ML outcomes (classification, feature selection) depend on the seed that have been used to perform train/test split, to select the feature, to run the chosen model.

If the results are unstable because of this amount of randomness, then the model is bad and should not be used/generalized.

For example, with different seeds different features may be selected and the model could provide totally different metrics

Literature review

1. *Evaluation of feature selection methods (MDA, SHAP, LIME) and index of “instability”* <https://arxiv.org/abs/2005.12483>
2. *Evaluation of the impact of T/T split on cardiovascular imaging real datasets.* They explored several methods to train the models (single t/t split, k-fold CV, repeated k-fold cv, bootstrap) and several models <https://www.nature.com/articles/s41598-021-93651-5>
3. *Evaluation of dataset dimension and size of T/T split on several models in quantitative structure-activity/property (QSAR/QSPR) relationships and classification* <https://mdpi-res.com/d_attachment/molecules/molecules-26-01111/article_deploy/molecules-26-01111-v2.pdf?version=1614303925>
4. See paper from Xu and Goodacre for a use of MixSim applied to generalization of performance in split train/test
5. *Stable learning via sample reweighting:* performance of ML diminish when training and test set do not have the same distribution. Predictive model that works well with any x point, with sample reweighting.
6. *On Model Stability as a Function of Random Seed:* for neural networks

**IDEA**

1. Generate synthetic data:

* Different overall sizes
* Different train/test sizes
* Increasing number of classes to identify
* Different class balance

**MixSim** model to generate datasets with different probabilities of misclassification and sample size. With this method you can define: n. classes, n. variables, size of each class and overlap between the classes. The result is a multimodal distribution.

After parameters are defined, you can generate the desired n. of samples

For this step we can also use dataset obtained from web, such as from Kaggle (example: Cancer Patients Data, Lung Cancer, Breast Cancer Dataset, Real Breast Cancer Data etc etc). In this way we could avoid the step of dataset creation.

Problem: check for datasets with more classes, usually they are binary.

1. Comparison:

* Different ML algorithms
* Different algorithm to select random seeds: Montecarlo simulation?
* Different feature selection

1. Definition of a measure of stability of the model and of dependence from the seed
2. Try on real dataset (our own) to verify our index

**GENERALIZATION ERROR (OUT-OF-SAMPLE ERROR):** measure of how accurately an algorithm is able to predict outcome values for previously unseen data.

*Estimates* of the generalization error through the learning process, which are called [learning curves](https://en.wikipedia.org/wiki/Learning_curve).

Step 1: lets generate 3 datasets, of 50, 500 and 5000 patients.

For each case create 2 and 3 clusters

We could try to build a continuum: from 50 to 5000 patients, adding 10 patients by time.

Using same n of classes and same overlapping.

Where is that the seed has impact, f.e. on the separation of T/T

**Random seed generator**

**R:** set.seed(kind)

If seed not specified R uses clock of the system to take one.

Default is Mersenne-Twister algorithm

To set the kind: random number generator (RNG) state can use different functions also on set.seed()

setRNG

“dqrng” package

“random” package

**Python:** random.seed() with Mersenne-Twister

Analysis for the LASSO case (classification in 2 classes)

We are interested in checking the Lambda selected, the performance (AUC), the n. of features obtained, the exact features obtained when the seed is changed.

In this case we are going to use CrossValidation, after separating the initial dataset in train and test set, which is the method most used to optimize parameters trying to avoid the randomness effect.

The seeds used are 1…1000, in order to avoid randomness at that point.

The seed affects the creation of the folds, while it does not have an effect on the starting point for choosing the parameter.

We set a fixed number of features. equal to 20.

We need to center and scale the data to apply the LASSO regression. Normalization is performed separately for train and test set, before running the CV

When the parameter lambda increases the number of selected feature decreases, additionally the features that were set a 0 with a small lambda WILL be 0 with larger lambda.

CHECK THE DIFFERENCE BETWEEN LARGER AND SMALLER N.FEATURE SELECTED

We want to check several factors:

1. What happens when we vary the n. of samples? (100 to 500)
2. What happens when we vary the n. of folds for CV? (3, 5, 7, 10)

PER QUESTO PUNTO SELEZIONEREI SOLO UN N.SAMPLES (ex: 300)

1. What happens when we vary the algorithm that generates the sequence from the seed in R?

PER QUESTO PUNTO SELEZIONEREI SOLO UN N.SAMPLES (ex: 300)

1. What happens if we perform the same procedure in R and Python, using the same data, the same seeds and the same generator algorithm?

Check the distribution of lambda, the best model (auc test), the CV of lambda and n.features

Let’s stay on binary classification.

Try to write a formula to consider the relation between sd of the results and the characteristics of the data and the model: sample size, “complexity” (information from the features), randomization algorithm and classification algorithm.

Are we able to write a function with those data?

Model with:

* Number of samples (60…1000)
* Number of features
* Problem complexity: 2,3 classes
* Data complexity (average absolute correlation or similar, like Mutual dependence)
* Model: Lasso, SVM with linear kernel, RF, NN?

As a first analysis with a linear regression

lm(formula = sd ~ N.samples + MeanCorr + Model, data = summary\_res)

we obtained:

Residuals:

(Provide a quick view of the distribution of the residuals, which by definition have a mean zero. Therefore, the median should not be far from zero, and the minimum and maximum should be roughly equal in absolute value.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Min | 1Q | Median | 3Q | Max |
| -0.0172847 | -0.0010571 | -0.0000884 | 0.0008309 | 0.0163532 |

Coefficients

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t value | Pr(>|t|) |
| (Intercept) | -1.179600e-01 | 5.529e-03 | -21.333 | < 2e-16 \*\*\* |
| N.samples | -1.025058e-05 | 1.432e-06 | -7.159 | 4.29e-11 \*\*\* |
| MeanCorr | 1.165969e+00 | 3.650e-02 | 31.940 | < 2e-16 \*\*\* |
| ModelRF | -2.542481e-03 | 6.059e-04 | -4.196 | 4.82e-05 \*\*\* |
| ModelSVM | -2.608314e-03 | 6.059e-04 | -4.305 | 3.14e-05 \*\*\* |

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.002968 on 139 degrees of freedom

Multiple R-squared: 0.9638, Adjusted R-squared: 0.9627

F-statistic: 924.9 on 4 and 139 DF, p-value: < 2.2e-16

Also, the percentage error (RSE/mean(sd))is equal to 8.4%

**Note that MeanCorr is positively associated with the standard deviation!! The higher the correlation the higher the sd. It also makes sense that the number of patients is negatively associated with standard deviation, since augmenting the number of patients diminishes the sd.**

**So, our final model is:**

**sd=-0.1180-1.025e-05\*N.samples+1.166\*MeanCorr+(0\*Lasso-2.542e-03\*RF-2.608e-03\*SVM)**

Lasso is indeed the basis of the categorical variable “Model”, so there is no need to sum anything when it’s the chosen model.

Thanks to an ANOVA analysis we found that the categorical variable “Model” is significant for our final model:

Analysis of Variance Table

Model 1: sd ~ N.samples + MeanCorr

Model 2: sd ~ N.samples + MeanCorr + factor(Model)

Res.Df RSS\_Df Sum of Sq F Pr(>F)

1 141 0.0014371

2 139 0.0012248 2 0.00021235 12.05 1.493e-05 \*\*\*

The radiomics dataset has 600 samples and a high number of features (168), that are highly correlated: we can try to run a step of diminishing the number of features, but before let’s see if our model works properly.

Actually, only one stdev for radiomics falls in the 95% prediction interval of our model.

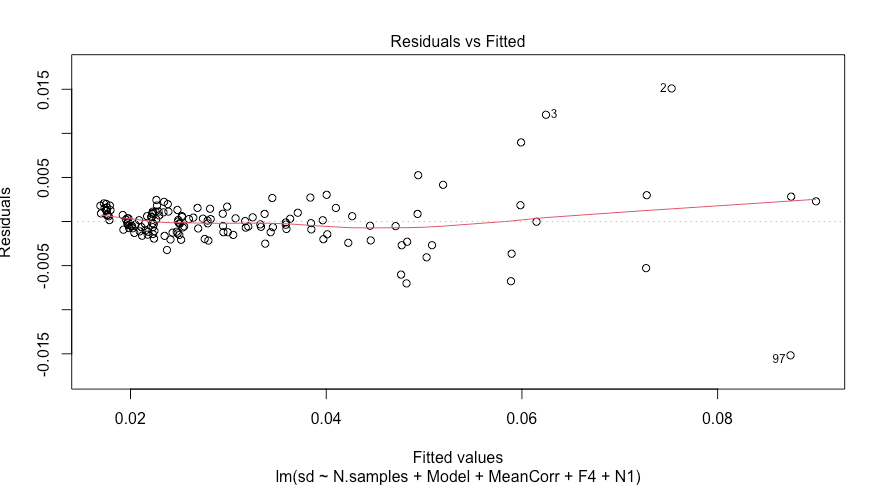
**Addition of complexity measures**

The model above was expanded with the addition of measures of complexity:

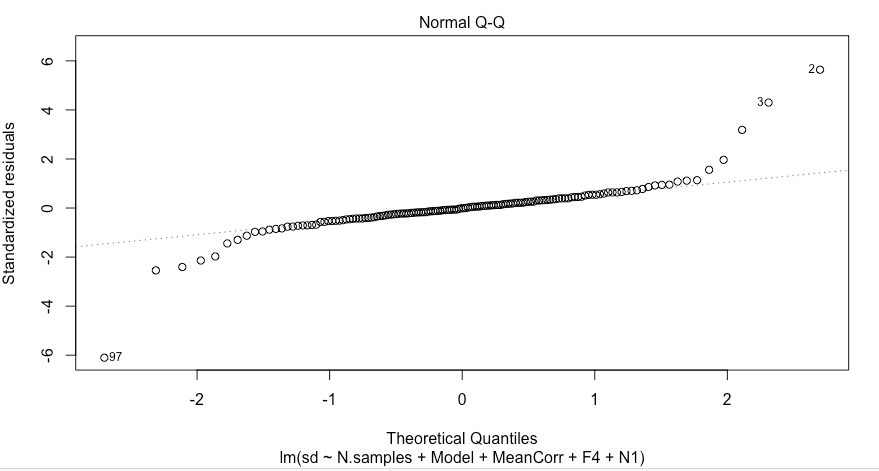
****

However, checking the diagnostic plot we saw that:

* Residuals plot: we still have increasing spread of residuals for alrge fitted values: we need to tranform the response variable to compress large values (like logarithms,1/y)



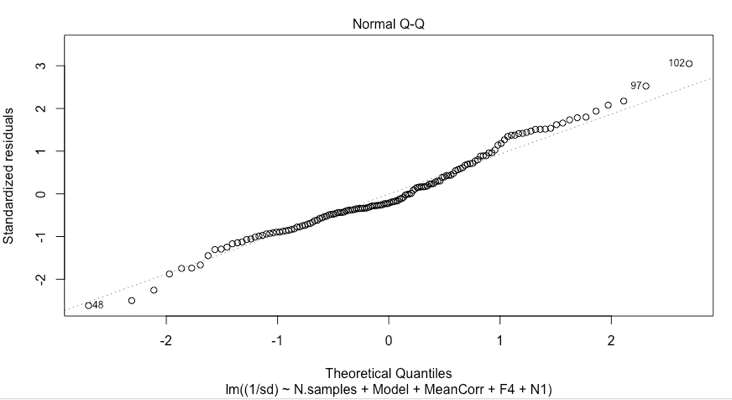
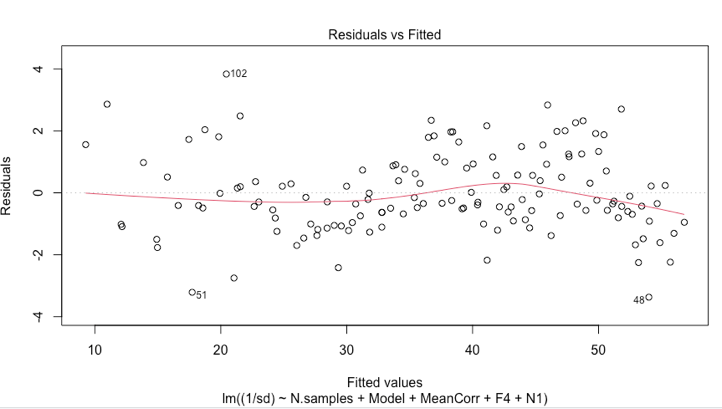
* Q-Q plot: we are in presence of too heavy tails



**Model of 1/sd**

The easiest solution to solve the first problem is to model 1/sd, to compress the large values.

Thus, we used the model:



With this transformation we see a more random distribution of residuals and also an improved Q-Q plot!

Residuals:

Min 1Q Median 3Q Max

-3.3696 -0.8275 -0.2820 0.8168 3.8359

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3.003e+01 7.584e+00 3.960 0.000120 \*\*\*

N.samples 3.498e-02 1.018e-03 34.370 < 2e-16 \*\*\*

ModelRF 2.815e+00 2.700e-01 10.423 < 2e-16 \*\*\*

ModelSVM 1.704e+00 2.700e-01 6.309 3.62e-09 \*\*\*

MeanCorr -1.472e+02 3.180e+01 -4.631 8.38e-06 \*\*\*

F4 6.218e+00 1.812e+00 3.431 0.000796 \*\*\*

N1 1.252e+01 9.522e+00 1.315 0.190803

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.323 on 137 degrees of freedom

Multiple R-squared: 0.9887, Adjusted R-squared: 0.9882

F-statistic: 1990 on 6 and 137 DF, p-value: < 2.2e-16

Also partial residuals plot are definitely good.

To check the goodness of the results on the validation data, we used the 95% prediction interval, which measures the uncertainty around a single value. 95% p.i. [low,up] means that 95% of samples have the y in that range.

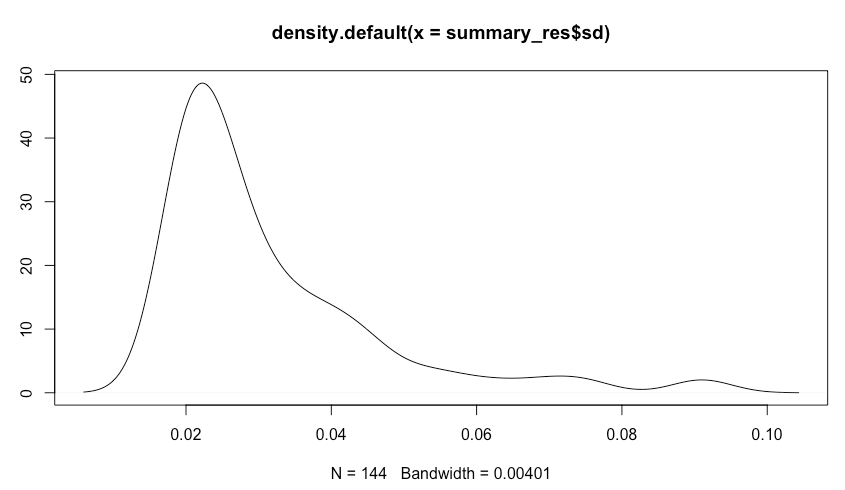
Since we modelled the inverse of standard deviation, to find the 95% p.i. interval of sd we have to set the interval as: [1/upper, 1/lower] and to avoid the values lower than 0, I assigned to negative values the minimum positive lower. WRONG!!!

This modelling this step was ok, but actually it predicts negative values also on external test sets. Thus, we have to solve it putting some constraints on the possible results: sd MUST always be non-negative and we have to model that.

**GLM**

We saw from the plot above that as long as the standard deviation on test auc increases, also its variability increases: the responses (std dev on auc) is likely not to come from normal distribution.

Indeed, the overall distribution of sd is clearly left skewed, with .



We can try to use a GLM to model our response. GLMs are composed by a random and systematic components that have to be modeled. The systematic component assumes the form where is the link function and .

Since our response, st dev is a non negative value we coukd describe the random component as:

We have now to decide which link function to use, in order then to estimate parameters for both components.

With the first easy GLM [sd ~ N.samples+Model+MeanCorr+F4+N1, Gamma(link = "log")] we found that the model is still not good (diagnostic plots)

--------

We need to solve the problem of correlated samples. The idea could be to generate 3 different models, one for each algorithms, and see what happens.

**Datasets for validation:**

1. Simulated radiomic features: 100, 300, 600 samples; 168 features;
2. Radiomics Lung (Our internal data, 270 samples, 172 variables mostly continuous, response: pN)
3. External validation datasets from Kaggle:
4. Cancer (“Lung cancer prediction”, 697 over 1000 samples, 23 variables mostly integer and age, response: risk for developing lung cancer, we consider High and Medium)
5. Breast cancer (“Breast Cancer Dataset (Binary Classification Prediction for type of Breast Cancer)”, 569 samples, 31 continuous var, response: malignant or benign)
6. Diabetes (“Early Classification of Diabetes”, <https://doi.org/10.1007/978-981-13-8798-2_12>, 520 samples, 16 binary variables and age, response: diabetes or not)

We see that models generated overfits, likely because training data are too similar and because we have few data for training now (only 48 datasets for each classifier). It can also be that the problem can’t be generalized.

First we could try to have more data, using GANs to produce different “datasets” coming from different real data, like those from Kaggle. In this way we could generate with different sample size, different number of features, different types of features.

How to better generate data

<https://stirlingcodingclub.github.io/simulating_data/index.html>

<https://search.r-project.org/CRAN/refmans/fungible/html/monte.html>

<https://www.r-bloggers.com/2011/10/simulating-data-following-a-given-covariance-structure/>

<https://community.rstudio.com/t/how-to-generate-random-covariance-matrix/101501>

<https://stats.stackexchange.com/questions/215497/how-to-create-an-arbitrary-covariance-matrix>

<https://search.r-project.org/CRAN/refmans/LaplacesDemon/html/dist.Wishart.html>

<https://cran.r-project.org/web/packages/clusterGeneration/clusterGeneration.pdf>

<https://rdrr.io/cran/clusterGeneration/man/genRandomClust.html>

If I use clusterGeneration, I can generate datasets with different number of rows, diferent number of variables with different levels of noise and of separation.

For example, I can generate 600 datasets with different combination of:

* Separation: 0.01 (Low), 0.21 (medium), 0.324 (high)
* Number of non noisy variables: from 10 to 50
* Number of noisy variables: 5 or 10
* Cluster sizes are in a range [30,200].

I generated 10 replicates for each case. So I have a lot to play with now!