|  |  |
| --- | --- |
| Amadeusz Masny1  1AGH Akademia Górniczo-Hutnicza  al. Mickiewicza 30, 30-059 Kraków | Kraków, 30 May 2019 |
|  | |
| MACHINE LEARNING HYPERPARAMETERS OPTIMIZATION WITH HYPERBANDSTER FRAMEWORK, BASED ON A EUROPEAN COURT OF HUMAN RIGHTS OPEN DATA | |
|  | |

Abstract**:** This paper presents one of the available methods for machine learning hyperparameters optimization. The main point of reference for this survey is related work done based on a data from European Court of Human Rights project. The main discussed problem touches an improvement of predictability process from previous related work.

Keywords: ECHR (European Court of Human Rights), ML (Machine Learning), HPBANDSTER (Hyperband optimization on Steroids)

1. Introduction

The simple and traditional approaches for building any predictive ML model are mostly focused on a simple model preparation and evaluation. One of the last steps in such a process is often not taking into consideration (fine tuning of such a model) or just treated manually. Even there are a lot of examples how to tune hyperparameters of particular ML model, still these examples are mostly focused on a manual search (trial and error).

The amount of globally generated data is constantly growing. An increasing number of companies and people started to use ML methods for different tasks such a predictive modelling. These aspects and many more bring a challenging situation where Data Scientist is facing a problem with a time. The most known time consuming process for ML is a learning phase, but this is not the only one that requires huge persistence. To build a better model, most likely better fitted to the specific data and problem, Data Scientist should tune particular algorithm.

Tuning process depends on a specific ML algorithm. Classical methods used for hyperparameter tuning include manual search and GridSearch over hyperparameter space. Both cases are really not time efficient. Researchers came with different ideas to solve that problem. Namely, they have harnessed well known optimization methods that work over multidimensional minimization problems. Bayesian Optimization and Hyperband combined into a one algorithm could lead hyperparameter tuning into the new horizons of a time efficiency.

The research described here is related to ECHR-OD project that is the database for diverse problems, based on the European Court of Human Rights documents available on HUDOC [1].

1. PREVIOUS RESEARCH

This whole work is based on a previous research done as a ECHR-OD project that aims to create one good quality, open database for ECHR documents available for Data Scientists. This paper tends to reveal the potential of ML hyperparameters tuning in case of a comparison between the old achieved results and results after hyperparameters tuning. All of the information about previous research including each preparation and evaluation step can be found on a research Github webpage [2].

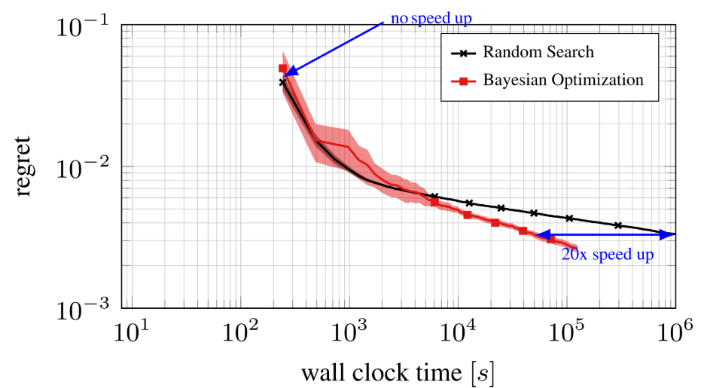
1. HYPERBANDSTER

The writer approach to perform an automated hyperparameter tuning was usage of AutoML [3] solution, more precisely the Hyperbandster [4] framework. It consists of a Bayesian Optimization and Hyperband techniques. Validation performance each of the ML model can be formulated as a function of their hyperparameters. This kind of situation can be then further passed as an optimization problem (maximization or minimization). Both BO and HB tackle such a problem.

**3.1 Bayesian Optimization**

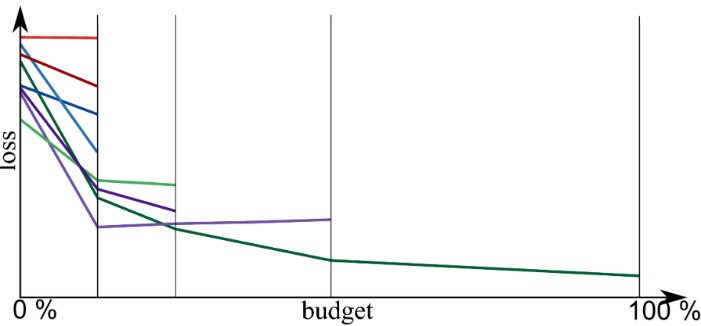
BO uses a probabilistic model to obtain a hyperparameter configuration. Based on a previous ML model validation results, it can further investigate promising configuration hyperparameters space. The standard BO process iterates over these three steps:

1. Select the point that maximizes the acquisition function
2. Evaluate the objective function at this point
3. Add the new observation to the data and refit the model

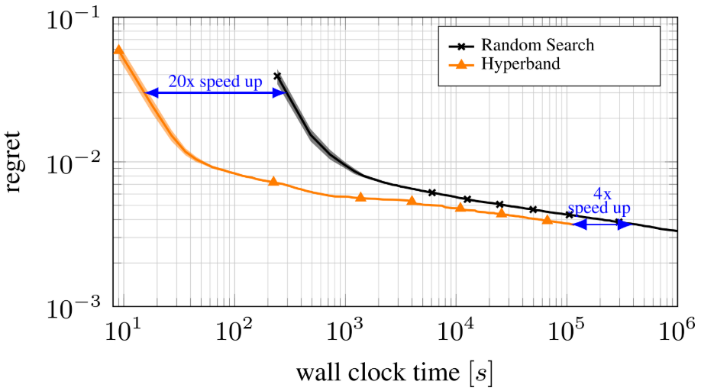
As Bayesian Optimization is not very efficient in the beginning of entire optimization process it shows huge improvement over time.

*Fig. 1. BO and Random Search Comparison for Different Time Budgets [6]*

**3.2 Hyperband**

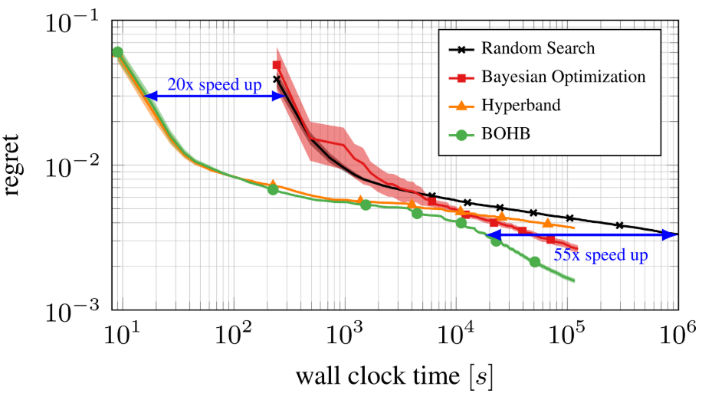
HB is one of the HPO (Hyper Parameter Optimization) strategy that uses a SH (Successive Halving) on the evaluated models. In the other words, it uses cheap to evaluate approximations of the objective function on smaller budgets and keeps only the best half of the evaluated configurations for next budget (SH).

*Fig. 2. Hyperband with Successive Halving for the Next Budgets [6]*

Typically, HB outperforms Random Search and BO over smaller budgets, the reason is, it evaluates bigger amount of configurations at the same time because of the usage of objective function approximation. As budget goes higher, BH typically it’s worse than others algorithms.

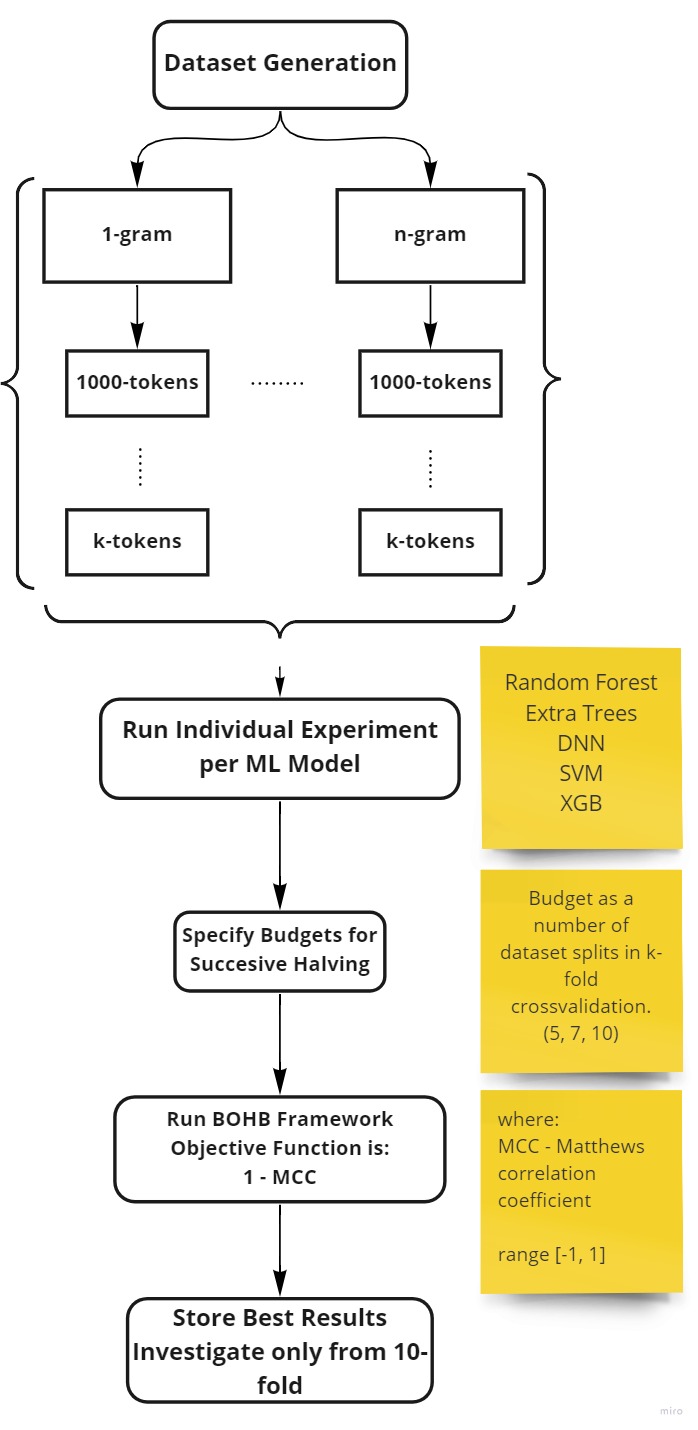
*Fig. 3. Hyperband Performance Compared to Random Search [6]*

**3.3 Robust and Efficient Hyperparameter Optimization at Scale.**

BOHB combines both Bayesian Optimization and Hyperband techniques at once. In the early steps it replaces random configuration sampling from Hyperband to model based sampling from BO. It significantly improves later evaluations for higher budgets and achieves very similar results on smaller budgets comparing to standalone Hyperband technique.

*Fig. 4. BOHB Performance Comparison [6]*

1. EXPERIMENT ARCHITECTURE

Below are presented the whole experiment architecture, all of the experiment components used and every step to reproduce. It is worth mention that somehow the results of this experiment are not fully reproducible. To be more specific, writer used all of the necessary preparation to keep reproducibility on the highest possible level as setting random seeds for models training and data shuffling but the specific behavior of Hyperbandster framework do not guarantee to have the same results after every experiment run. However, this is not a critical point at all, Hyperbandster framework is used only to expose the best possible hyperparameter configurations, in that case, all of the best configurations could be reused on (out of framework) additional experiments to confirm hyperparameters configuration reproducibility.

*Fig. 5. High Level Experiment Architecture*

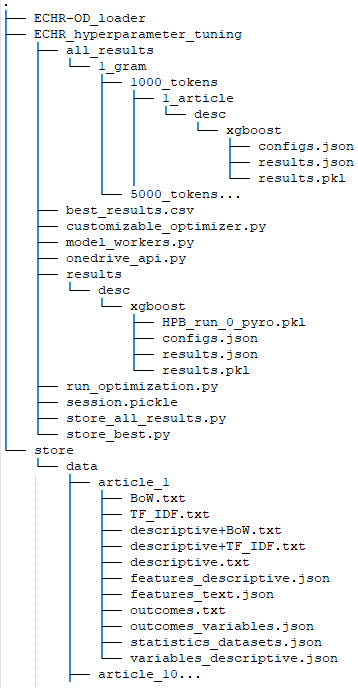
**4.1 Procedure**

Basically the whole procedure is described below:

1. Dataset Generation Process – based on previously created scripts from ECHR-OD project, writer generates range of binary datasets for further computation. Datasets consist of several parameters: (n-grams, k-tokens, t-flavors) where n ∈ {1,2,3,4,5,6}, k ∈ {1000, 5000, 7000, 10000, 30000, 60000, 80000, 100000} and t ∈ {Descriptive, Bag of Words, Descriptive + Bag of Words}. In addition, each of the parameter combination is presented in individual article from which dataset was generated. There are 11 articles. So the overall combination number that this experiment should handle is: n \* k \* t \* (articles number) = 1584. By combination number, writer have in mind the total number of whole experiment repetitions, this not includes the number of tested ML models and BOHB budget runs. If the case is to know the total BOHB runs in overall, this should be calculated as BOHB\_tr = n \* k \* t \* (articles number) \* (number of tested models) \* (BOHB budget runs) = 79200.
2. Hyperbandster Framework Part - BOHB\_tr is pretty huge, still if there is a need to run such a computation it will take really long time. The partial solution for that is to redistribute / parallelize computations. Parallelization is done by splitting computation per ML model and each flavor for different threads. In addition, huge advantage goes along with HB algorithm when during early phase only approximation of the objective function is computed. This computation is available by successively increasing k-fold crossvalidation parameter, because many algorithms need a bigger computing time during fitting large amount of data. It is worth mention that Hyperbandster framework always minimizes the objective function. MCC (Matthews correlation coefficient) is a desired metric to optimize, it tells Data Scientist much more adequate information about model performance than simple accuracy metric. The numeric range for MCC is between -1 and 1, bigger equals better. Here comes a little change in an objective function to optimize, instead of simple MCC, algorithm takes 1-MCC formula to maximize MCC score.   
   MCC consists of number of True Positives, True Negatives, False Positives and False Negatives.
3. Results – as a whole process generates results for every computed k-fold crossvalidation, only the results from 10-fold are taken into consideration in a further comparison.

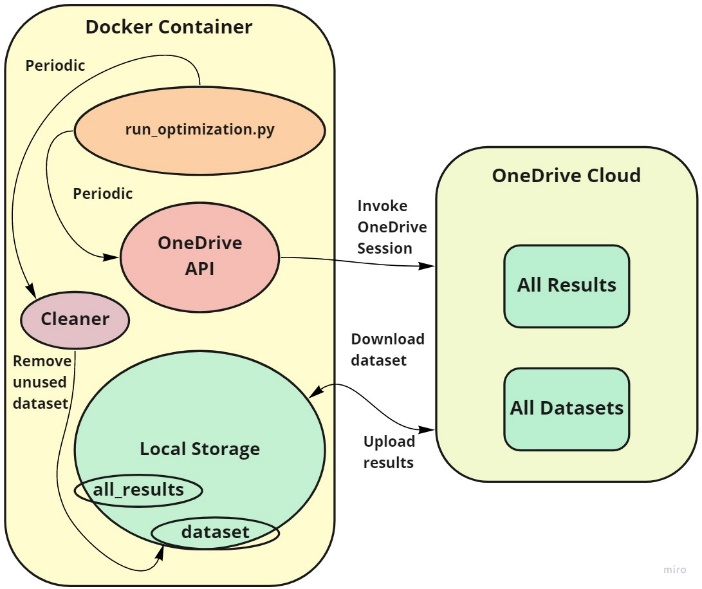
For Low Level Experiment Architecture please consider to look at the experiment Github [5] repository.

**4.2 Docker Container**

For easy reproduction and computation resources separation Docker image is built with all of the experiment components already preinstalled.

*Fig. 6. Docker Directory Structure*

To be able to use this environment, it is only need to install a Docker and run “amasend/echr:version2” image. [Figure 6] presents a container directory structure where experiments are held and results are stored temporary. To be able to run a whole process, two simple steps should be made.

* As the results are automatically stored in a cloud environment (Microsoft OneDrive), there is a need to request an appropriate session file from an experiment owner. In the early release, there is no any option to generate session file by your own. All of the results are also stored locally under /ECHR\_hyperparameter\_tuning/all\_results but to be able to compute them, session need to be established with a cloud where datasets reside.
* “run\_optimization.py” script should be invoked from a python executable

*Fig.7. Datasets and Results Storage Structure*

[Figure 7] shows a high level process of local to cloud storage communication and synchronization.

1. TIME OBSTACLES

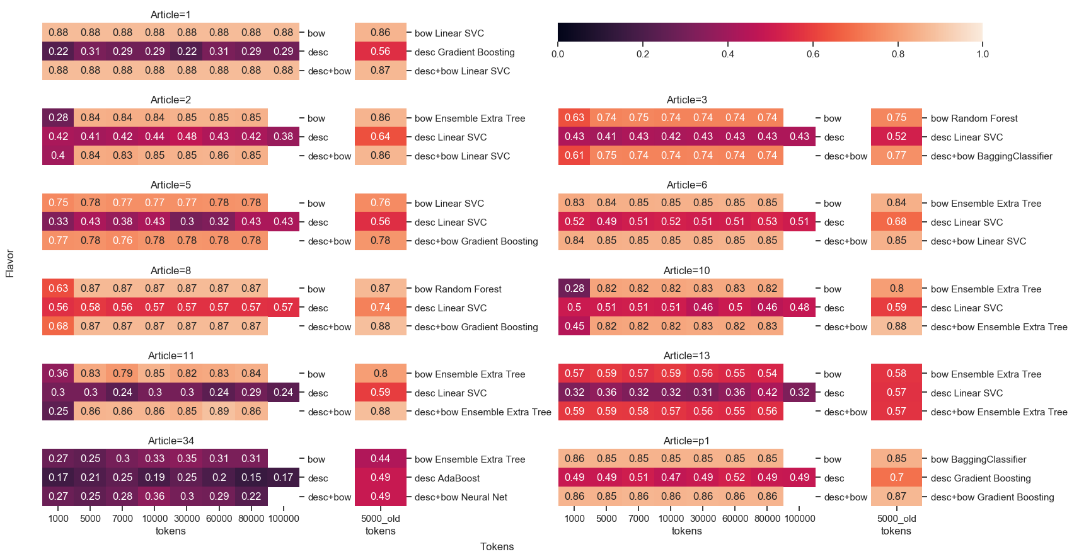
One of the greatest obstacles related to ML models evaluation is a computing time. In this experiment it is no different. As earlier mentioned, huge number of iterations creates tremendous barrier for obtaining the results fast. Unfortunately, writer did not have opportunity to use distributed computing system during experiment period, instead, standalone Linux server with 64GB of RAM and 2 Intel Xeon CPUs with 16 threads was used. Even that configuration is not powerful enough to handle all of the computation at once. The end result is prioritization of some of the promising algorithms that show the best results for small n-grams and k-tokens. Further comparison is based only on a part of the parameters configuration (n-grams, k-tokens, t-flavors) for some of ML models. Full computing is done only for two of the ML models: Random Forest and Extreme Gradient Boosting which are as follows the fastest and the best in the meaning of score.

1. RESULTS COMPARISON

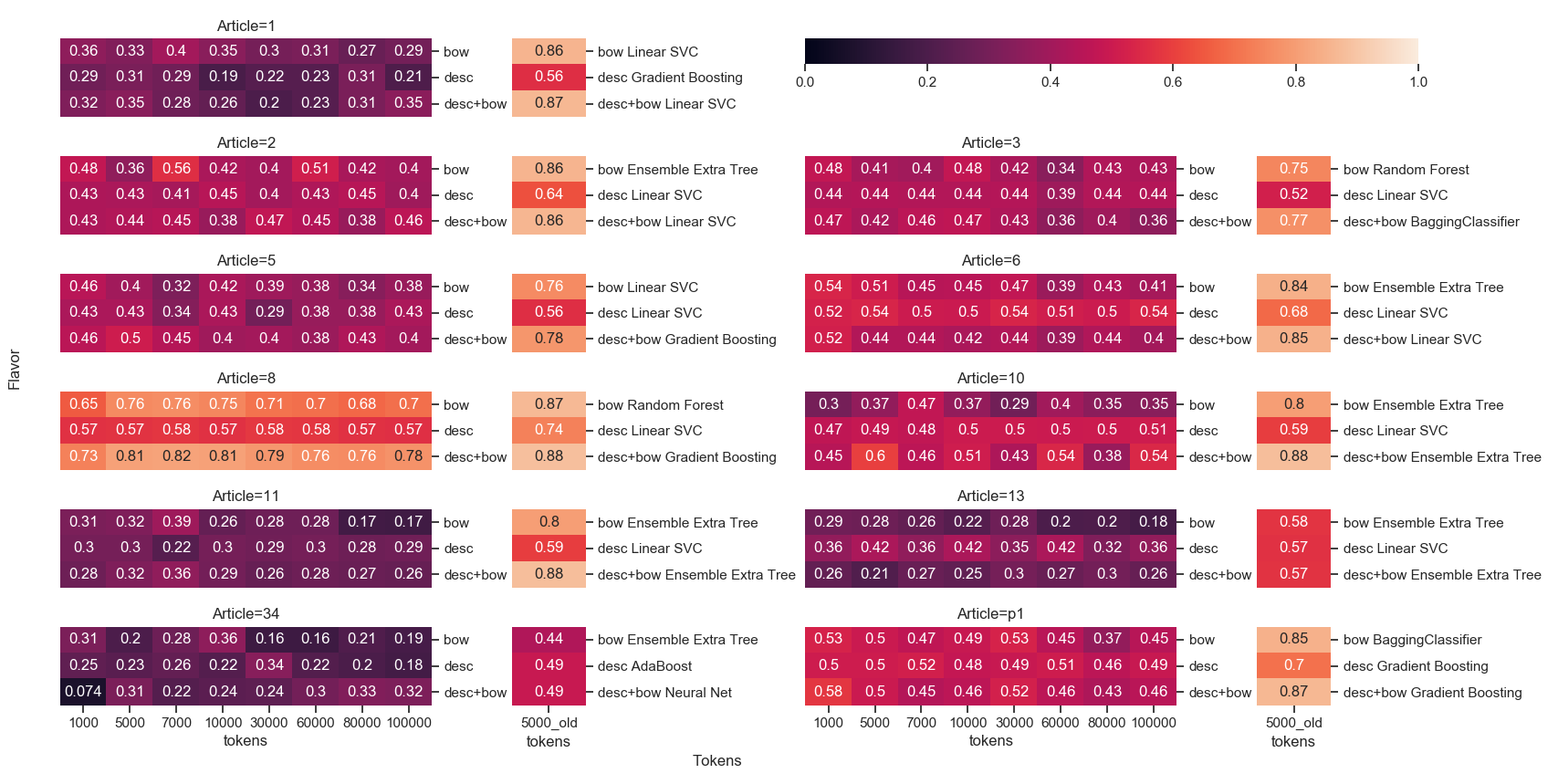
Below are presented the partial results from this experiment, also there is a direct comparison between new partial results and old the best results [Fig. 8 and Fig.9]. The reason why there are only partial results is lack of time for producing all. Two first figures [Fig. 8 and Fig. 9] present only Random Forest model evaluation based on the best MCC metric for particular n-gram and for all flavors and tokens. As reader can clearly see, Random Forest was not the most appearing model with the best result for old experiment but should have in mind that after hyperparameter tuning, that model is really close to beat old results or it is even better in some cases. Unfortunately, there is a huge gap in a descriptive flavor even after hyperparameter tuning, still it could be the reason of the size of training data per case. [Fig. 9] also depict how much important is n-gram parameter, lower n-gram is much worse than higher one [Fig.8]. There can be seen one additional option for speed up computing time after all, because a descriptive flavor is not dependent on a n-gram component, it could be computed only once. [Fig. 10] depicts small partial results from XGBoost model. Even if there are not fully computed results, reader can see a huge improvement comparing to old the best results, especially with descriptive flavor what is the most important part of this experiment. Nearly for all articles XGBoost beat old the best models. Unfortunately, XGBoost computation is relatively slow.

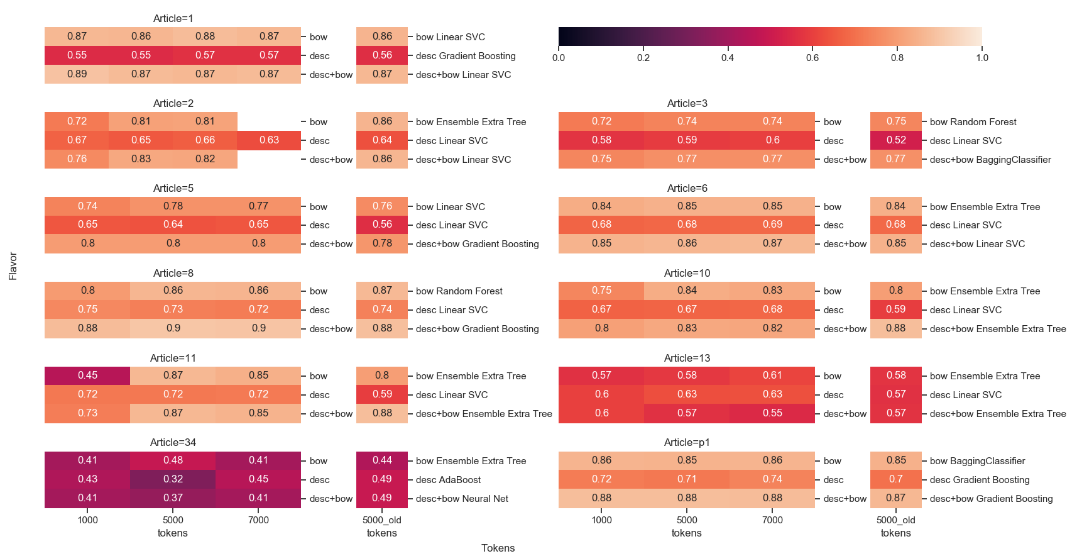
[Fig. 11, Fig. 12, Fig. 13] show confusion matrices for Random Forest and XGBoost models. As clearly depicted, descriptive flavor is really hard for classification, algorithms mostly perform overfit on training data and cannot correctly predict most of the hard cases. Comparing to bag of words flavor, XGBoost model rather doing fine on it and there are no signs of overfit.

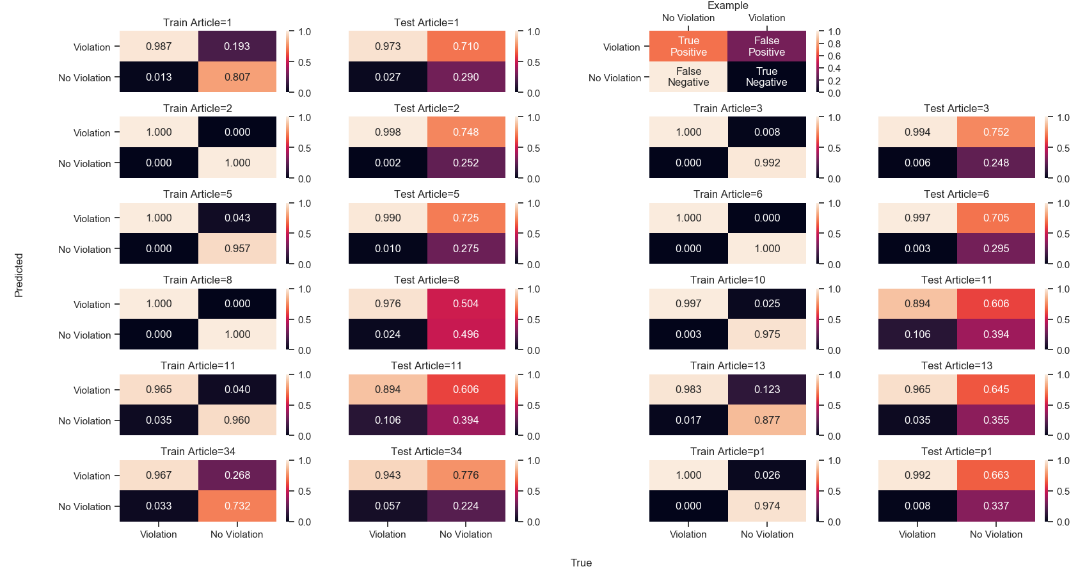
*Fig. 8. Random Forest 5-grams All Tokens Comparison to Old Best Results (4-grams)*

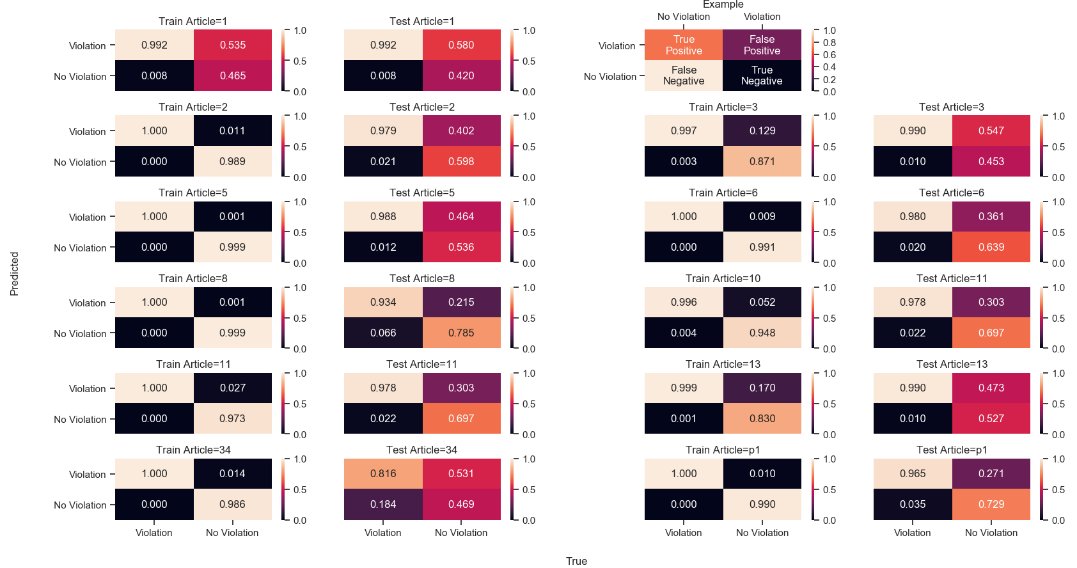


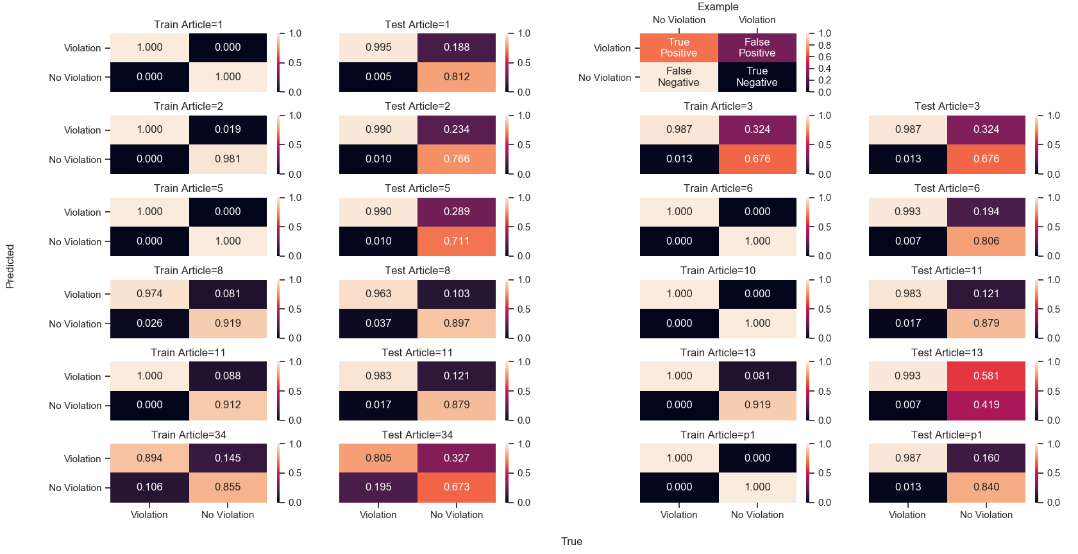
*Fig. 9. Random Forest 1-gram All Tokens Comparison to Old Best Results (4-grams)*

*Fig. 10. XGBoost 5-gram (1000, 5000, 7000)Tokens Comparison to Old Best Results (4-grams)*



*Fig. 11. Random Forest* *Confusion Matrices Train and Test for 5 n-grams and 5000 tokens (desc flavor)*

*Fig. 12. XGBoost Confusion Matrices Train and Test for 5 n-grams and 5000 tokens* (desc flavor)

 *Fig. 13. XGBoost Confusion Matrices Train and Test for 5 n-grams and 5000 tokens* (bow flavor)

1. SUMMARY

This document wants to show one of the available automated method for hyperparameter tuning in various machine learning algorithms. It is mostly focused on expanding the previous research about European Court of Human Rights cases classification. To be able to achieve better and more consistent results whole new work was done in the meaning of new python scripts and usage of an automated optimization framework along with Docker containers and cloud storage capability. The new partial results depict a huge improvement comparing to the old experiment what is the clear confirmation that the process of an automated hyperparameter tuning is advisable for such situations.

1. BIBLIOGRAPHY

[1.] HUDOC *European Court of Human Rights*   
https://hudoc.echr.coe.int/eng#{%22documentcollectionid2%22:[%22GRANDCHAMBER%22,%22CHAMBER%22]}  
Accessed: 30.05.2019

[2.] ECHR-OD\_predictions   
Author: Alexandre Quemy  
https://github.com/echr-od/ECHR-OD\_predictions  
Accessed: 30.05.2019

[3.] AutoML  
https://www.automl.org/  
Accessed: 30.05.2019

[4.] BOHB: *Robust and Efficient Hyperparameter Optimization at Scale*  
Authors: Stefan Falkner, Aaron Klein, Frank Hutter  
https://ml.informatik.uni-freiburg.de/papers/18-ICML-BOHB.pdf  
Accessed: 30.05.2019

[5.] ECHR\_hyperparameter\_tuning  
Author: Amadeusz Masny  
https://github.com/amasend/ECHR\_hyperparameter\_tuning  
Accessed: 30.05.2019

[6.] BOHB blog post  
https://www.automl.org/blog\_bohb/  
Accessed: 30.05.2019