[16.09.2021]

Bulky mediastinal lymphoma classification with ML-techniques

Activity status n. 1

Matteo Barbetti

University of Firenze
INFN - Firenze

Objective

Classification of *bulky mediastinal lymphoma* using **Machine Learning** techniques applied to PET and CT images.

Lymphoma types:

1. *Hodgkin* (HL)
2. *Gray Zone* (GZ)
3. *Primary Mediastinal Lymphoma* (PML)

PLANNED

From low-level data

- Data: PET and CT images
- Technique: Image classification
- Algorithm: *Convolutional Neural Nets*

IN PROGRESS

From high-level data

- Data: features drawn by LIFEx from images
- Technique: Binary and multiclass classification
- Algorithm: *Logistic regression* + *Random Forest*

First attempts: *high-level dataset*

Database of 119 rows x 101 columns

- **different data types** → need for homogeneous dataset
  - numbers
  - dates
  - texts
  - intervals
- **dim[*instance space*] ~ dim[*feature space*]** → need for reducing the feature space

Public repository on GitHub to **prepare**, **clean** and **study** the high-level dataset through Jupyter Notebooks:

- `data_preparation` -- data format correction
- `data_visualization` -- correlated features removal
- `binary_classification` -- classification in HL and non-HL
- `multiclass_classification` -- classification in HL, GZ and PML

mbarbetti/LNHunter

`data_preparation`

Feature list

ID
Lymphoma typer (HL =1, GZ = 2, PML =3)
Data nascita
dPET staging
SUVmin (SUV)
SUVmean (SUV)
SUVstd (SUV)
SUVmax (SUV)
MTV (# vx)
MTV (mL)
SMTV (mL/Kg)
TLG (SUV*mL)
STLG (SUV*mL/Kg)
MTV (# vx) TOT
MTV (mL) TOT
SMTV (mL/Kg) TOT
TLG (SUV*mL) TOT
STLG (SUV*mL/Kg) TOT

SHAPE_Volume(mL)
SHAPE_Volume(vx)
SHAPE_Sphericity[onlyFor3DROI])
SHAPE_Surface(mm2)[onlyFor3DROI]
SHAPE_Compacity[onlyFor3DROI]
PARAMS_DistanceOfNeighbours
PARAMS_NumberOfGreyLevels
PARAMS_BinSize
PARAMS_IntensityResampling
PARAMS_BoundsRangeOfValueAfterDiscre
PARAMS_ZSpatialResampling
PARAMS_YSpatialResampling
PARAMS_XSpatialResampling

...ue only for PET or NM)
...etized volume sought
DISCRETIZED_SUVbwpeakSphere1mL(value only for PET or NM)
DISCRETIZED_TLG(mL)[onlyForPETorNM]
DISCRETIZED_HISTO_Skewness
DISCRETIZED_HISTO_Kurtosis
DISCRETIZED_HISTO_ExcessKurtosis
DISCRETIZED_HISTO_Entropy_log10
DISCRETIZED_HISTO_Entropy_log2
DISCRETIZED_HISTO_Energy[=Uniformity]

NGLDM_Coarseness
NGLDM_Contrast
NGLDM_Busyness
GLZLM_SZE
GLZLM_LZE
GLZLM_LGZE
GLZLM_HGZE
GLZLM_SZLGE
GLZLM_SZHGE
GLZLM_LZLGE
GLZLM_LZ...
GLZLM_GL...
GLZLM_ZL...
GLZLM_ZP
TimePosit...
zLocatio...

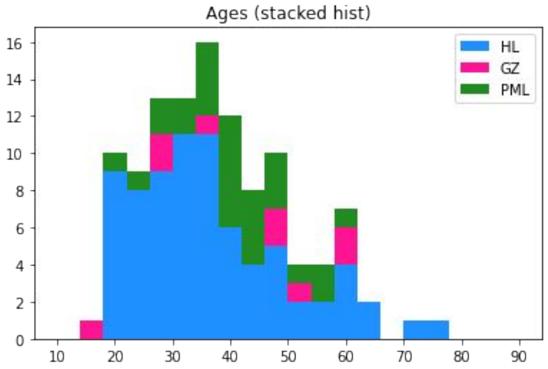
CONVENTIONAL_SUVbwm...
CONVENTIONAL_SUVbwm...
CONVENTIONAL_SUVbws...
CONVENTIONAL_SUVbwm...
CONVENTIONAL_SUVbwQ...
CONVENTIONAL_SUVbwQ...
CONVENTIONAL_SUVbwQ...
CONVENTIONAL_SUVbwS...
CONVENTIONAL_SUVbwK...
CONVENTIONAL_SUVbwE...
CONVENTIONAL_SUVbwpeakSphere0,5mL:discretized volume sought
CONVENTIONAL_SUVbwpeakSphere0,5mL(value only for PET or NM)
CONVENTIONAL_SUVbwpeakSphere1mL:discretized volume sought
CONVENTIONAL_SUVbwpeakSphere1mL(value only for PET or NM)
CONVENTIONAL_TLG(mL)[onlyForPETorNM]

CHECK_Cluster(s)ToSmall
GLCM_Homogeneity[=InverseDifference]
GLCM_Energy[=AngularSecondMoment]
GLCM_Contrast[=Variance]
GLCM_Correlation
GLCM_Entropy_log10
GLCM_Entropy_log2[=JointEntropy]
GLCM_Dissimilarity
GLRLM_SRE
GLRLM_LRE
GLRLM_LGRE
GLRLM_HGRE
GLRLM_SRLGE
GLRLM_SRHGE
GLRLM_LRLGE
GLRLM_LRHGE
GLRLM_GLNU
GLRLM_RLNU
GLRLM_RP

- number
- date
- text
- interval

Dataset transformation

- CSV file → *Pandas DataFrame* (Python object)
- Features with no clear contents **dropped**
  - `PARAMS_IntensityResampling`
  - `PARAMS_BoundsRangeOfValueAfterDiscretisation(SUVbw)`
  - `CHECK_Cluster(s)ToSmall`
- **Data format correction**
  - decimal separator : “,” → “.”
  - date notation : “dd/mm/yyyy” → *datetime* (Python object)
- **New feature added**
  - age, from dates subtraction (then dropped)



99 → 95 FEATURES

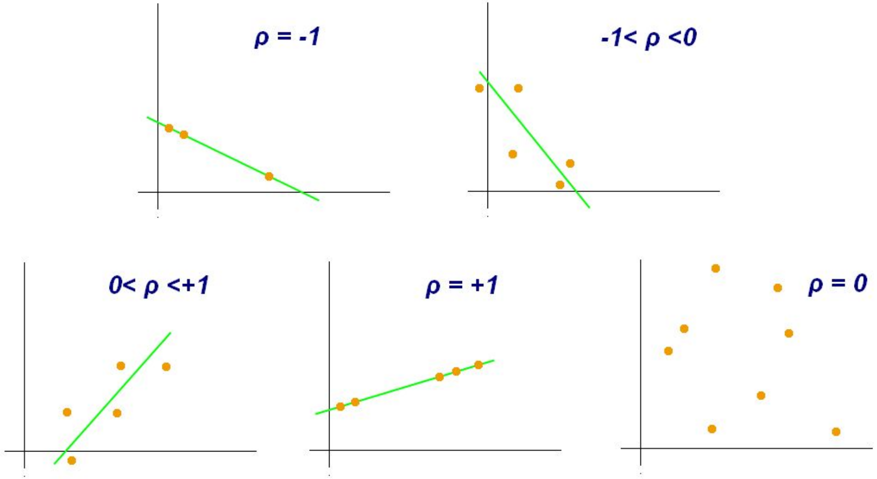
```
data_visualization
```

Feature correlation [1/2]

Two features strongly **correlated** bring the same statistical information then, in order to **increase** the Machine Learning capabilities, one should **clean** the dataset to have independent features.

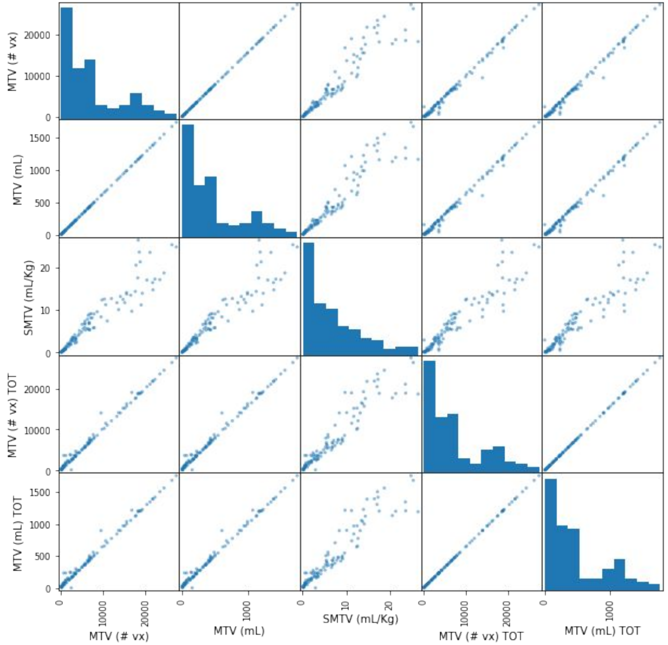
$$\rho_{X,Y} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_j (x_j - \bar{x})^2 \sum_k (y_k - \bar{y})^2}}$$

Pearson correlation coefficient



Feature correlation [2/2]

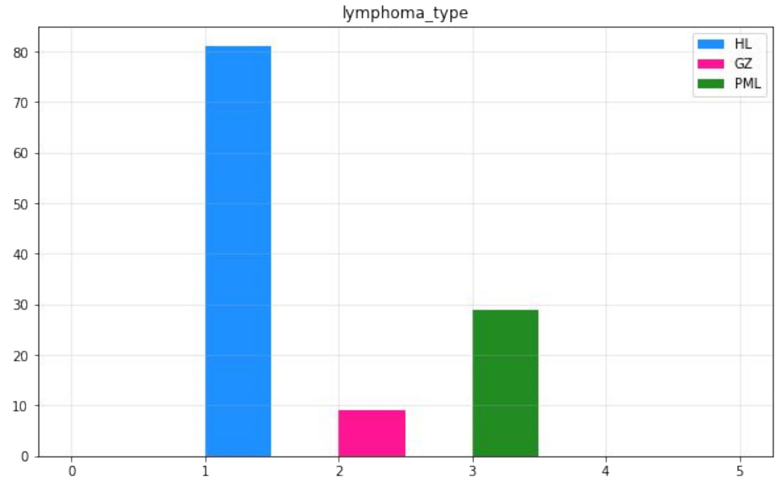
- Pandas DataFrame is able to compute easily **pairwise correlation of columns** (features), excluding NA/null values
- Here a graphical example for a bunch of variables
  - `MTV (# vx)`
  - `MTV (mL)`
  - `SMTV (mL/Kg)`
  - `MTV (# vx) TOT`
  - `MTV (mL) TOT`
- Variables with strong correlations **dropped**
  - $\rho_{X,Y} \geq |0.75| \rightarrow$ *strong correlation*



95 → 17 FEATURES

```
binary_classification
```

Instance space



- Missing information
  - some features not available for all the instances
  - instances **dropped** to keep homogeneity
  - 119 → 101 rows (*instances*)
- Dataset clearly **unbalanced**
  - HL-class is over-represented w.r.t. the other two ones
  - classification suffers from unbalancing
- **Binary classification**
  - HL and non-HL classification
  - dataset a bit more balanced
  - NHL-HL ratio : 55.4%

Train/Test split

DATASET | size: 101 | ratio: 100%

TRAIN-SET | size: 80 | ratio: 80%

TEST-SET | size: 21 | ratio: 20%

To ensure that our Machine Learning model will be able to **generalize** well to new cases, namely to behave like a *true predictor*, we should avoid to use all the dataset to train our model. A better option is to **split the data into two sets**:

- ***training set*** to train the model
- ***test set*** to test the model

The error rate on new cases is called the ***generalization error***, and we can estimate it evaluating the trained model on the test set.

Model optimization: *k-Folds Cross Validation*

In the search of the best possible predictor, we should avoid that our model learns by heart the training instances, because it will result in a **greater generalization error**.

A possible solution is to use ***k-Folds Cross Validation***, an algorithm based on the splitting of the training set into k different subsets. We will use k-1 subsets to train the model and leave the last subset to validate it. Then, the average performance of our model against each of the folds can be used as a robust score to build an **optimization problem**, namely to find the best possible model.

In the Jupyter Notebook we have used k = 3!

TRAIN-SET

TRAIN	TRAIN	VALIDATION	→ score
TRAIN	VALIDATION	TRAIN	→ score
VALIDATION	TRAIN	TRAIN	→ score

mean score ± standard deviation

Performance measures: *accuracy, precision & recall*

The ***accuracy*** is generally not the preferred performance measure for classifiers, especially when one are dealing with **unbalanced datasets**. In fact, classifiers succeed in recognizing instances belonging to most frequent label, resulting in high accuracy even if the other instances are mis-classified.

In order to ensure a model evaluation robust against unbalanced datasets, one can use ***precision*** or ***recall*** scores.

Actual labels \ Predicted labels	Negative	Positive
Negative	**True Negative (TN)**	**False Positive (FP)**
Positive	**False Negative (FN)**	**True Positive (TP)**

In the Jupyter Notebook we have used RECALL!

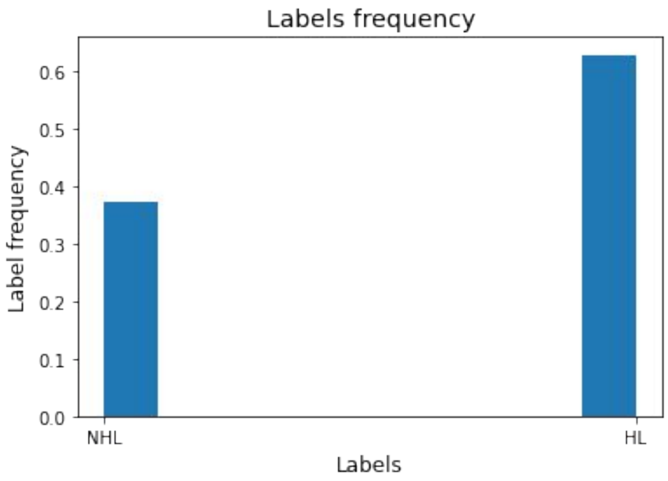
$$\mathrm{accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

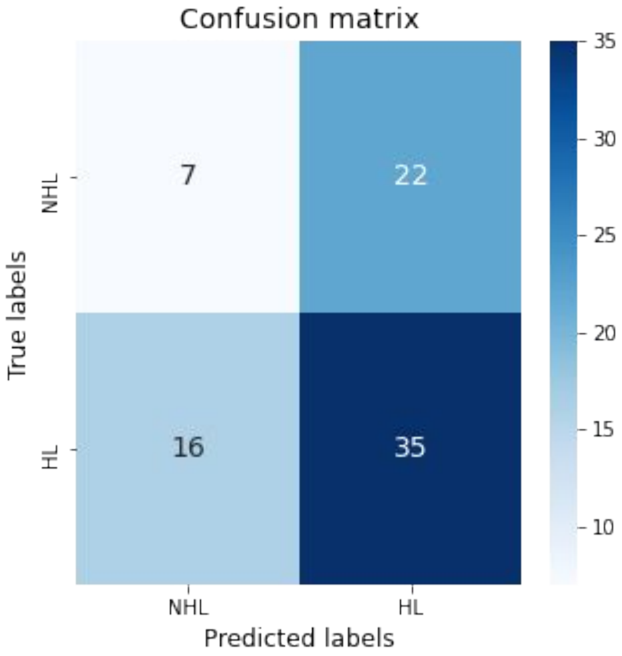
$$\boxed{\mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}}$$

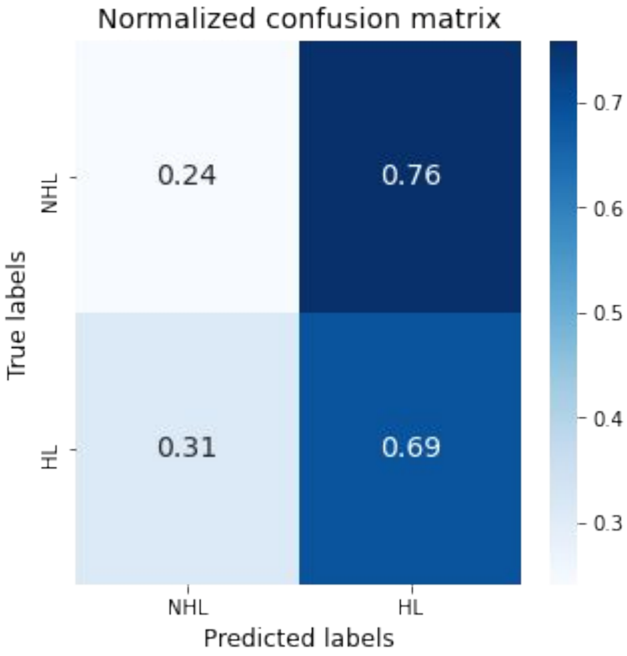
Dummy Classifier: *model baseline*

- The ***Dummy Classifier*** is a simple classifier that makes predictions using simple rules:
  - computes the frequency of each label
  - interprets such frequencies as probabilities
  - predicts new instances label on the basis of computed probabilities
- This classifier is useful as a simple baseline to compare with other (real) classifiers

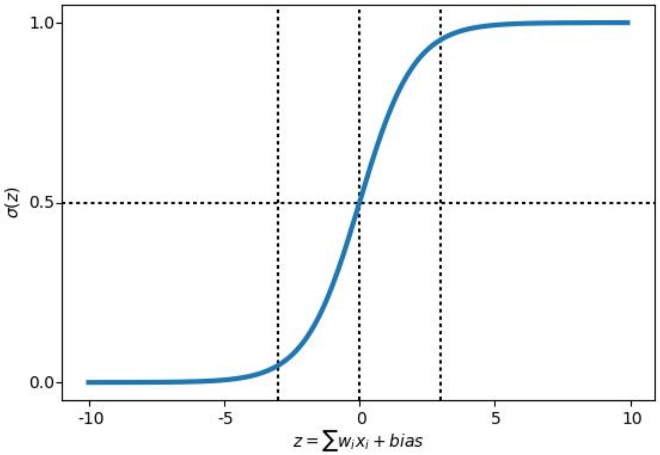


Dummy Classifier: *performance*





Logistic regression



The ***Logistic Regression*** (LR) is a statistical model that in its basic form uses a **logistic function** σ(·) to model a binary dependent variable.

- LR model computes a weighted sum of the input features z (plus a bias term)
- This weighted sum are mapped it to a number between 0 and 1 through the logistic function
- The logistic output σ(**x**) can be interpreted as the **probability** that **x** belongs to the positive class

$$z(\mathbf{x}_i) = w_0 + \sum_{j=1}^{n} w_j \, x_{i,j} = \mathbf{x}_i^T \mathbf{w}$$

weighted sum of features

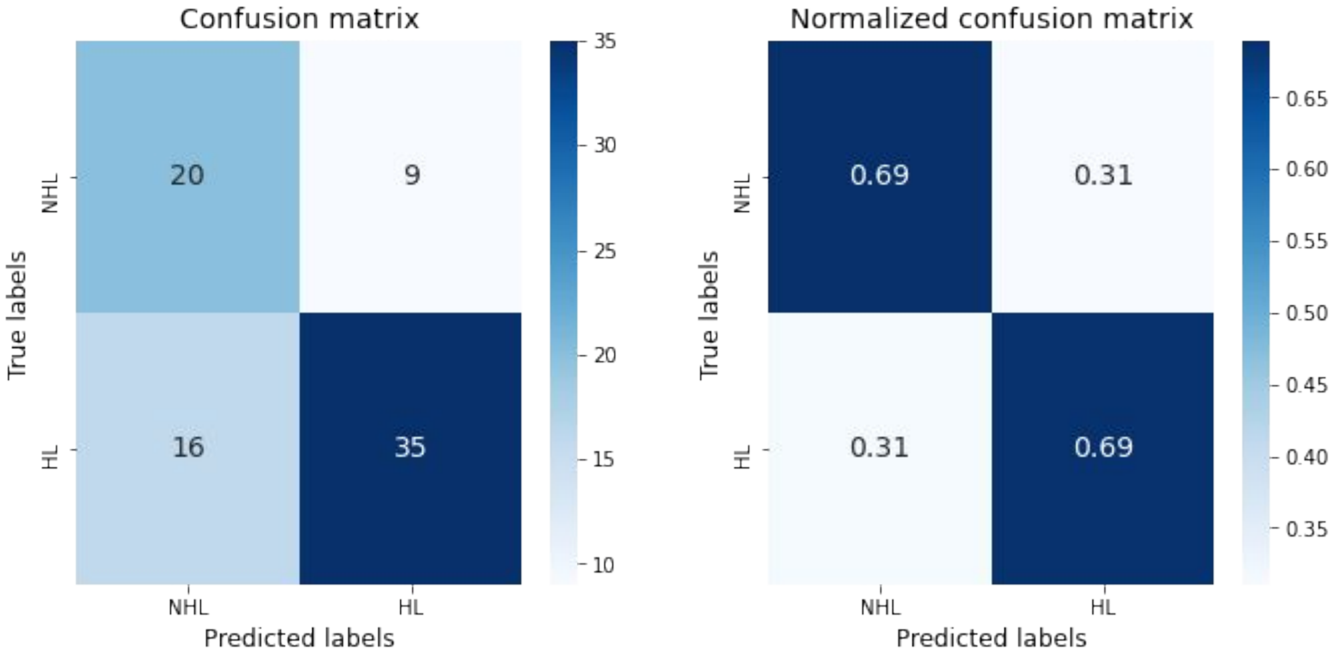
$$\hat{p}_i \doteq \sigma\left(z(\mathbf{x}_i)\right) = \frac{1}{1 + \exp(-z)}$$

< 0.5 → NHL

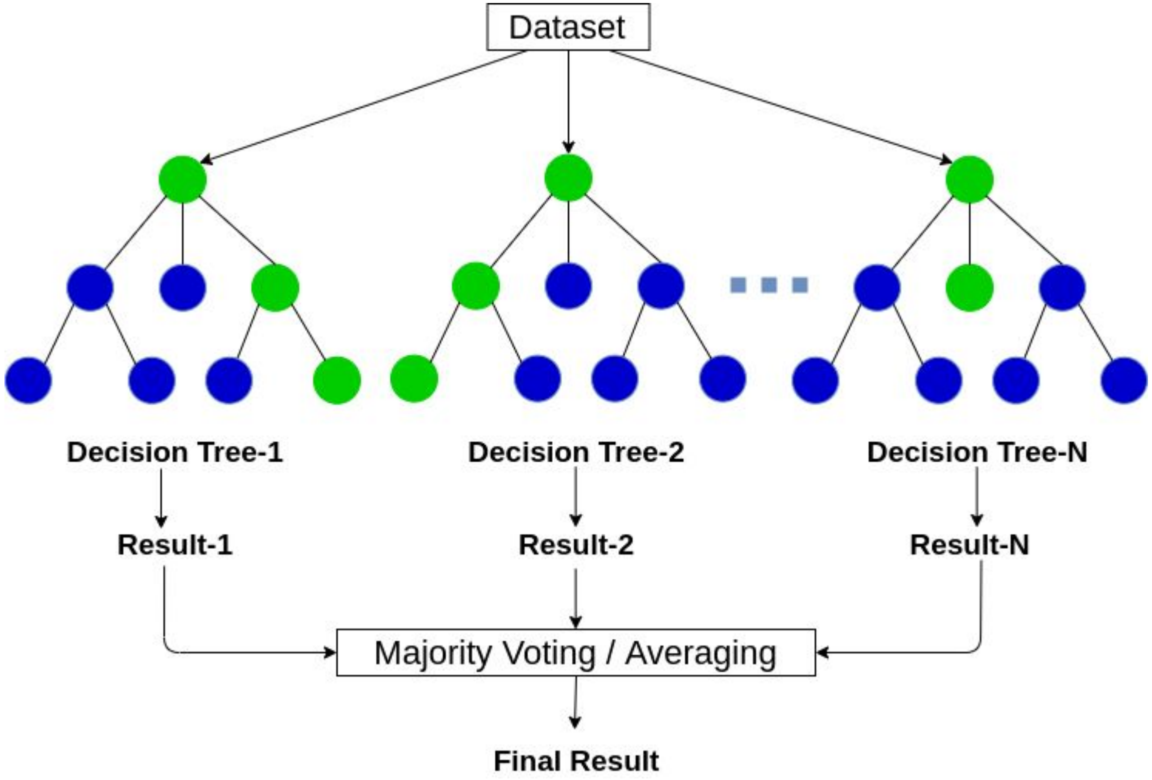
≥ 0.5 → HL

predictions

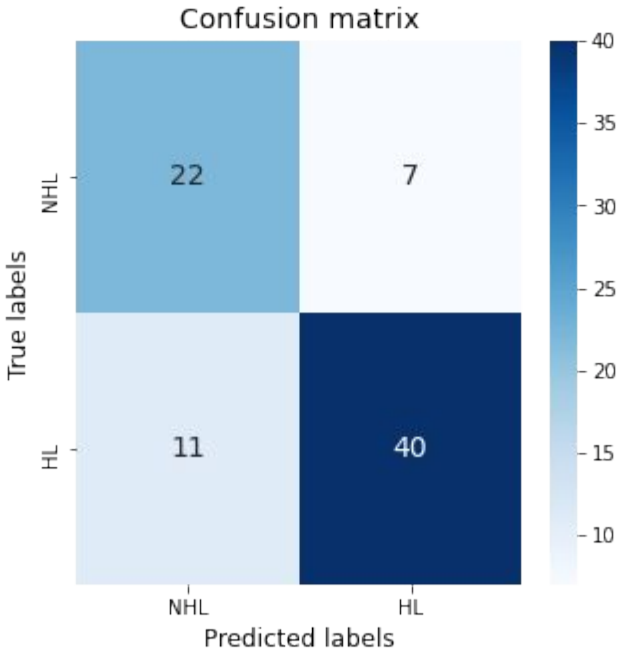
Logistic regression: *performance*

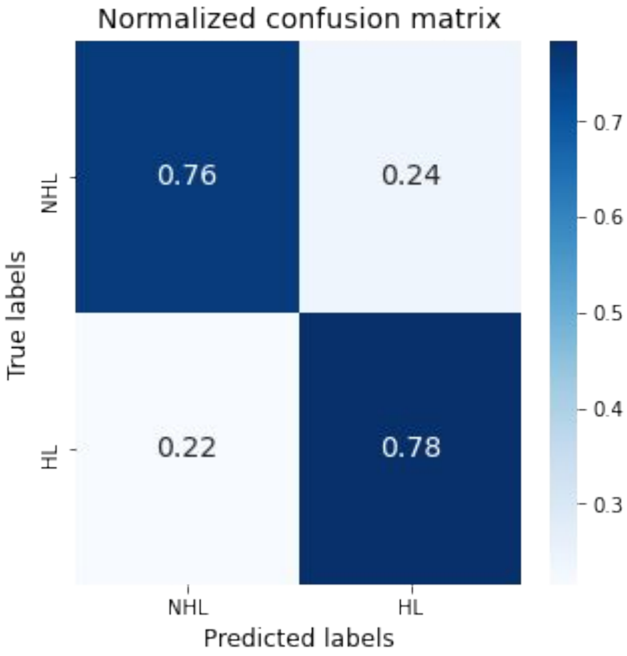


Random Forest classifier

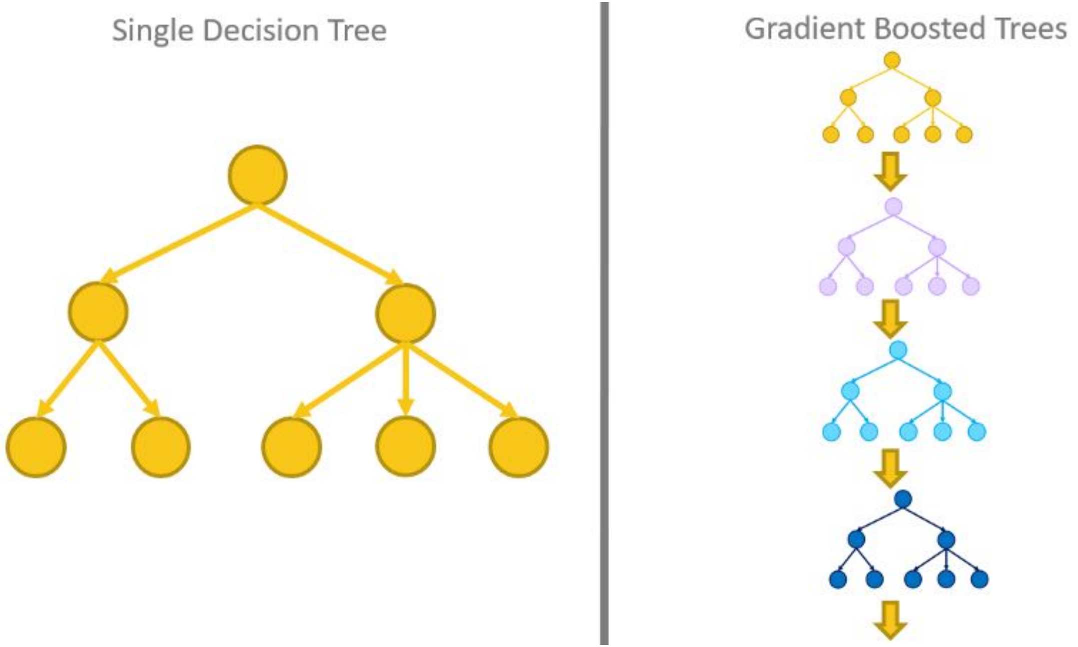


Random Forest classifier: *performance*



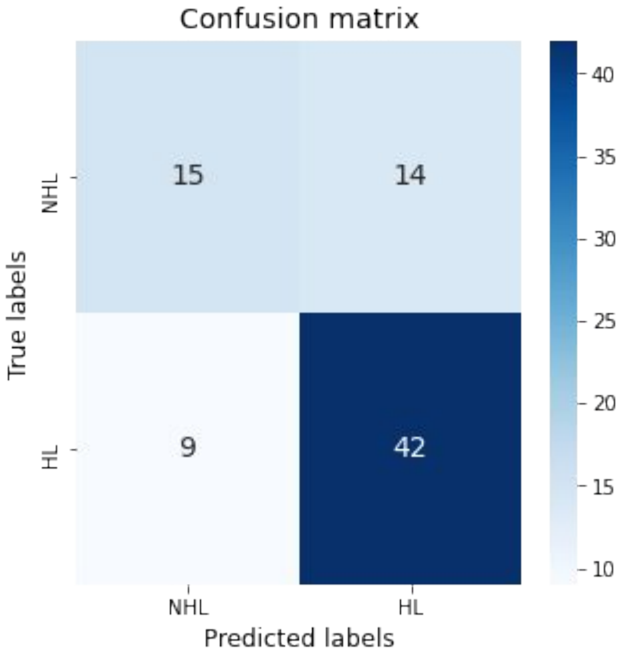


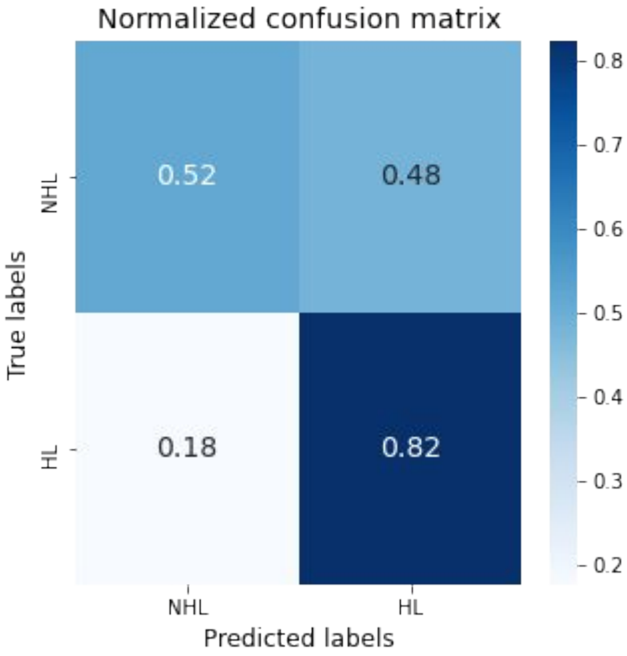
Gradient BDT classifier



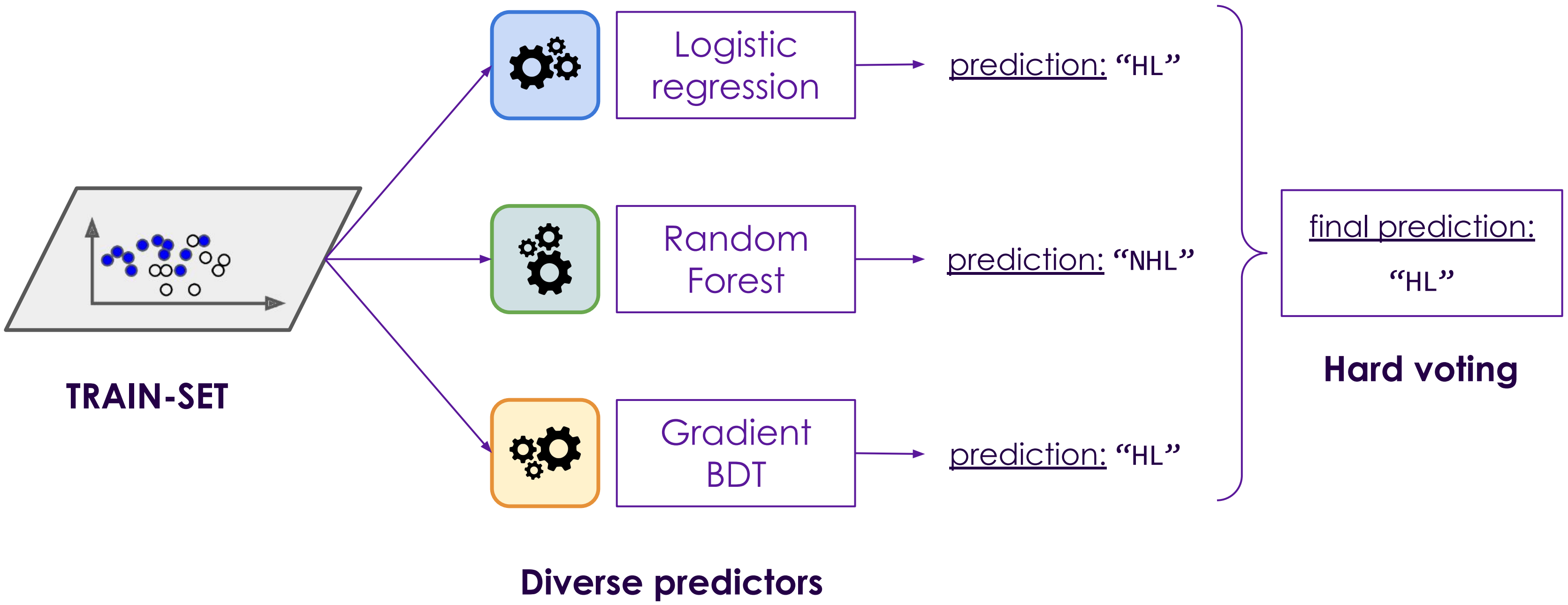
- ***Gradient BDT*** combines multiple simple decision trees into a stronger learner
- Each tree is trained on the **residuals** from the previous sequence of trees
- All trees are then combined together using an additive model

Gradient BDT classifier: *performance*

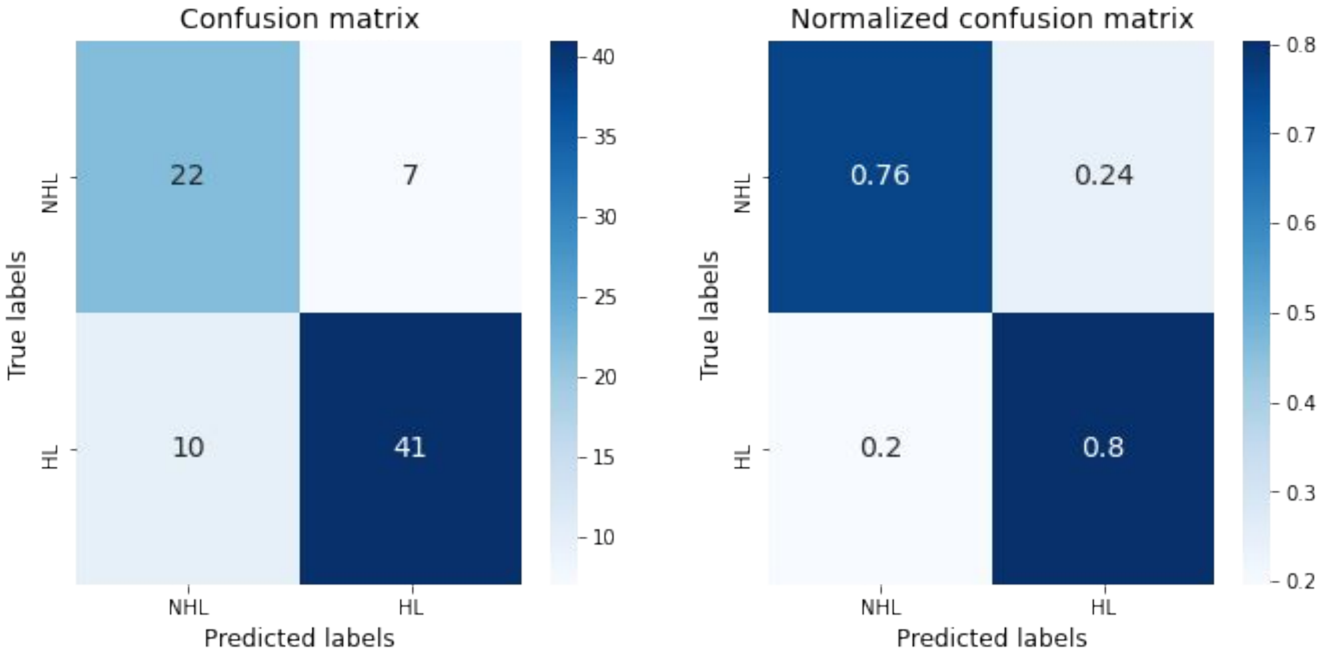




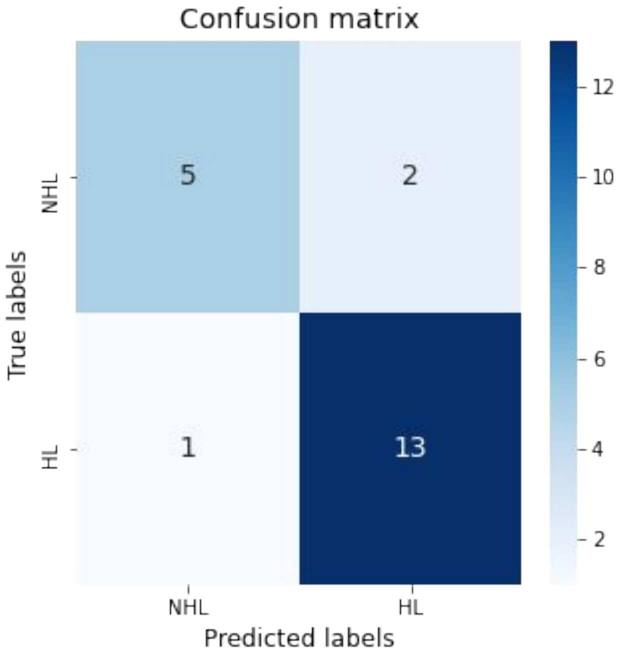
Model combination

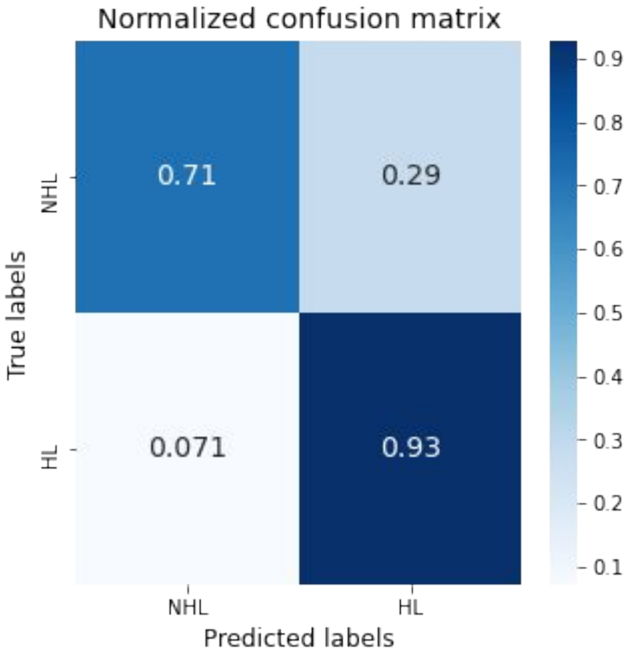


Model combination: *performance*



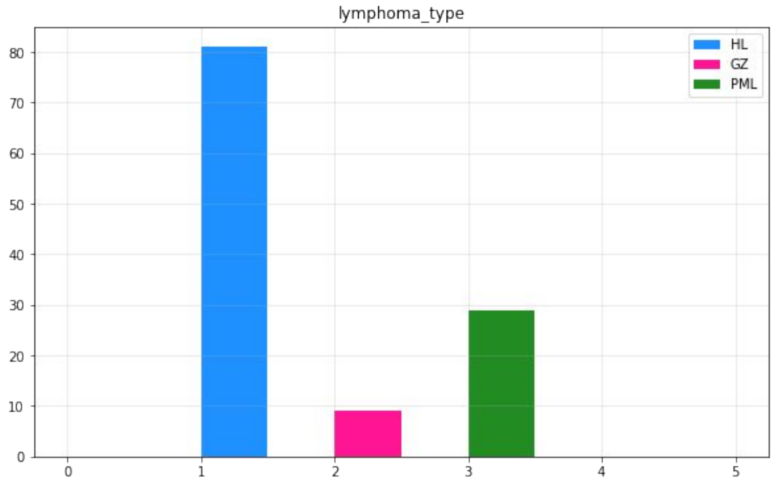
Results on the test-set





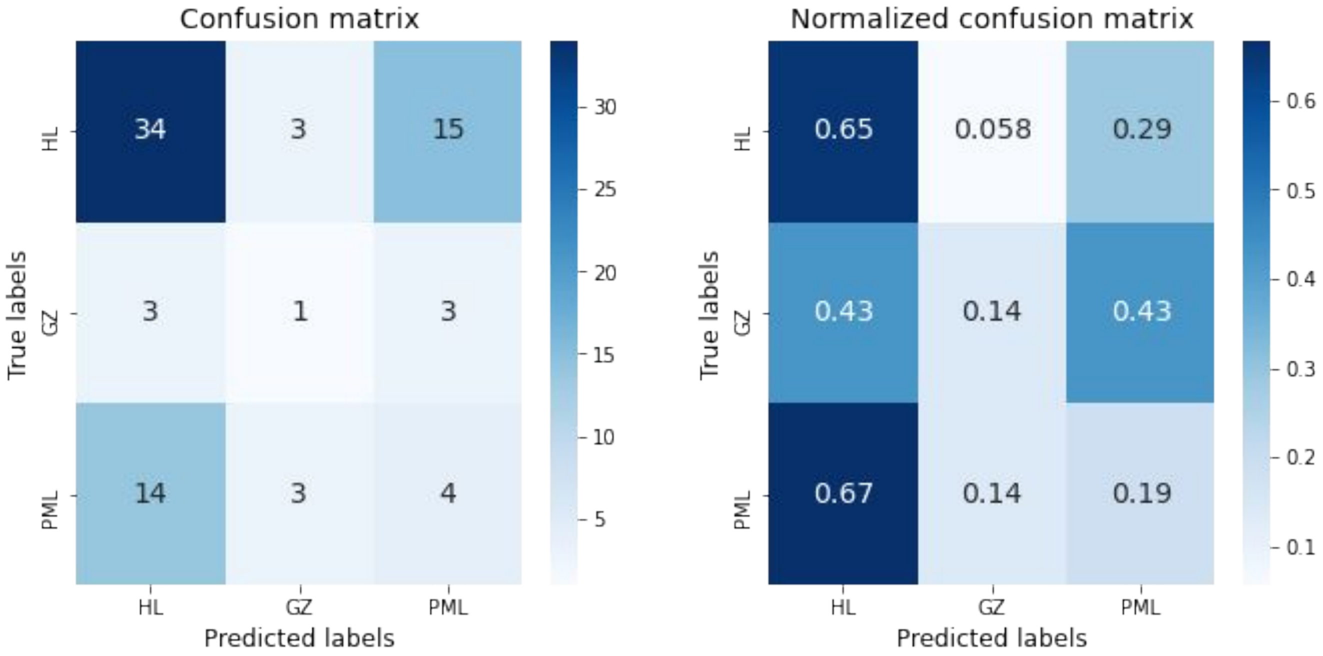
```
multiclass_classification
```

Instance space

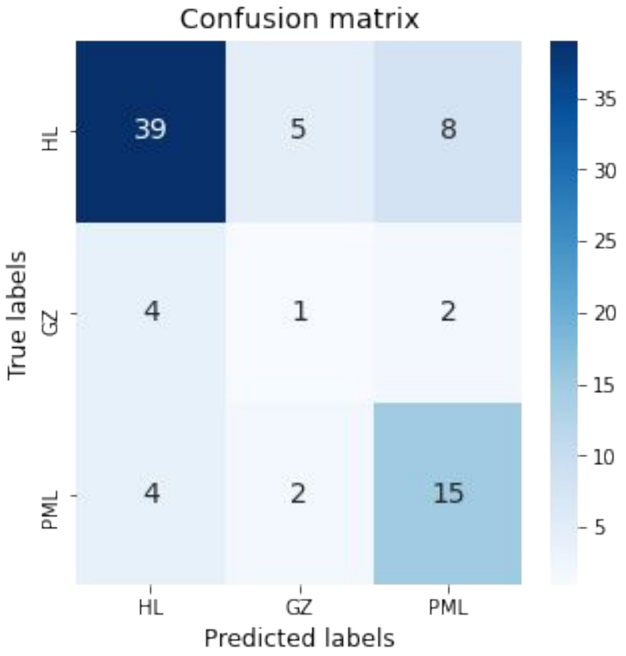


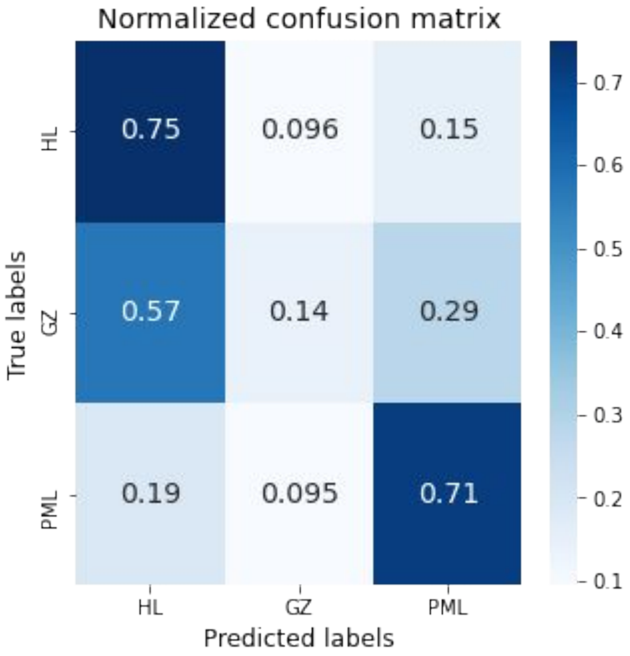
- Missing information
  - some features not available for all the instances
  - instances **dropped** to keep homogeneity
  - 119 → 101 rows (*instances*)
- Dataset strongly **unbalanced**
  - HL-class is over-represented w.r.t. the other two ones
  - classification suffers from unbalancing
- **Multiclass classification**
  - HL, GZ and PML classification
  - GZ-HL ratio : 13.8%
  - PML-HL ratio : 41.5%
- Train/Test split : **80%/20%**
- k-Folds Cross Validation with **k=3**

Dummy Classifier

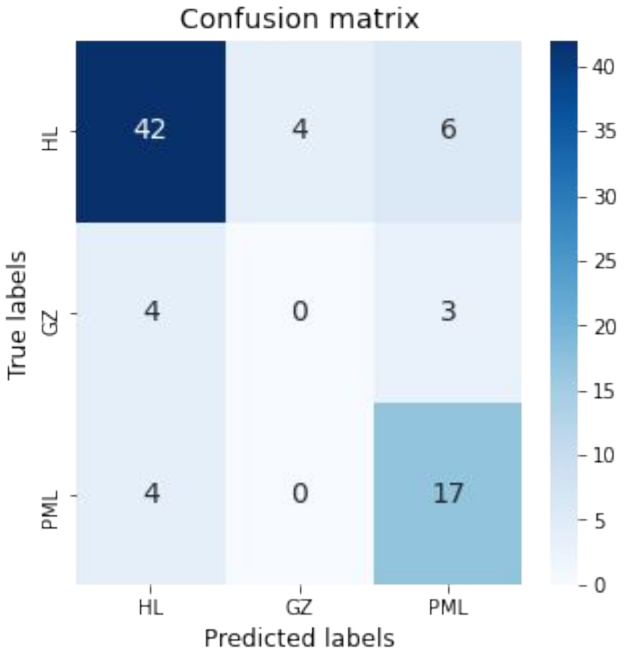


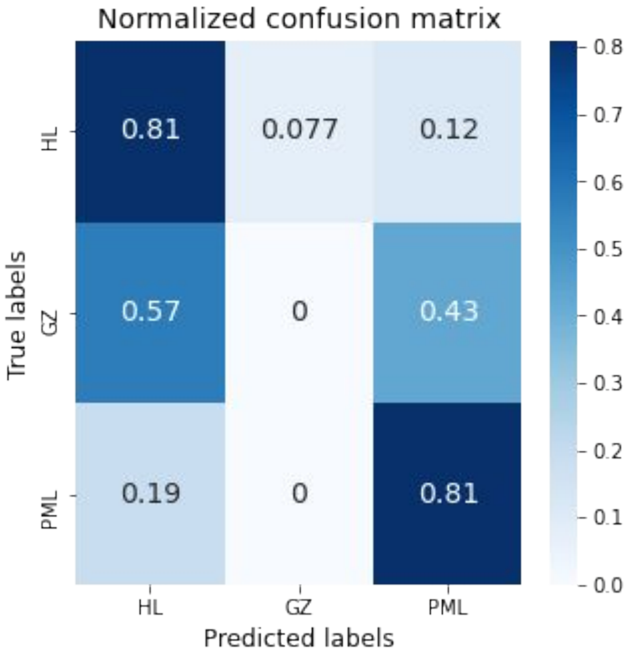
Logistic regression: *performance*



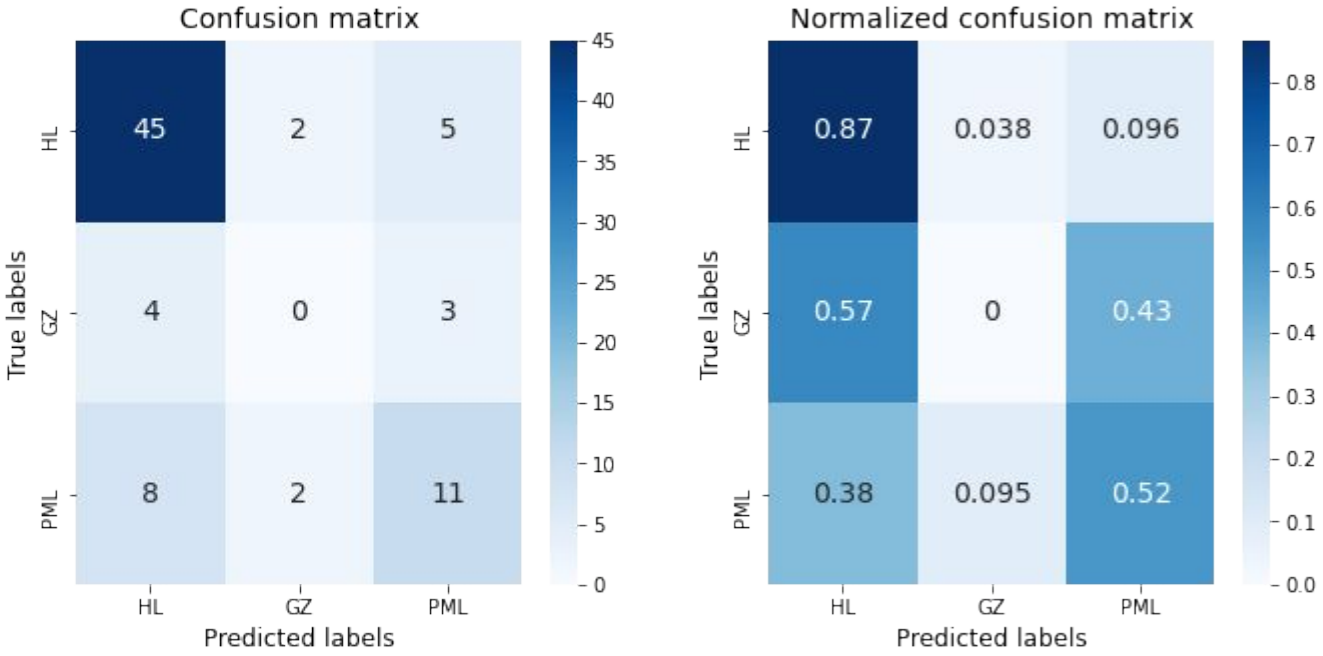


Random Forest classifier: *performance*

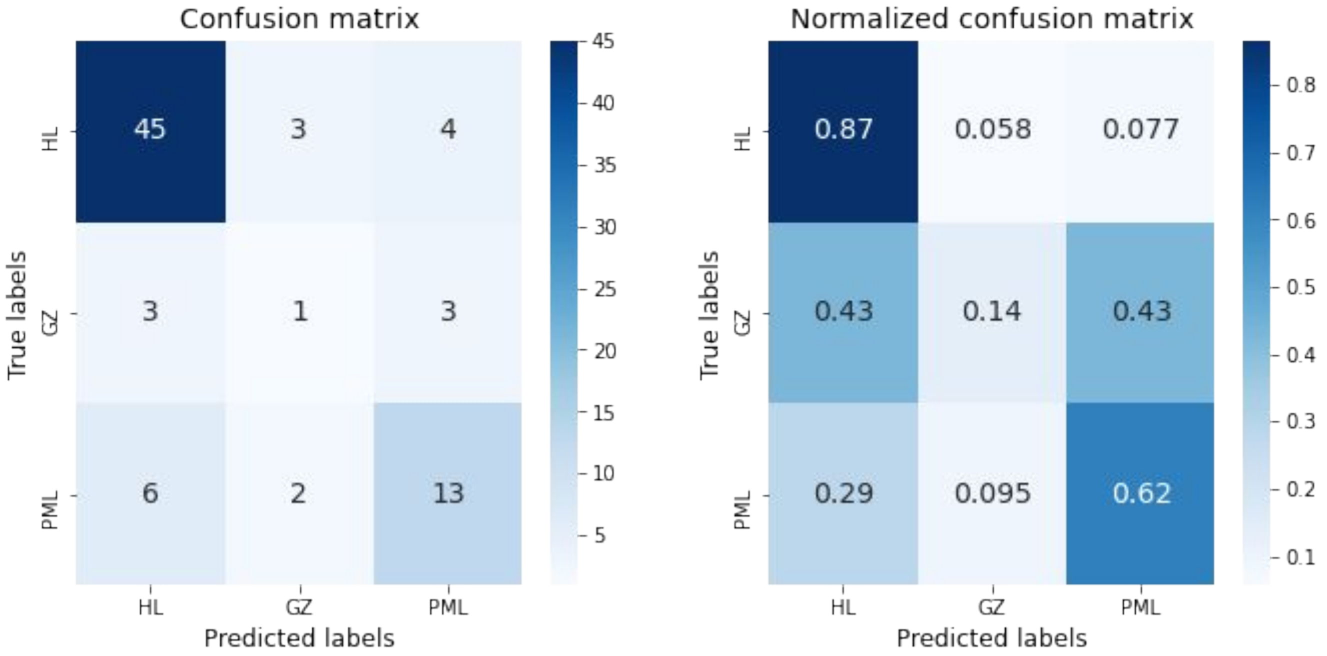




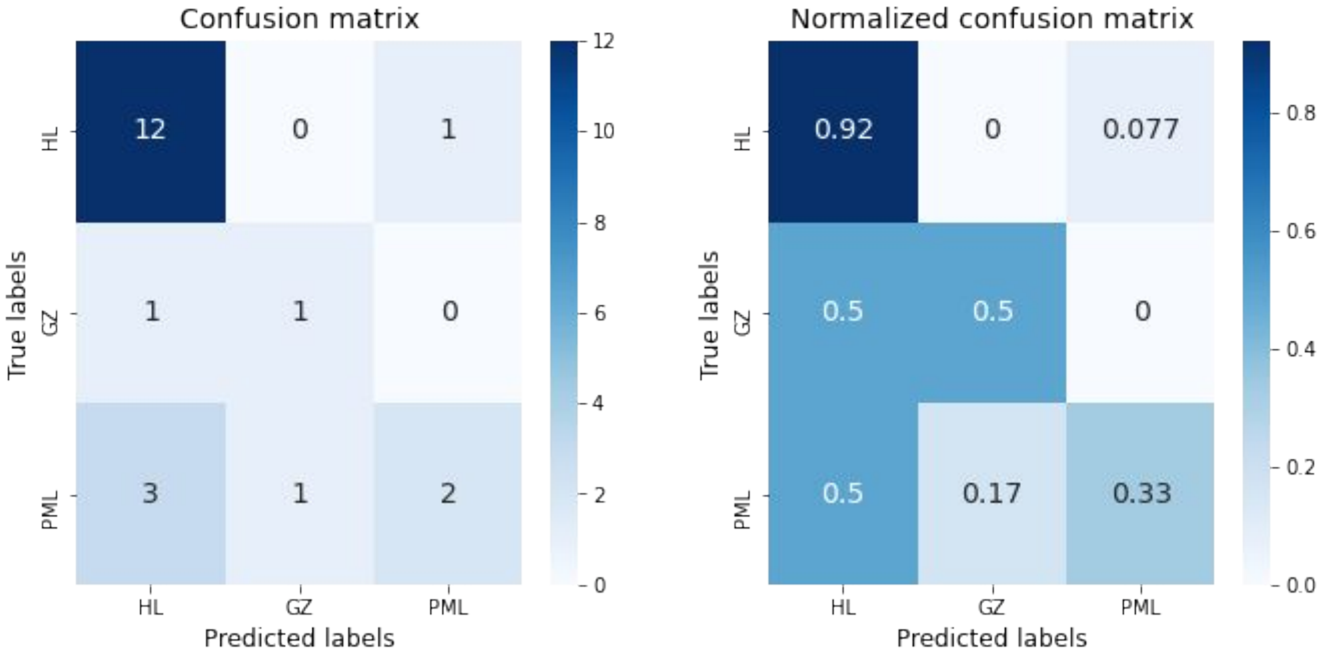
Gradient BDT classifier: *performance*



Model combination: *performance*



Results on the test-set



Conclusion

- Preliminary studies
- There is room for improvements
  - **increasing** the dataset → more data
  - **balancing** the dataset → expanding *low-represented* classes
  - working with **low-level data** → *image classification* techniques
- Still need for taking a look at literature

Open to any kind of suggestions!

Backup

Undertraining & overtraining