Deep Forest

Katarína Nocarová

My thesis

- Deep forest and reactions
- Results from article, newer results and new implementation
- Compare DF to other classifiesrs
- Experiments with different hyperparameter settings
- AutoML
- Improvements, future work

MUNI
FI

Article

All based on article named Deep Forest by Zhi-Hua Zhou, Ji Feng from Nanjing University
https://arxiv.org/pdf/1702.08835.pdf
(2017/2020)

MUNI
FI

The ideas

- Success of deep neural networks

- Disadvantages of neural networks (hyperparameters, time and data consuming, blackbox, predetermined model complexity...)

- Success of ensemble methods and Random Forest

- Deepness rather than neurons – layer by layer processing, model complexity and in-model feature transformation as means to representation learning ability

gcForest

- Random Forest implementation
- Cascade Forest Structure
- Multi-grained scanning

Cascade Forest Structure

- Each level processes information and outputs results to a new layer
- Each layer as an ensemble of different decision tree forests – ensemble of ensembles
- Different forests for diversity (such as random forest and completely-random tree forest)
- Each forest will produce an estimate of class distribution, by counting the percentage of different classes of training examples at the leaf node where the concerned instance falls, and then averaging across all trees in the same forest

Class vector generation

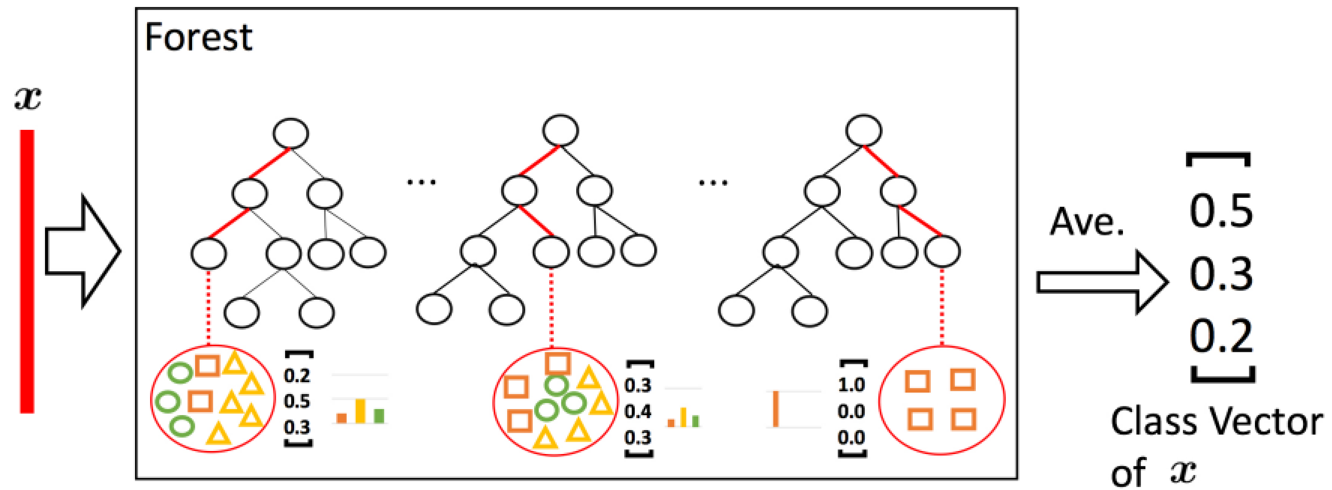


Fig. 3. Illustration of class vector generation. Different marks in leaf nodes imply different classes.

Example

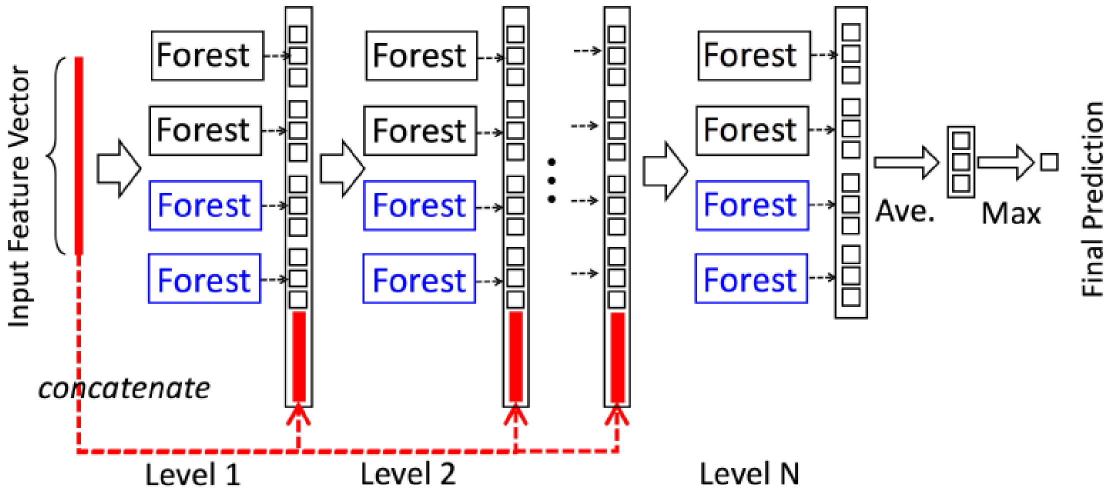


Fig. 2. Illustration of the cascade forest structure. Suppose each level of the cascade consists of two random forests (black) and two completely-random tree forests (blue). Suppose there are three classes to predict; thus, each forest will output a three-dimensional class vector, which is then concatenated for re-representation of the original input.

Multi-Grained Scanning

Using sliding windows to scan the raw features

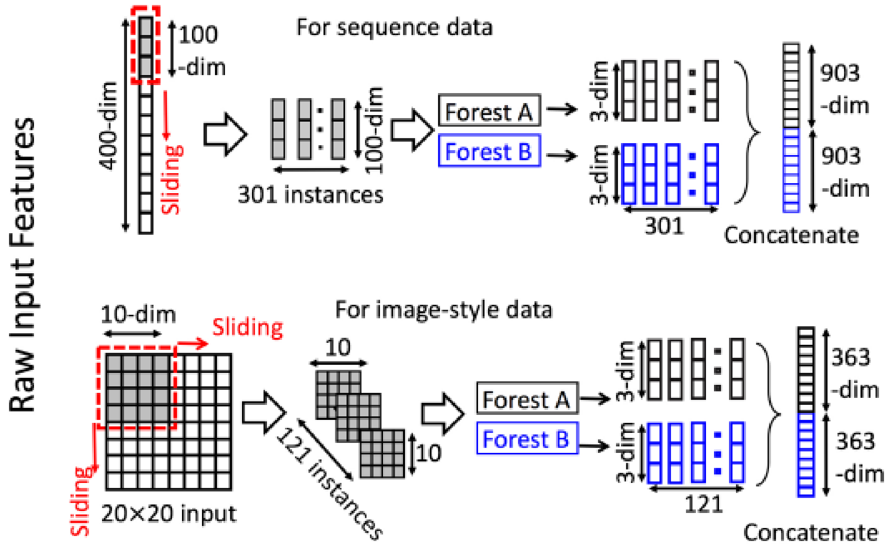


Fig. 4. Illustration of feature re-representation using sliding window scanning. Suppose there are three classes, raw features are 400-dim, and sliding window is 100-dim.

Overall procedure

- Multi-grained scanning
- Cascade forests
- Adding layers untill convergence of validation performance
- Final prediction will be obtained by aggregating the class vectors at the last level

MUNI FI

Overall procedure

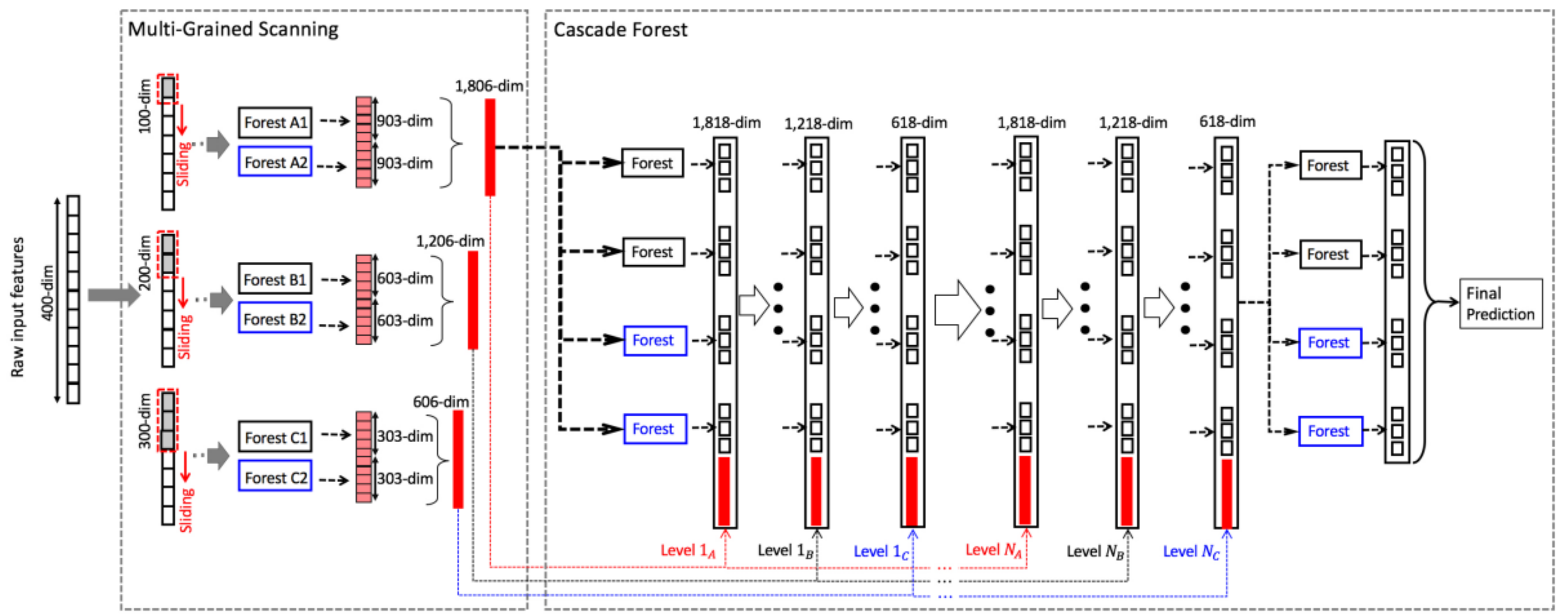
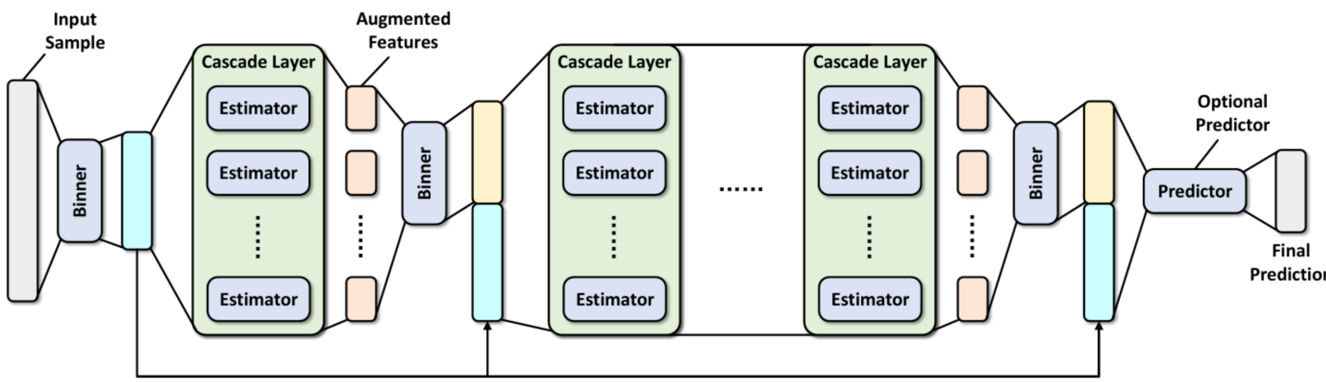


Fig. 5. The overall procedure of gcForest. Suppose there are three classes to predict, raw features are 400-dim, and three sizes of sliding windows are used.

DF21

- Newer implementation – improvement on efficiency
- https://github.com/LAMDA-NJU/Deep-Forest
- Documentation: https://deep-forest.readthedocs.io/en/stable/

```
pip install deep-forest
```



How to use

Classification

```python
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from deepforest import CascadeForestClassifier

X, y = load_digits(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
model = CascadeForestClassifier(random_state=1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
acc = accuracy_score(y_test, y_pred) * 100
print("\nTesting Accuracy: {:.3f} %".format(acc))
>>> Testing Accuracy: 98.667 %
```

Results

- From article promising – comparable to neural networks

- Several models in different fields (examples)
  - Anti-cancer drug response https://www.sciencedirect.com/science/article/abs/pii/S1046202318303232
  - E-commerce consumers' repurchase behavior https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0255906
  - Detection of Cash-Out Fraud https://dl.acm.org/doi/abs/10.1145/3342241
  - Classification of cancer subtypes https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2095-4

Their results

GTZAN	accuracy
gcForest	65.67
CNN	59.2
MLO	58
Random Forest	50.33
Logistic Regression	50
SVM (rbf kernel)	18.33
DF21	51.26667

MNIST	accuracy
gcForest	99.26
LeNet-5	99.05
Deep Belief Net	98.75
SVM (rbf kernel)	98.6
Random Forest	96.8
DF21	98.066

Their results

sEMG	accuracy
gcForest	71.3
MLP	45.37
Random Forest	38.52
SVM (rbf kernel)	29.62
Logistic Regression	23.33
DF21	33.96296

ORL	5	7	9
DF21	0.957	0.973333	98.5
gcForest	91	96.67	97.5
Random Forest	91	93.33	95
CNN	86	91.67	95
SVM	80.5	82.5	85
kNN	76	83.33	92.5

Their results

IMDB	accuracy
gcForest	89.16
CNN	89.02
MLP	88.04
Logistic Regression	88.62
SVM (linear kernel)	87.56
Random Forest	85.32
DF21 (reduced)	88.704

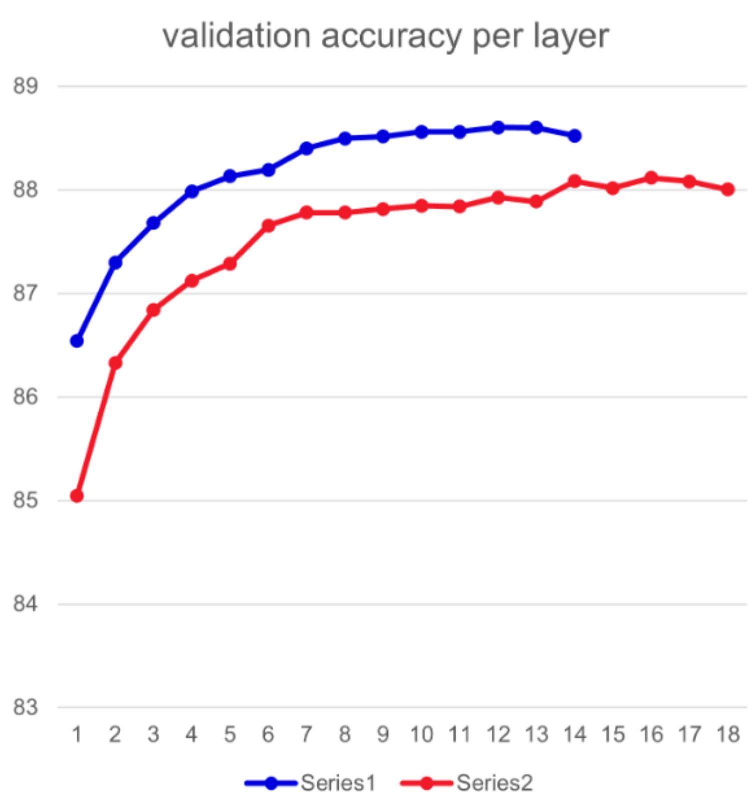
CIFAR-10	accuracy
ResNet	93.57
AlexNet	83
gcForest(gbdt)	69
gcForets(5grains)	63.37
Deep Belief Net	62.2
gcForest(default)	61.78
Random Forest	50.17
MLP	42.2
Logistic Regression	37.32
SVM (linear kernel)	16.32
DF21	51.19

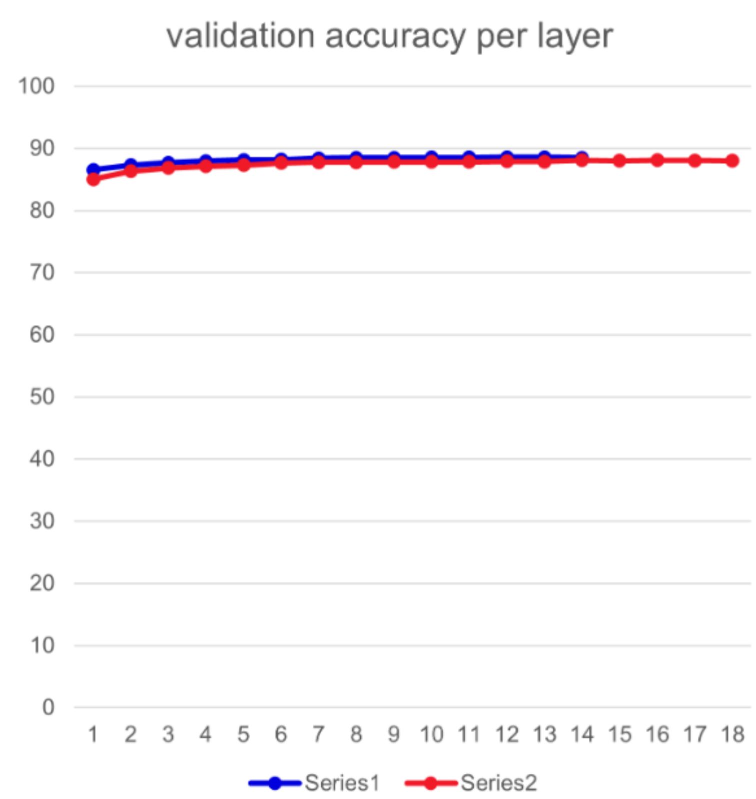
Their results

Low dimension	LETTER	ADULT	YEAST
gcForest	97.4	86.4	63.45
Random Forest	96.5	85.49	61.66
MLP	95.7	85.25	55.6
DF21	97.195	86.063	59.50673

Multi-grained scanning influence	MNIST	GTZAN	sEMG
gcForest	99.26	65.67	71.3
CascadeForest	98.02	52.33	48.15
DF21	98.06	51.26	33.96

Layers


validation accuracy per layer
89
88
87
86
85
84
83
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Series1
Series2


validation accuracy per layer
100
90
80
70
60
50
40
30
20
10
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Series1
Series2

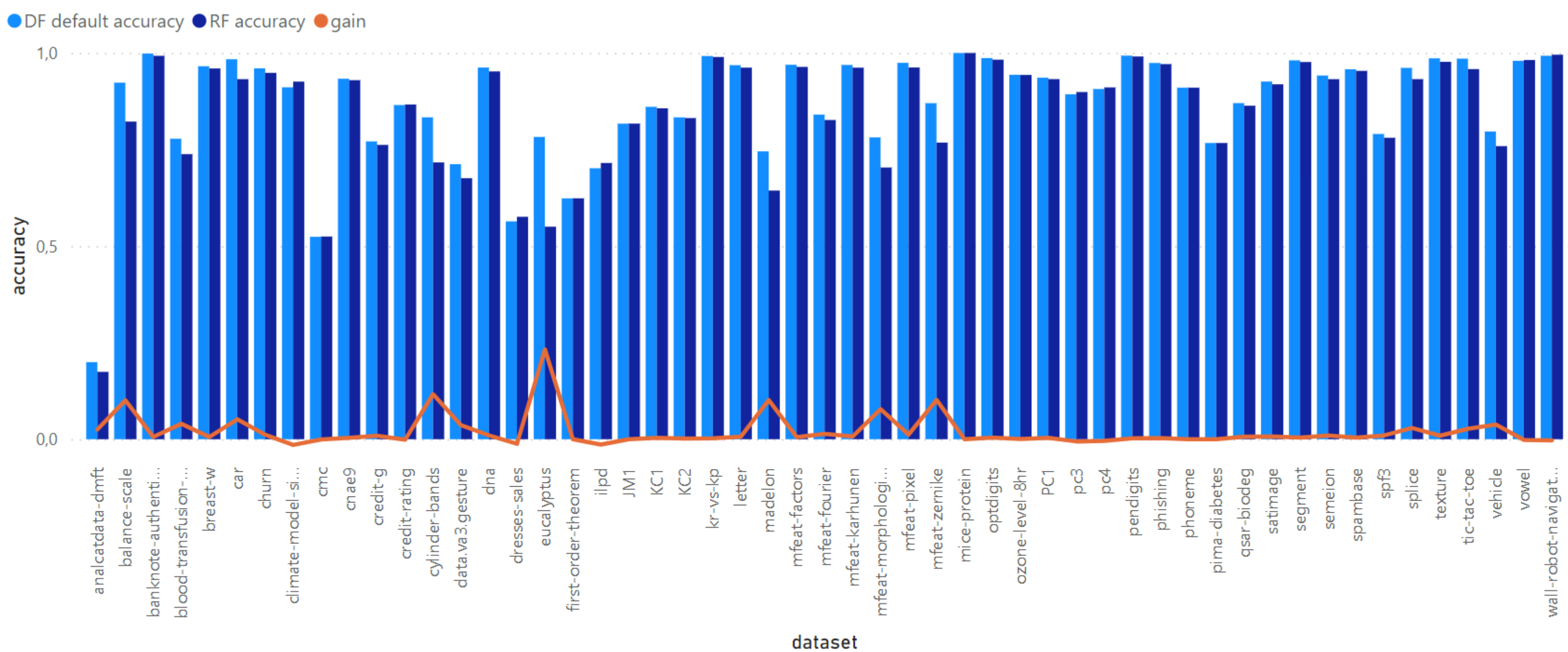
Compared to other classifiers

- 10 other classifiers, 55 datasets

- Better than average in 85% cases
- Average gain 5.5%
- 47% best accuracy
- Better than Random Forest in 72%, average difference 0.019
  Comparable runtime* (on average 3 times slower)

MUNI FI

Compared to Random Forest



Parameter tuning

- https://deep-forest.readthedocs.io/en/stable/parameters_tunning.html#
- For accuracy (n_estimators, n_trees, max_layers, use_predictor)
- Faster speed (n_jobs, n_bins, max_deapth, n_estimators, n_trees, min_samples_leaf, n_tolerant_rounds)
- Lower memory (partial_mode)

MUNI FI

Parameters

- class deepforest.CascadeForestClassifier(n_bins=255, bin_subsample=200000, **bin_type='percentile', max_layers=20**, **criterion='gini', n_estimators=2, n_trees=100,** max_depth=None, min_samples_split=2, min_samples_leaf=1, use_predictor=False, predictor='forest', predictor_kwargs={}, backend='custom', n_tolerant_rounds=2, delta=1e-05, partial_mode=False, n_jobs=None, random_state=None, verbose=1)

Experiments with parameters

- 2*2*3*4*4*4 = 768 parameter settings with grid search
- 57 datasets -> 43 776 runs
- 5-fold cross validation -> 218 880 runs

MUNI FI

Bin_type and criterion

bin_type	Average of time	Average of accuracy	occurences in best quartile
interval	20,34	0,8714	4572
percentile	21,64	0,8727	5613

criterion	Average of time	Average of accuracy	occurences in best quarile
entropy	21,08	0,8722	5194
gini	20,90	0,8719	4991

MUNI
FI

Max_layers and n_estimators

max_layers	Average of time	Average of accuracy	occurences in best quartile
3	17,35	0,8717	3031
5	22,66	0,8722	3577
20	22,95	0,8722	3577

n_estimators	Average of time	Average of accuracy	occurences in best quartile
1	6,75	0,8723	3027
2	12,74	0,8723	2656
4	25,37	0,8718	2276
6	39,10	0,8718	2226

N_trees

n_trees	Average of time	Average of accuracy	occurences in best quarile
100	5,08	0,8720	2404
200	9,67	0,8724	2724
500	23,33	0,8722	2578
1000	45,87	0,8716	2479

Predictor and use_predictor

predictor	Average of time	Average of accuracy	occurences in best quartile
forest	23,27	0,8726	2358
lightgbm	19,46	0,8713	2691
none	21,29	0,8745	3357
xgboost	19,94	0,8699	1779

use_predictor	Average of time	Average of accuracy
False	21,29	0,8745
True	20,89	0,8712

MUNI
FI

Metalearning

- To create a model able to recommend parameters for a selected dataset based on its metadata
- To reduce the amount of parameter combinations needed

MUNI
FI

Thank you for attention

Main source: https://arxiv.org/pdf/1702.08835.pdf