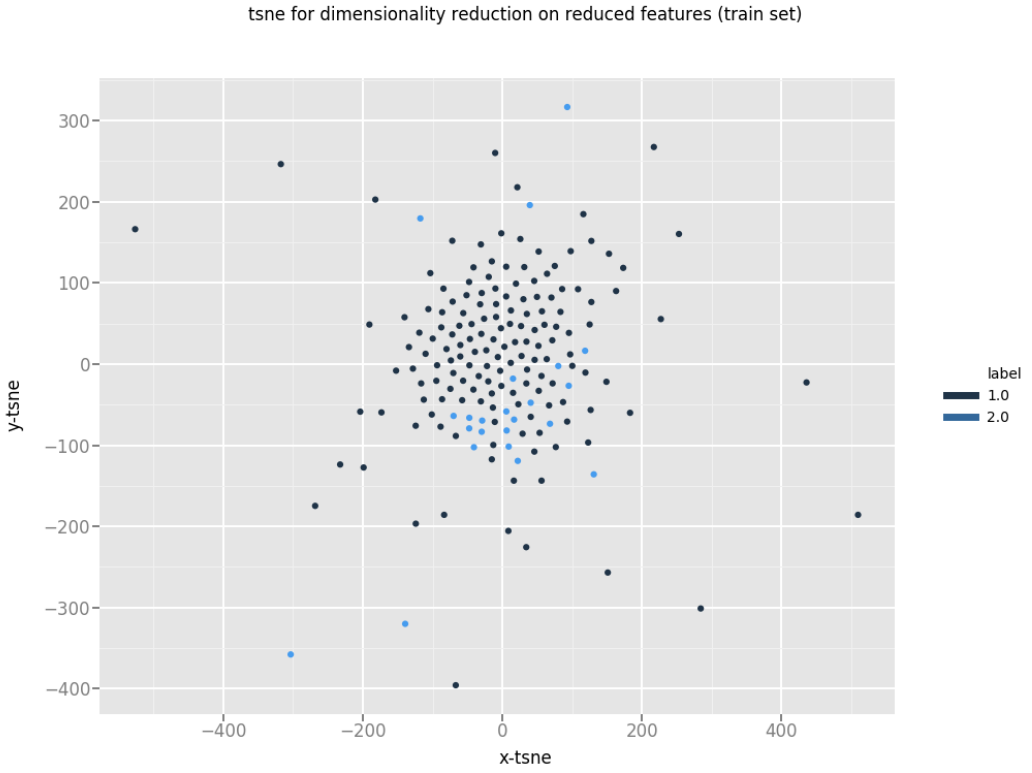

Figure 9: tSNE on train dataset after zscore transformation and using only top 48 statistically significant features (navy blue are healthy controls and light blue are patients)

Therefore, we have two datasets to try: one with the top 65 features (Fig. 9), and one with the top 48 features (Fig. 10).

2.2 Experimentation to find the best prediction (60P)

2.2.1 (a)Base-line experiments

In this exercise we also used Python and the script can be found in **2.2a.py.**
With the information that we had from the exploratory analysis, we had to train the best model in order to predict to which class the unlabeled data belongs. In this exercise we used the well known GridSearchCV from Python to find the best classifier with the best set of parameters performing N Fold Validation. The following steps were taken.

Preprocessing

Before training a model we had some work to do. First of all we generated 2 datasets, one with 48 features and the other one with 65 features according to previous results. The code can be found in **create_65_and_48.py** and the csv's are **train_48.csv** and **train_65.csv**. Finally we had to deal with the imbalanced data since the patient are really few compared to the number of healthy people (23 - 156), so we created another dataset using oversampling technique, obtaining a new dataset which contains a number of sick people as the number of the healthy people.

Classifiers

In this exercise we used three classifiers:

- Decision Trees
- KNN
- Support Vector Machine

For each classifier we declared a list of parameters in order to feed GridSearchCV that it would find the best parameter setting for each classifier:

- Decision Trees parameters : max_leaf_nodes, max_depth, criterion, splitter.
- KNN parameters : n_neighbors, algorithm, weights.
- SVM parameters : C, kernel, gamma.

Training

Divided accordingly our subset of data for the training and the other pair for the test, GridSearchCV automatically performed the 10 Fold Cross Validation and for each classifier fitted the model with the training set.

Results

For each classifier we have a result that consists of:

- percentage with respect to precision, recall and f1_score;
- confusion matrix that summarizes the number of TP,FP,FN,TN;

- Matthew Correlation Coefficient, a good score for binary classification, which uses the following formula: $\frac{TPXTN-FPXFN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
- Normalized Accuracy, a score from sklearn library that tells us how accurate the model was.

Conclusions Base-line experiments

We ran the algorithm 3 times, first of all with all the features, then with 65 and finally with 48. Then we ran the algorithms using oversampling before.

The exploratory analysis that we did before actually helped us a lot because using all the features we obtained for each classifier an average of 0.9 in f1_score. Instead, using 65 features still doesn't improved so much the performance. But when we used just 48 features there was a lot of improvement obtaining an average of 0.94 with peak of 0.96, 0.97. So we did observe **overfitting** with the original set of features and we tackled it by cutting down the number of features.

When we finally implemented the oversampling of the minority class, f1_score for SVD had an average of 0.99, which is a really good result and for the other two classifiers an average of 0.95 for the Decision Tree and 0.96 for the KNN with peak of 1 for every classifier.
So we **confirmed** that the important features could just be 48 and the other features are not really needed, because otherwise our model would perform worse.

2.2.2 (b) Ensemble

The code can be found in **2.2b.py** and **2.2bRF.py**.
In this exercise we used the best models found in 2.2a thanks to GridSearchCV. The best **experimental results** can be found in the code in 2.2b.py when creating the models and giving them the best parameters. We selected these parameters and classifier **because** in the previous script(2.2a.py) we created a list with each combination of parameters with their respective classifier according to GridSearchCV best estimation. From this list we firstly sorted it according to the best score that they gave us and then we took the top 5. It's important to report that Decision Tree Classifier is not in our top5 classifier. Our top5 classifiers include 2 SVM classifiers with different parameters and 3 KNN Classifiers.
After declaring the classifiers, we feed a Voting Classifier, that implements a Majority Voting classifier. For instance, one entry is classified as the majority of classifiers predicted the most (hard voting) or averaging the class-probabilities (soft voting).

Comparison RF vs Ensemble Classifier

We ran the Ensemble Classifier in **2.2b.py** and Random Forest in **2.2bRF.py**.
We can see that Ensemble Classifier that we build is much better than Random Forest with the following average scores:

- F1_score : 0.99
- MatthewCC: 0.97
- Normalized Accuracy: 0.98

Random Forest results are not so exciting because the results are average and sometimes worse than the previous part (a).
Both Classifier run with best dataset structure: 48 features and oversampling.

2.2.3 (c) Summary of our experimnents

A diagram of the experiments we performed is shown in figure 10.

First, we only used the first 179 rows of the dataset because those were the labeled ones. Then, we applied ANOVA to only get the most important features that distinguish patients vs. controls and kept only 65 and then 48 features (from the initial 186 features).

The 2 classes were imbalanced (156 healthy vs 23 patients). To overcome this issue, we oversampled the minority class. We preferred this approach because it uses all information and the number of records is not high enough to significantly reduce performance.

Afterwards, we ran 3 types of classifiers: K Nearest Neighbors, Decision Trees, and Support Vector Machine, each with different settings. We performed 10-fold Cross Validation (70% for training, 30% for testing). To test the performance of these classifiers, we used Matthews Correlation Coefficient and F1 score.

From all these classifiers, we selected the top 5 (table 4) based on these metrics. Using them, we built an ensemble classifier, using voting strategy.

Table 4: Top 5 classifiers

Classifier	Params	Score
SVM	'svm__C': 10.0, 'svm__kernel': 'rbf', 'svm__gamma': 1.0	**0.982**
KNN	'knn__algorithm': 'ball_tree', 'knn__weights': 'distance', 'knn__n_neighbors': 25	0.973
KNN	{'knn__algorithm': 'kd_tree', 'knn__weights': 'distance', 'knn__n_neighbors': 25}	0.973
KNN	{'knn__algorithm': 'brute', 'knn__weights': 'distance', 'knn__n_neighbors': 25}	0.973
SVM	{'svm__C': 1.0, 'svm__kernel': 'rbf', 'svm__gamma': 1.0}	0.964

2.2.4 (d) Predictions

Our predictions were generated by the script **prediction.py** using the ensemble classifier and the most important 48 features, which we also used in training. The index goes from 180 to 359, corresponding to the subjects that were not initially labeled. Our classifier predicted a total of **23** patients.

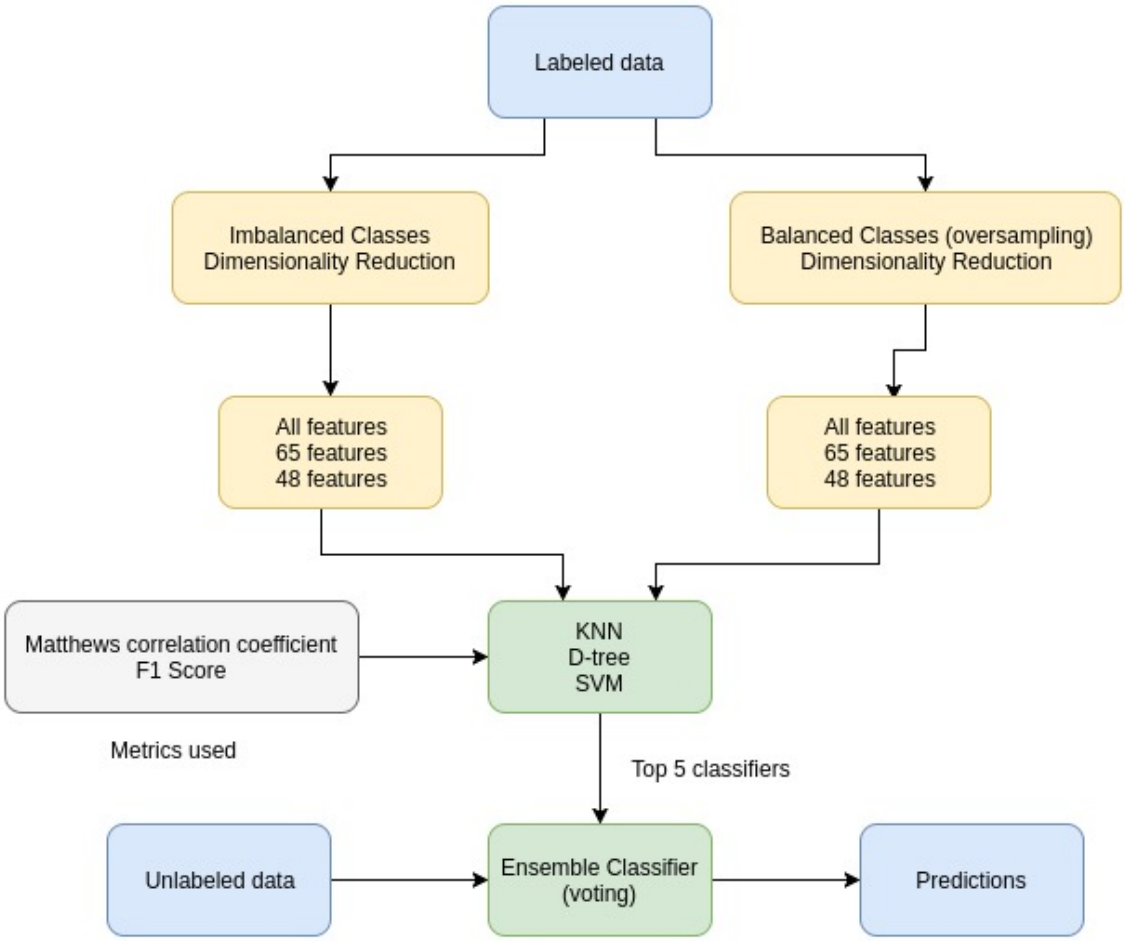

Figure 10: Diagram of performed experiments