
autoMLk Documentation

Release 0.0.1a

pierre-chaville

Apr 08, 2018

1	Content	3
1.1	User guide	3
1.1.1	Home	3
1.1.2	Dataset	3
1.1.3	Results and best models	6
1.2	Installation	19
1.2.1	Pre-requisites	19
1.2.2	Installation	22
1.2.3	Basic installation	22
1.2.4	Advanced configuration	23
1.3	Architecture	26
1.4	DataSet	26
1.5	Searching	28
1.6	List of models	28
1.6.1	Models level 1	28
1.6.2	Ensembles	29
1.7	Pre-processing steps	30
1.7.1	categorical encoding:	30
1.7.2	text encoding:	31
1.7.3	imputing missing values:	31
1.7.4	feature scaling:	31
1.7.5	feature selection:	31
2	Indices	33

This toolkit is designed to be integrated within a python project, but also independently through the interface of the app.

The framework is built with principles from auto-sklearn, with the following improvements:

- web interface (flask) to review the datasets, the search results and graphs
- include sklearn models, but also Xgboost, LightGBM, CatBoost and keras Neural Networks
- 2nd level ensembling with model selection and stacking
- can be used in competition mode (to generate a submit file from a test set), on benchmark mode (separate train set and public set) and standard mode.

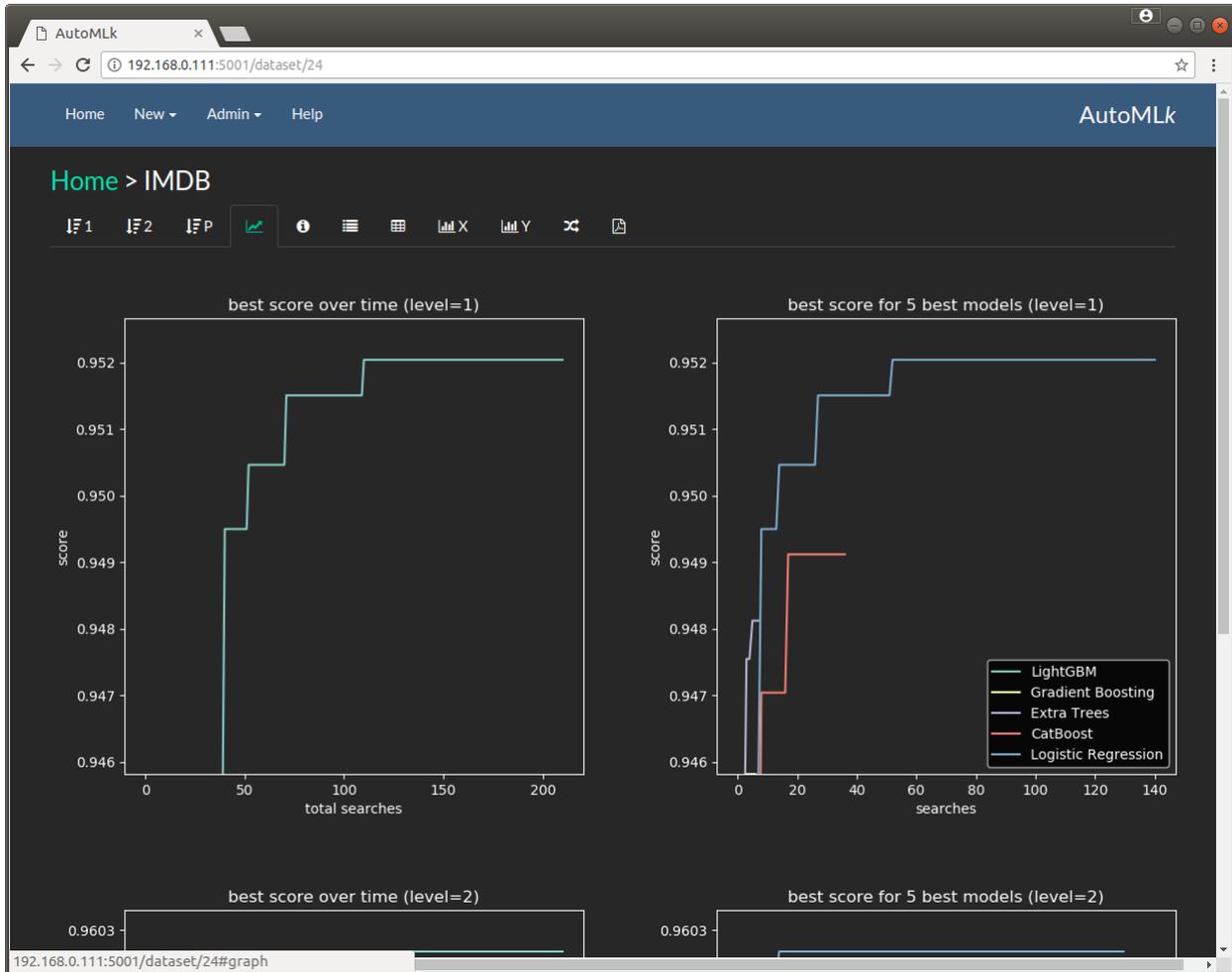


Fig. 1: Best models by eval score

We have provided some public datasets to initialize the framework and compare results with best scores.

1.1 User guide

The dataset and the results of the search are best viewed with the web app through a standard browsers.

to start the app, please go in the web folder and run the app server:

```
python run.py
```

then access the app in a browser with the follwoing url:

```
http://localhost:5001
```

or from another machine with the ip address of the machine where the server is running:

```
http://192.168.0.10:5001
```

(in this example, we suppose the address of the server is 192.168.0.10)

1.1.1 Home

The home page shows the list of datasets:

You can select a list of datasets from a specific domain, with the selector at the top right:

1.1.2 Dataset

To import the list of preloaded datasets (or your own list), you can select the option 'Import' in the menu 'New':

You may create directly a dataset by using the 'Dataset' option in the menu 'New':

You may afterwards update *some fields* of a dataset by using the edit icon in the list of datasets in the home page:

name	status	#	best result	description	actions
Benchmarks/Text					
IMDB (copy)	searching	100 118	Logistic Regression	0.94906 +/-0.0024 0.9464	classification 25 rows x 3 cols, 1 text cols
IMDB	completed	389 4437	Stacking Logistic Regression	0.96026 +/-0.002 0.95765	classification 25 rows x 3 cols, 1 text cols
Benchmarks/Basic					
Ames Housing	searching	100 102	Linear Regression	0.13543 +/-0.0175 0.15955	regression 1 rows x 81 cols, 54 categ. cols, 19 missing cols
Titanic	searching	100 102	Extra Trees	0.83705 +/-0.0142 0.827	classification 0 rows x 12 cols, 4 categ. cols, 3 missing cols
Abalone	searching	100 103	CatBoost	2.10999 +/-0.0766 2.20692	regression 4 rows x 9 cols, 1 categ. cols
Wine (classification)	searching	90 104	Extra Trees	0.8963 +/-0.0327 0.9414	classification 6 rows x 13 cols, 1 categ. cols
Wine	searching	100 155	Extra Trees	0.39147 +/-0.0296 0.4247	regression 6 rows x 13 cols, 1 categ. cols
Bank Marketing	searching	101 132	LightGBM	0.95036 +/-0.0009	classification 41 rows x 21 cols, 10 categ. cols

Fig. 1.1: list of datasets in autoMLk

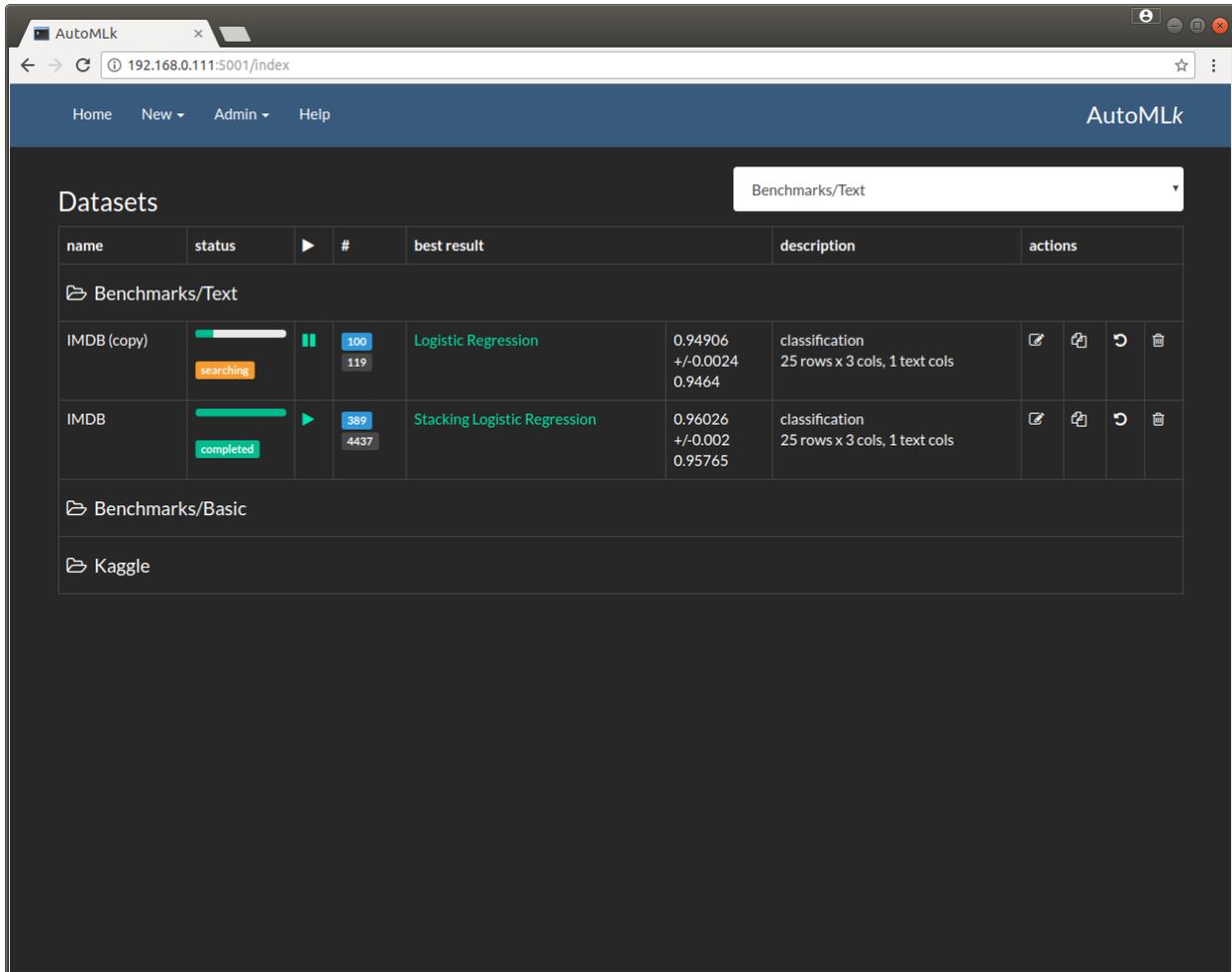


Fig. 1.2: list of datasets per domain

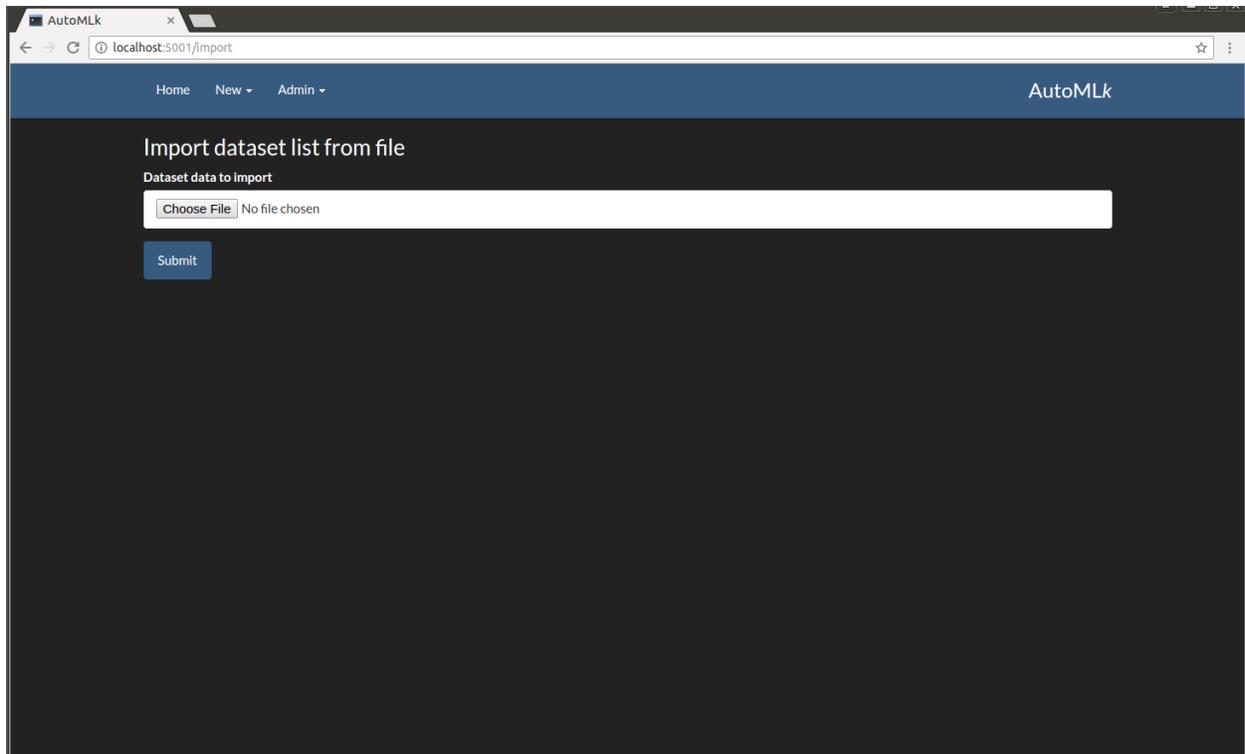


Fig. 1.3: import a list of datasets

We can access to a specific dataset in clicking on the row of the required dataset. When a dataset is created, there is only the features and analysis of the data available:

By clicking on the various tabs, we can view:

We need to launch the search process with various models in order to access to be results

1.1.3 Results and best models

When the search is launched, 3 additional tabs are available:

And per pre-processing steps:

The graph of the best results over time:

And after a while, the best ensembles:

The best ensembles

And then by clicking on a specific model access to the details

And then on a specific round:

Where we can view the performance and the predictions:

The screenshot shows a web browser window with the URL `localhost:5001/create`. The page title is "AutoMLk". The navigation menu includes "Home", "New dataset", "Import", and "Monitor". The main heading is "Create new dataset".

The form is organized into two columns of input fields:

- Left Column:**
 - Name:
 - Description:
 - Source:
 - Public dataset
 - URL:
 - Problem type:
 - Metric:
 - Other metrics:
- Right Column:**
 - Columns file:
 - Train set file:
 - Test set file:
 - Submit set file:
 - Submit column:
 - Folds cv:
 - Holdout %:
 - Y column:

Fig. 1.4: create a new dataset

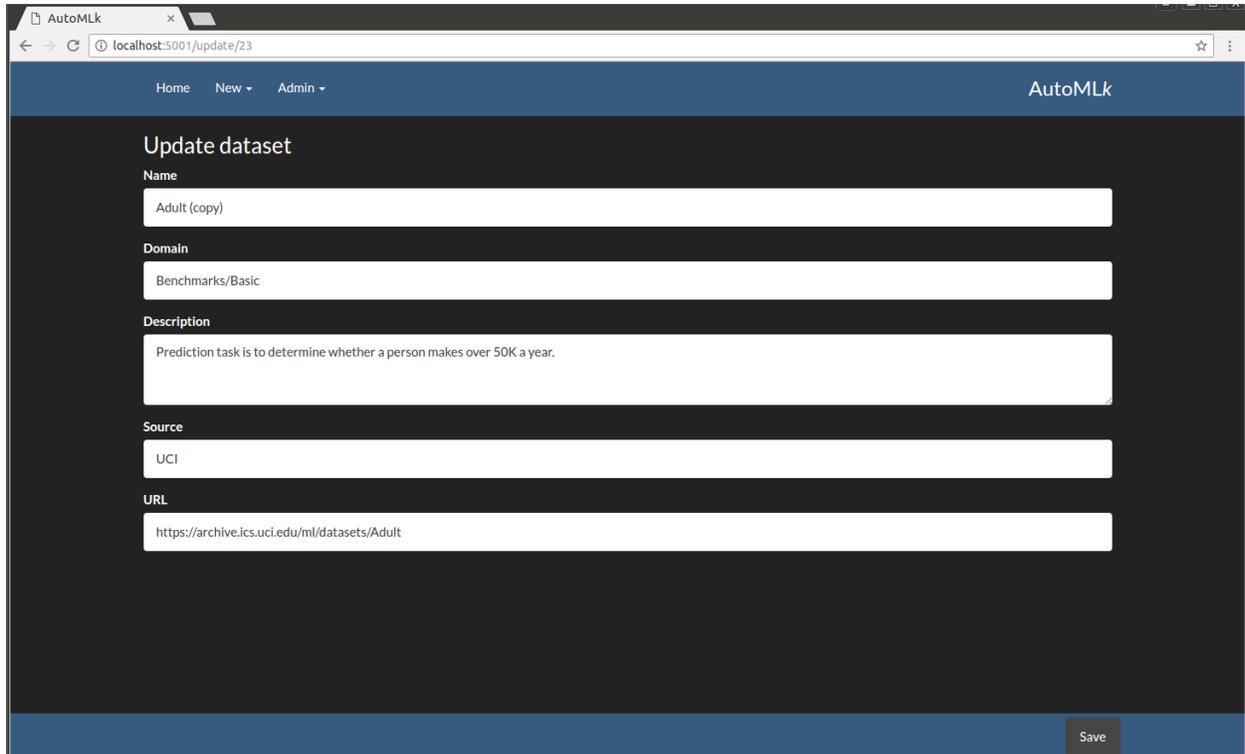


Fig. 1.5: update a dataset

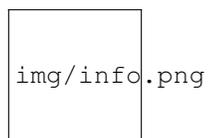


Fig. 1.6: parameters of the dataset

Home > IMDB

NB: edit features is deactivated because the search has started

name	description	keep	raw_type	type	missing	unique	values
id	id	✗	object	categorical		25000	5814_8, 2381_9, 7759_3, 3630_4, 9495_8
sentiment	Sentiment of the review; 1 for positive reviews and 0 for negative reviews	✓	int64	numerical		2	1, 0
review	Text of the review	✓	object	text		24904	With all this stuff going down at the moment with MJ i've started listening to his music, watching the odd documentary here and there, watched The Wiz and watched Moonwalker again. Maybe i just want t...

Fig. 1.7: the list of features of the dataset

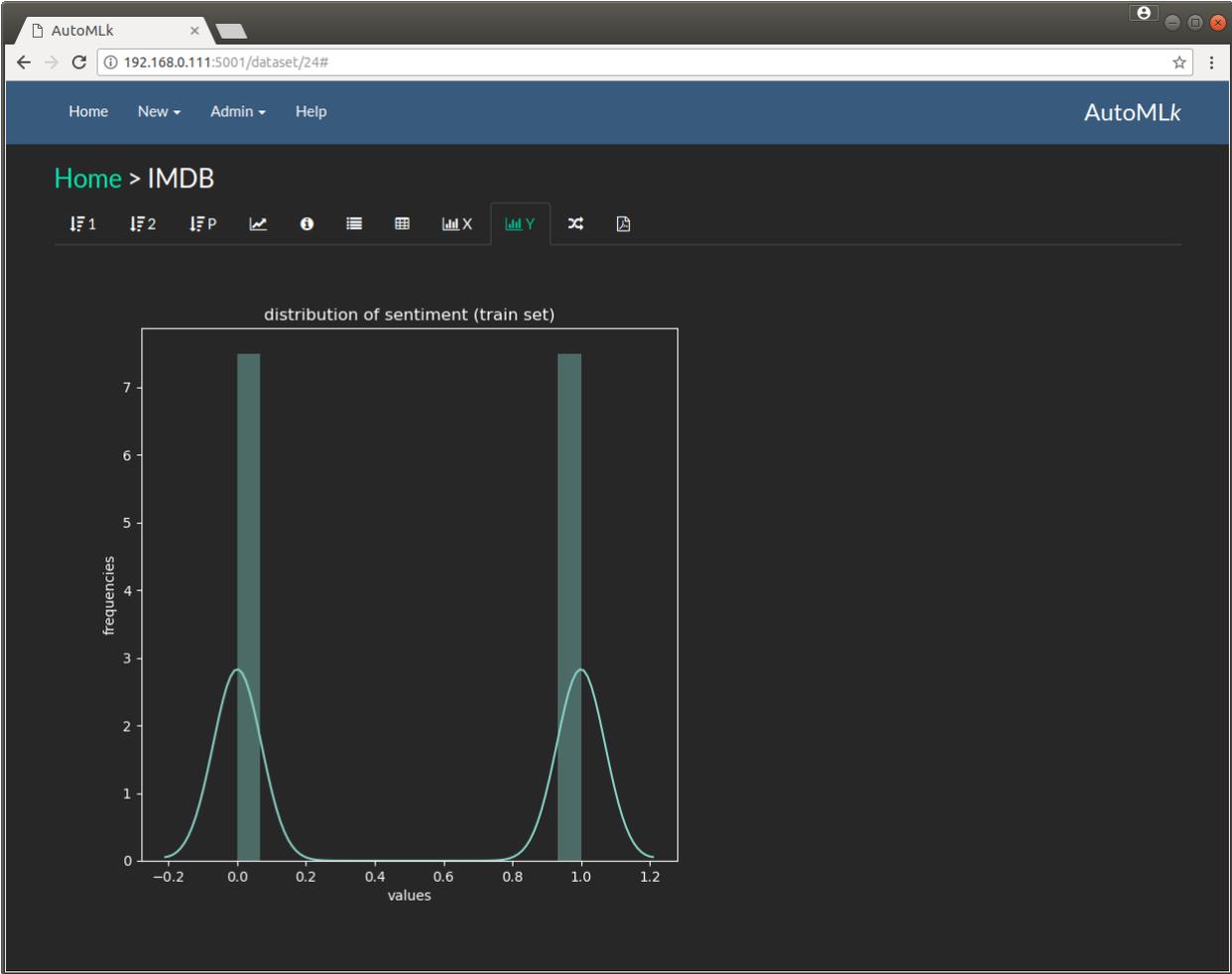


Fig. 1.8: the histogram of the target column



Fig. 1.12: The evolution of the best scores in time

found in 6 models and 182 configurations

model	cv	eval	test	other	#	duration	params
Stacking Logistic Regression	0.96026 +/-0.002 0.95765	0.96013	0.96167	accuracy: 0.89505, precision: 0.89462, recall: 0.8956, f1: 0.89511	38	0.0s 3.0s	penalty: l2, dual: False, tol: 0.0001, C: 0.0909, fit intercept: True, intercept scaling: 1, solver: sag, max iter: 1000, multi class: ovr
Stacking Gradient Boosting	0.95967 +/-0.0019 0.95696	0.95946	0.96094	accuracy: 0.89535, precision: 0.89295, recall: 0.8984, f1: 0.89567	25	0.0s 42.0s	n estimators: 296, max features: auto, max depth: None, min samples split: 2, min samples leaf: 0.2403, min weight fraction leaf: 0.0394, max leaf nodes: None, min impurity decrease: 0, learning rate: 0.1, criterion: mse, loss: exponential
Stacking LightGBM	0.95945 +/-0.0018 0.95695	0.95928	0.96062	accuracy: 0.8973, precision: 0.8943, recall: 0.9011, f1: 0.89769	41	0.0s 11.0s	boosting: gbd, learning rate: 0.0309, num leaves: 195, tree learner: serial, max depth: 74, min data in leaf: 38, min sum hessian in leaf: 0.001, feature fraction: 0.206, bagging fraction: 0.8528, lambda l1: 0, lambda l2: 0.2056, min gain to split: 0.1856, drop rate: 0.4651, skip drop: 0.4251, max drop: 50, uniform drop: True, xgboost dart mode: False, top rate: 0.2, other rate: 0.1, objective: binary, metric: binary_logloss, max bin: 255
Stacking XgBoost	0.95935 +/-0.0019 0.95669	0.95915	0.96041	accuracy: 0.89535, precision: 0.8921, recall: 0.8995, f1: 0.89578	32	0.0s 7.0s	booster: dart, eval metric: logloss, eta: 0.1757, min child weight: 120.0088, max depth: 53, gamma: 0, max delta step: 4, sub sample: 1, colsample bytree: 1, colsample bylevel: 1, lambda: 1, alpha: 0, tree method: approx, sketch eps: 0.03, scale pos weight: 1, objective: binary:logistic
Stacking Random Forest	0.95929 +/-0.0018 0.95675	0.95911	0.96037	accuracy: 0.8953, precision: 0.89193, recall: 0.8996, f1: 0.89575	15	0.0s 2mn7	n estimators: 821, max features: sqrt, max depth: None, min samples split: 2, min samples leaf: 1, min weight fraction leaf: 0, max leaf nodes: None, min impurity decrease: 0, criterion: entropy, class weight: None
Stacking Extra Trees	0.95913 +/-0.0018 0.95661	0.95897	0.95997	accuracy: 0.8955, precision: 0.89166, recall: 0.9004, f1: 0.89601	31	0.0s 15.0s	n estimators: 553, max features: auto, max depth: 32, min samples split: 2, min samples leaf: 1, min weight fraction leaf: 0, max leaf nodes: None, min impurity decrease: 0, criterion: gini, class weight: None

192.168.0.111:5001/dataset/24#ensembles

img/list_model.png

Fig. 1.13: details of the search by model

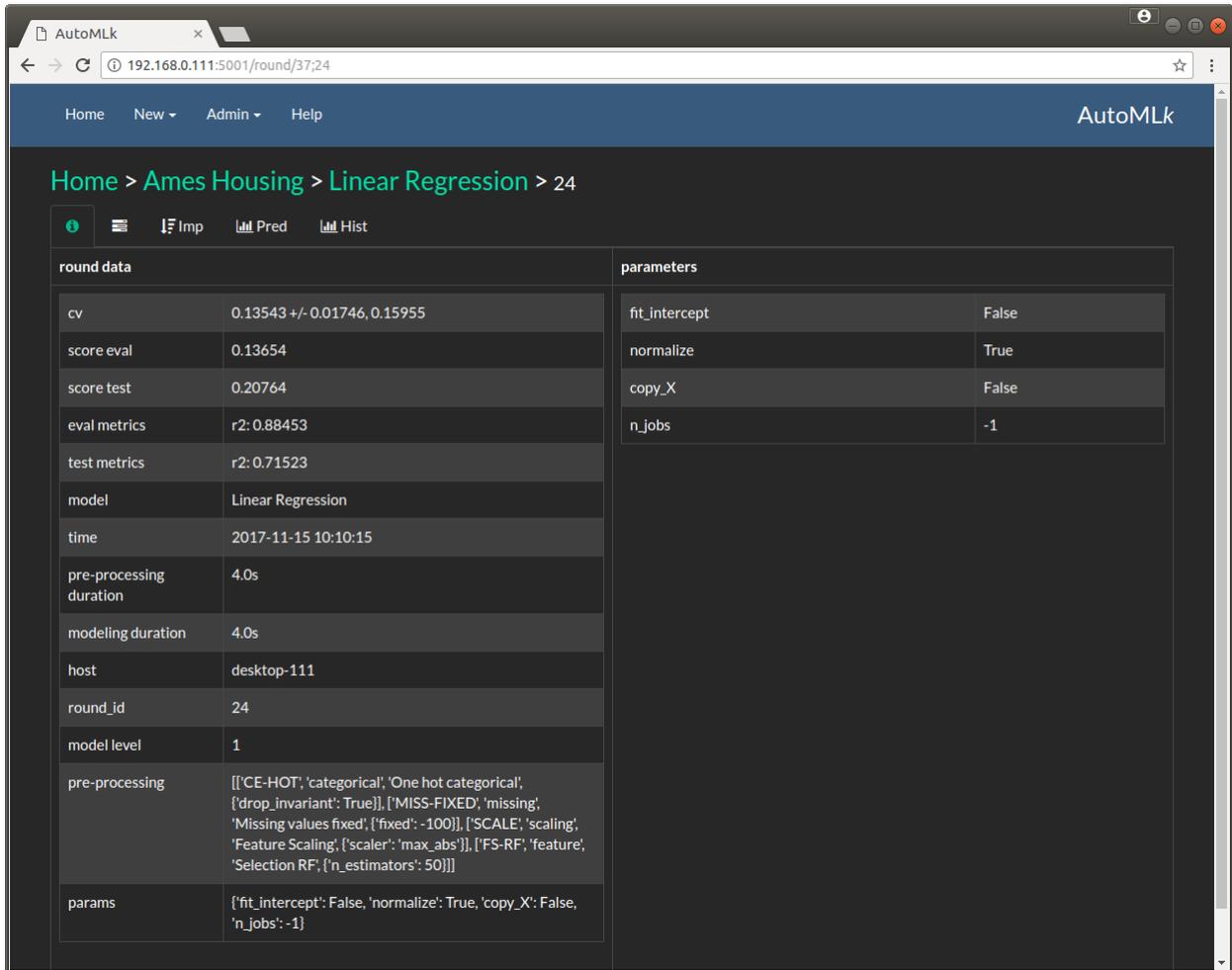


Fig. 1.14: a round with a se of model parameters and pre-processing



Fig. 1.15: details of the re-processing steps

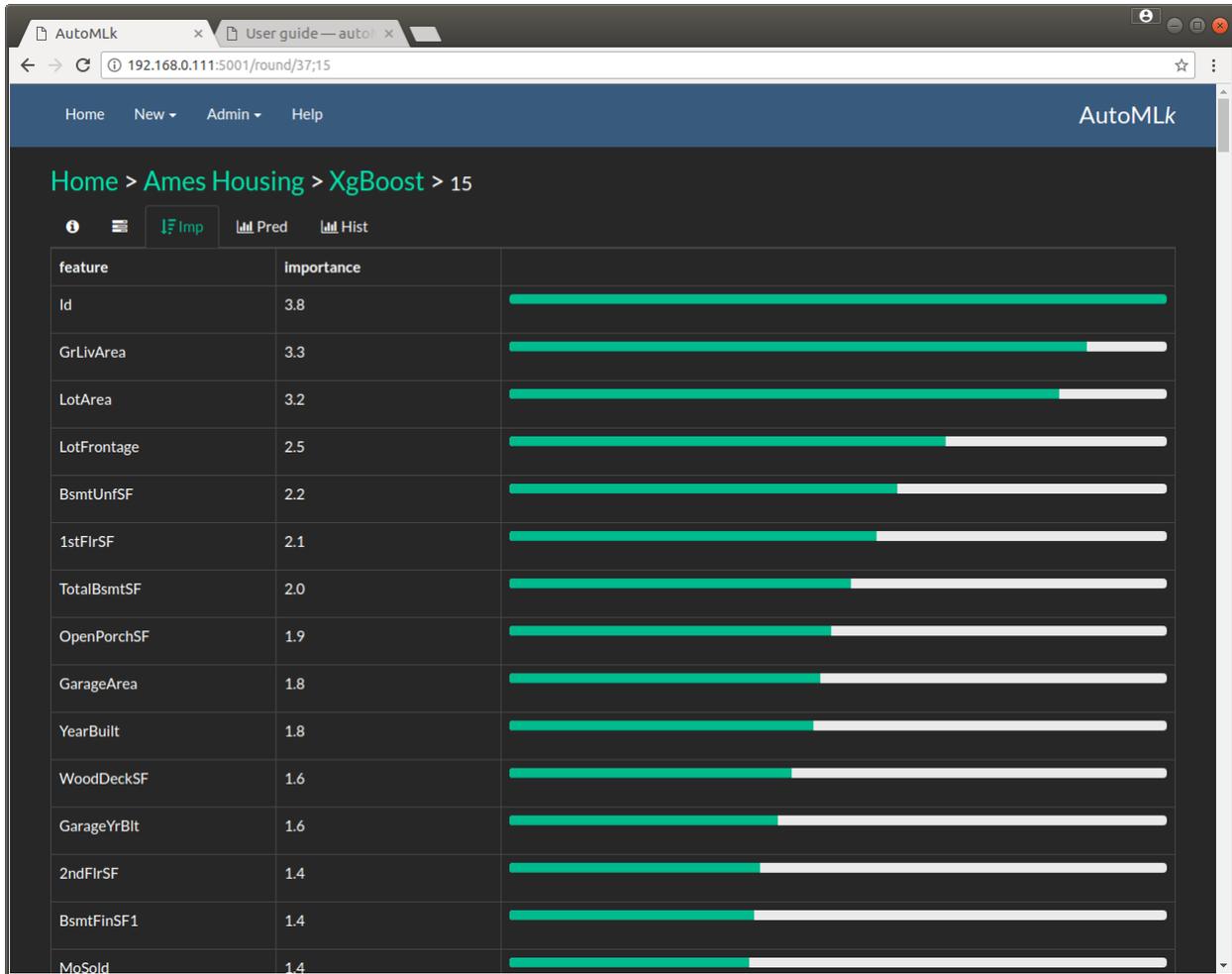


Fig. 1.16: feature importance scored by the model

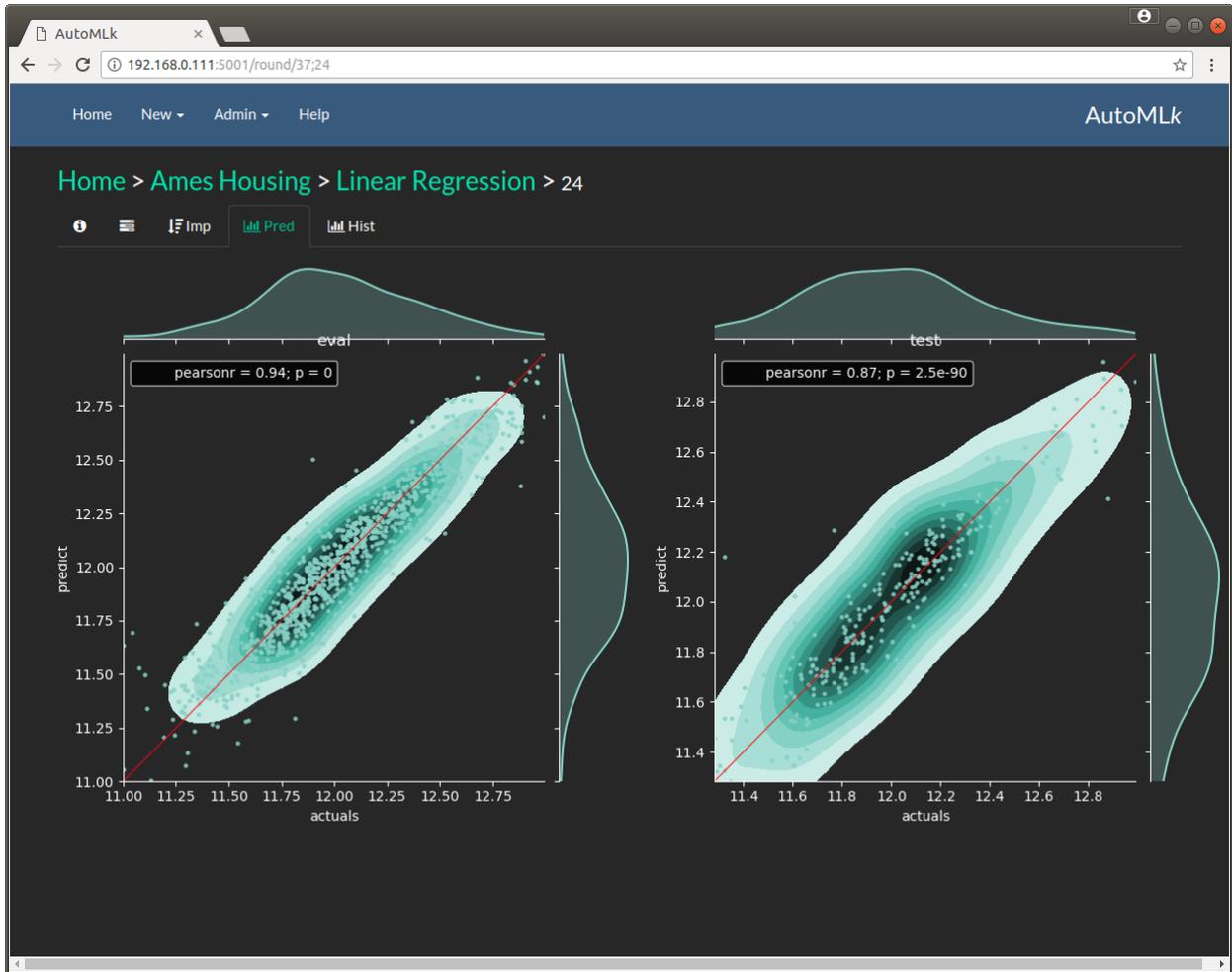


Fig. 1.17: predictions versus actuals (in regression)

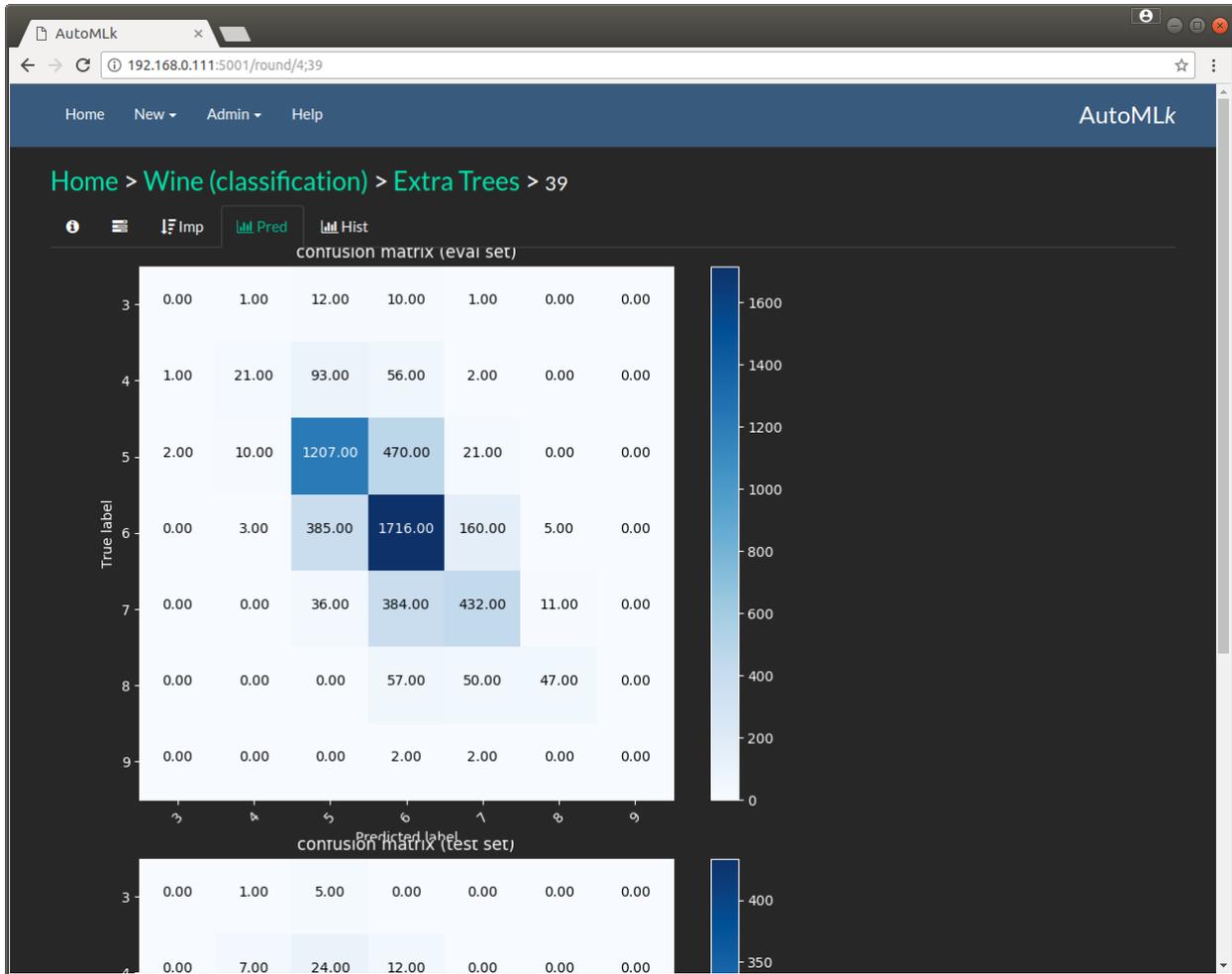


Fig. 1.18: and a confusion matrix (in classification)

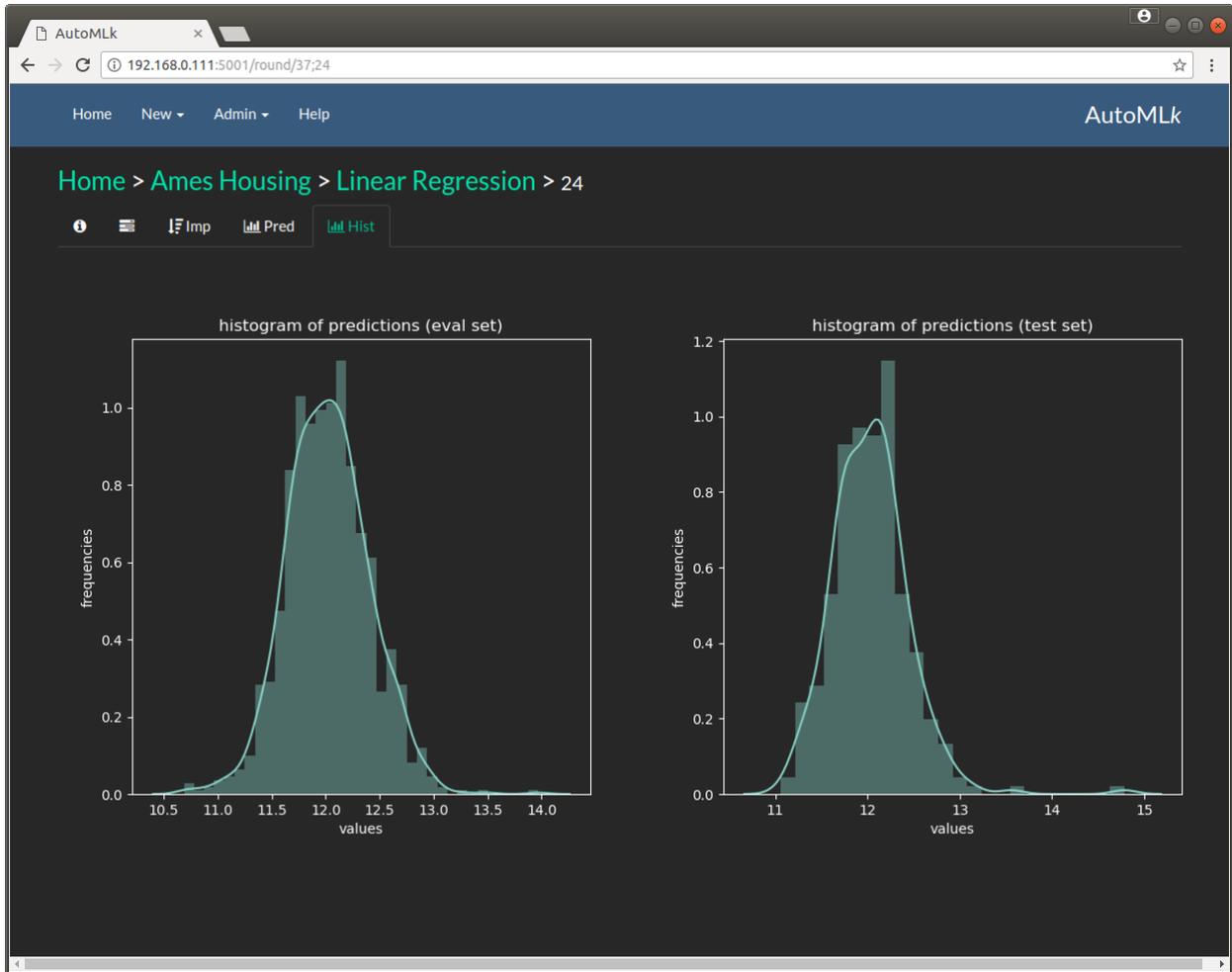


Fig. 1.19: and the histogram of the predictions

Admin

Monitoring

The monitoring screen displays the different status of the different components in the architecture: controller and workers

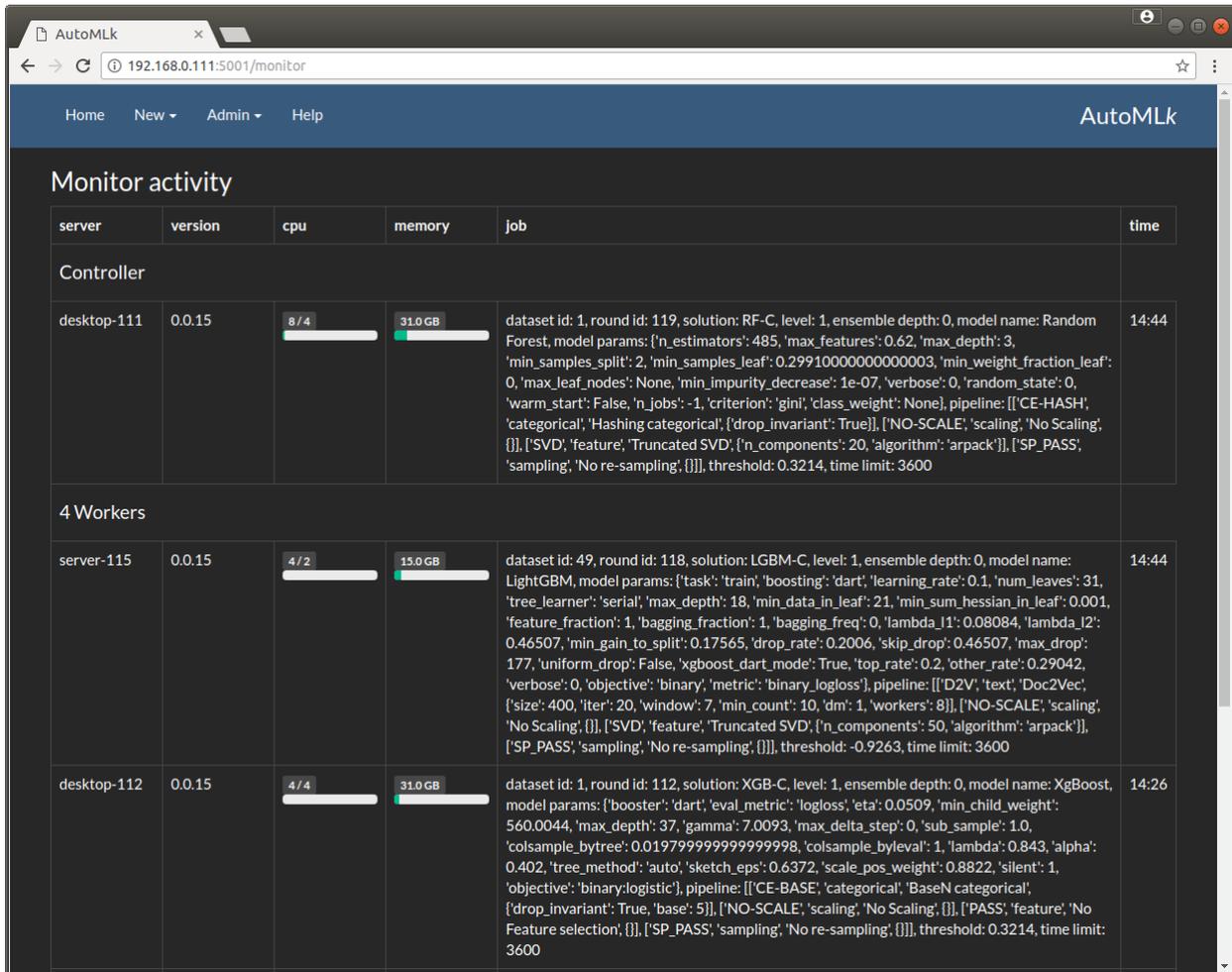


Fig. 1.20: monitoring panel

Config

It is also possible to modify the theme of the user interface directly from the config panel:

1.2 Installation

1.2.1 Pre-requisites

Sklearn version must be > 0.19, otherwise there will be several blocking issues.

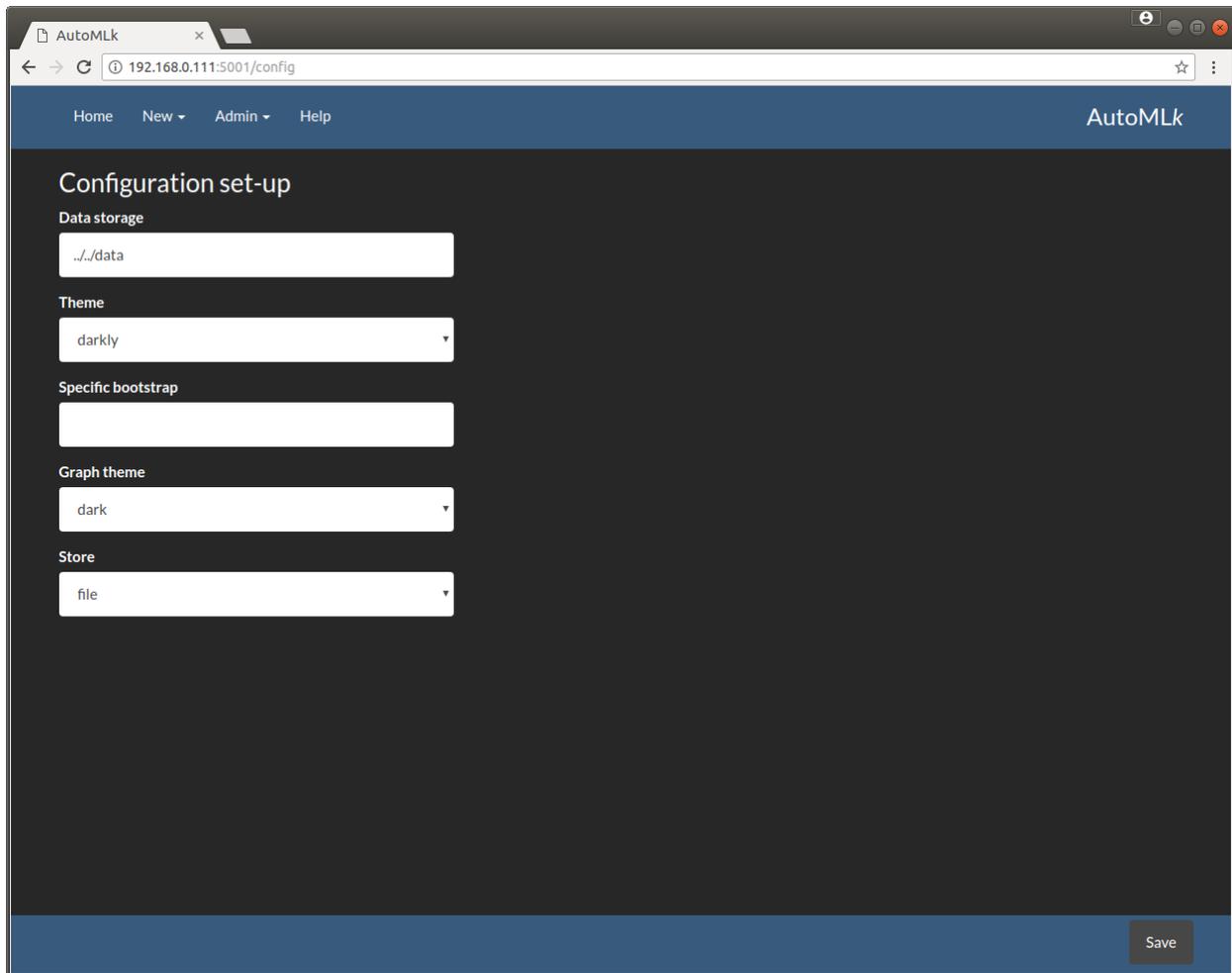


Fig. 1.21: configuration panel

domain	name	status / level	#	description	rows	actions
Benchmarks/Basic	Adult	created	0	Prediction task is to determine whether a person makes over 50K a year...	32 K	[edit] [share] [delete]
Benchmarks/Text	IMDB	pause	71 3774	The labeled data set consists of 50,000 IMDB movie reviews, specially selected for sentiment analysi...	25 K	[edit] [share] [delete]
Benchmarks/Basic	Ames Housing (log)	pause	128 3725	The Ames Housing dataset was compiled by Dean De Cock for use in data science education. It's an inc...	1 K	[edit] [share] [delete]
Benchmarks/Large	Lending Club	searching	39 187	These files contain complete loan data for all loans issued through the 2007-2015, including the cur...	887 K	[edit] [share] [delete]
Benchmarks/Basic	Titanic	searching	54 2600	The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912...	0 K	[edit] [share] [delete]
Benchmarks/Basic	Abalone	completed	414 2013	Predict the age of abalone from physical measurements, from the study: The Population Biology of Aba...	4 K	[edit] [share] [delete]
Benchmarks/Basic	Wine (classification)	pause	348 2365	The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For mor...	6 K	[edit] [share] [delete]
Benchmarks/Basic	Wine	created	9 0	The two datasets are related to red and white variants of the Portuguese "Vinho Verde" wine. For mor...	6 K	[edit] [share] [delete]
Benchmarks/Basic	Bank Marketing	created	9 0	he data is related with direct marketing campaigns of a Portuguese banking institution. The marketin...	41 K	[edit] [share] [delete]
Benchmarks/Basic	Adult (copy) (copy)	pause	9 0	Prediction task is to determine whether a person makes over 50K a year...	32 K	[edit] [share] [delete]

Fig. 1.22: configuration panel

to upgrade scikit-learn:

On conda:

```
conda update conda
conda update scikit-learn
```

If you do not use conda, update with pip:

```
pip install scikit-learn --update
```

Warning: if you use conda, you must absolutely update sklearn with conda

Additionally, you must also install category_encoders and imbalanced-learn:

```
pip install category_encoders
pip install imbalanced-learn
```

Optionally, you may install the following models:

- LightGBM (highly recommended, because it is very quick and efficient):

```
pip install lightgbm
```

- Xgboost (highly recommended, because it is also state of the art):

See Xgboost documentation for installation

- Catboost:

```
pip install catboost
```

- keras with theano or tensorflow:

See keras, theano or tensorflow documentation for installation

1.2.2 Installation

Download the module from github and extract the zip file in a folder (by default automlk-master)

Install as:

```
cd automlk-master
python setup.py install
```

1.2.3 Basic installation

The simplest installation runs on a single machine, with at least the following processes: 1. the web app 2. the controller, grapher and text worker 3. a single worker

These 3 components are run in a console (Windows) or Terminal (Linux).

The basic installation will use a data folder on the same machine. By default, the data folder should be created at one level upper the automlk-master folder.

For example, let's assume that autoMLk is created in the \$HOME (Linux) level or Documents (windows):

- **home**
 - **pierre**
 - * **automlk-master**
 - automlk
 - run
 - web
 - * data

If you want to use a data folder in another location, you can define this in the config screen.

To run the web app:

```
cd automlk-master/web
python run.py
```

This will launch the web app, which can be accessed from a web browser, at the following address:

```
http://localhost:5001
```

From the web app, you can now define the set-up and then import the example of datasets.

You can launch the search in a dataset simply by clicking on the start/pause button in the home screen, and view the results through with the web interface. The search will continue automatically until the search is completed.

To run the controller, grapher et text manager:

```
cd automlk-master/run  
  
python run_controller.py  
python run_grapher.py  
python run_worker_text.py
```

To run the workers on one or multiple machines:

On Linux:

```
cd automlk-master/run  
  
sh worker.sh
```

On Windows:

```
cd automlk-master/run  
  
worker
```

Note: This will run the python module ru_worker.py in an infinite loop, in order to catch the potential crashes from the worker.

1.2.4 Advanced configuration

Data server

The data are stored in a specific folder. In the default configuration, it is supposed to be on the same machine, and in the folder data. You may specify a different machine and location. The configuration is stored in the config.json file
{“data”: “././data”, “theme”: “bootswatch/3.3.7/darkly”, “store”: “file”, “store_url”: “192.168.0.18”}

The data folder must be accessible by all the machines with the following components: - web server - controller - worker

Web server

The web server should be on a separate machine than the workers, in order to guarantee the response times for the user interface.

If you want to use a data folder in another location, you can define this in the config screen.

To run the web app:

```
cd automlk-master/web  
  
python run.py
```

This will launch the web app, which can be accessed from a web browser, at the following address:

```
http://localhost:5001
```

From the web app, you can now define the set-up and then import the example of datasets.

You can launch the search in a dataset simply by clicking on the start/pause button in the home screen, and view the results through with the web interface. The search will continue automatically until the search is completed.

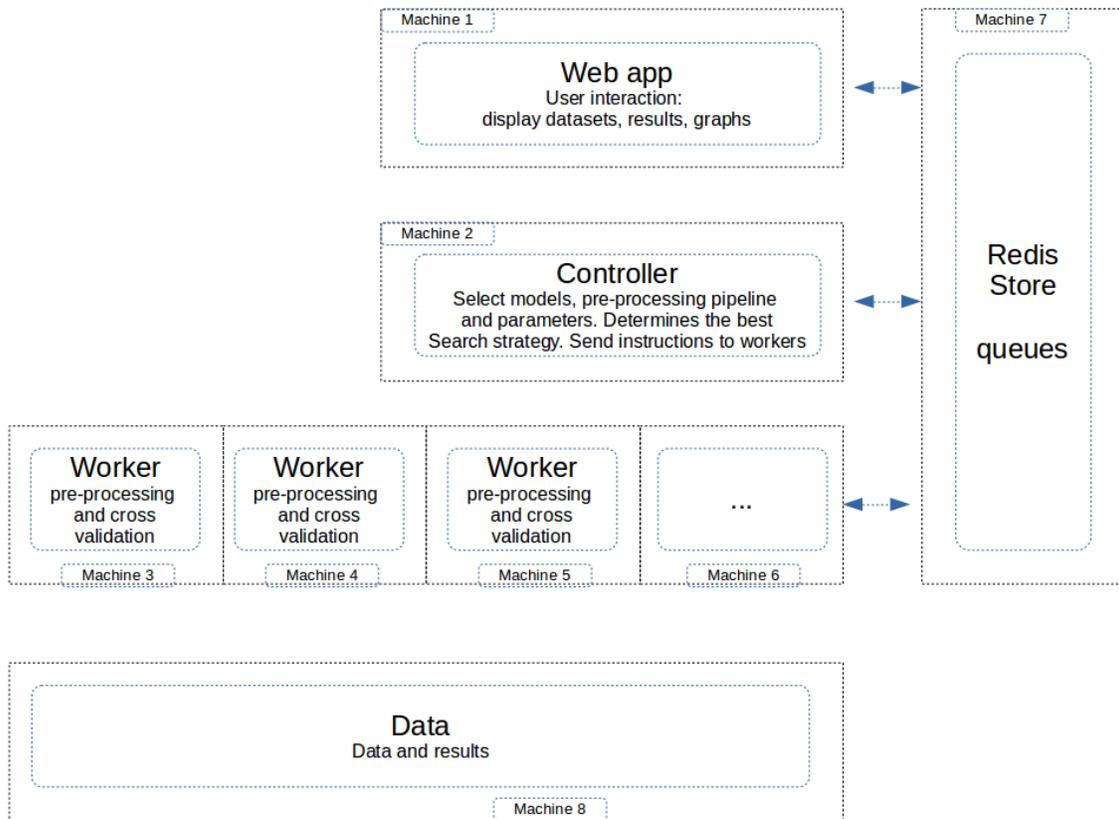


Fig. 1.23: independent components of the architecture

Store

The store by default is implemented using the file system, in the folder data/store, where 'data' is the folder defined for data storage.

The recommended mode is Redis, with the following advantages: - faster user experience of the web app, thanks to the in-memory storage of Redis which is very fast - more robust queuing and communication mechanism between controller and workers.

It is then highly recommended to use Redis for the store, when you have a cluster of multiple workers.

The installation of Redis is simple on Linux machines, and there is also a windows version available. Please see the Redis documentation directly to install and configure your Redis store.

The Redis server can be installed on the same machine as the web server.

Controller, grapher and text worker

The controller can be executed on the machine of the web server. It can also be installed if required on a specific machine.

It must be run in a standalone process, and we recommend that you install this process in a service (windows server) or a permanent process (Linux).

To run the controller:

```
cd automlk-master/run  
  
python run_controller.py  
python run_grapher.py  
python run_worker_text.py
```

Workers

The workers are the components in the architecture with the most significant impact: the speed of search is directly proportional to the number of workers. We recommend to run at least 4 workers, and with multiple datasets to be searched simultaneously, a cluster of 10 to 20 machines should deliver great performance and speed.

To run the worker:

On Linux:

```
cd automlk-master/run  
  
sh worker.sh
```

On Windows:

```
cd automlk-master/run  
  
worker
```

Note: This will run the python module ru_worker.py in an infinite loop, in order to catch the potential crashes from the worker.

1.3 Architecture

The architecture is distributed and can be installed on multiple machines

- the web app for user interaction and display results
- the controller manages the search between models and parameters
- the grapher generates graphs on a dataset asynchronously
- the texter generates unsupervised models for text sets
- the workers execute the pre-processing steps and cross validation (cpu intensive): the more workers are run in parallel, the quicker the results
- the Redis store is an in-memory database and queue manager

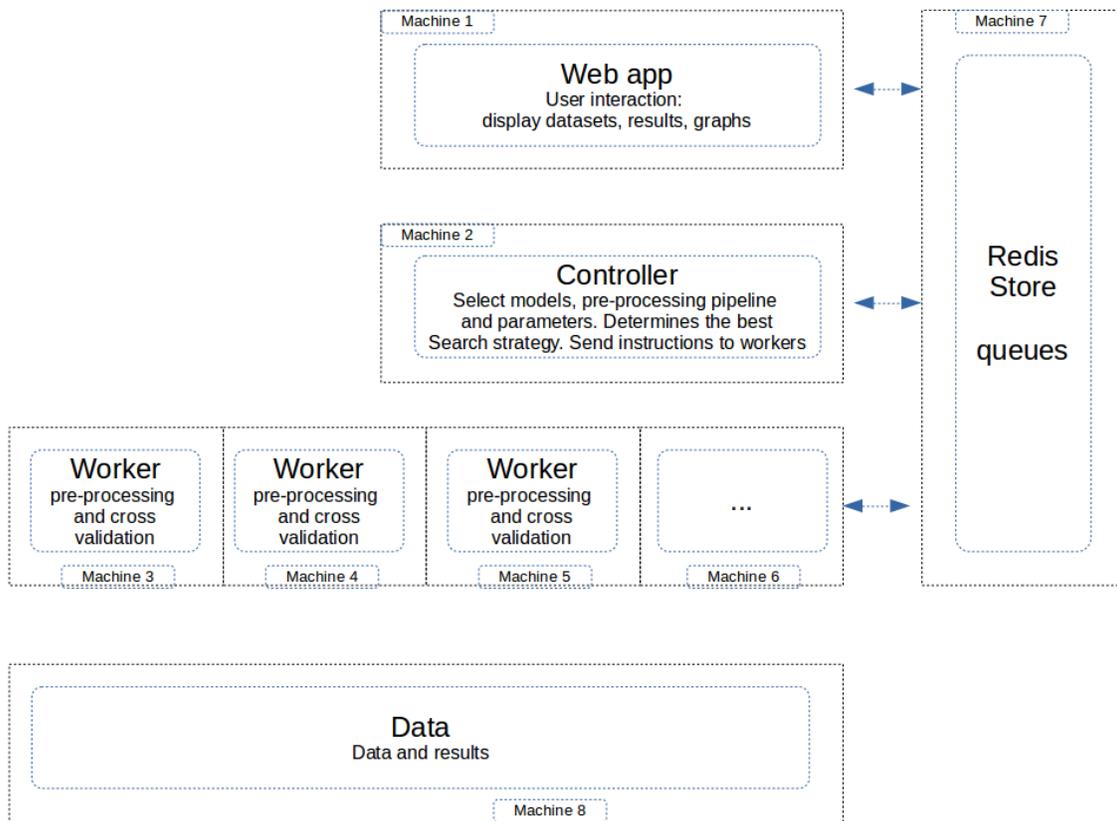


Fig. 1.24: independent components of the architecture

The software architecture is organized in concentric layers:

1.4 DataSet

The features of the automated machine learning are defined and stored in the DataSet object. All features and data of a DataSet object can be viewed with the web app.

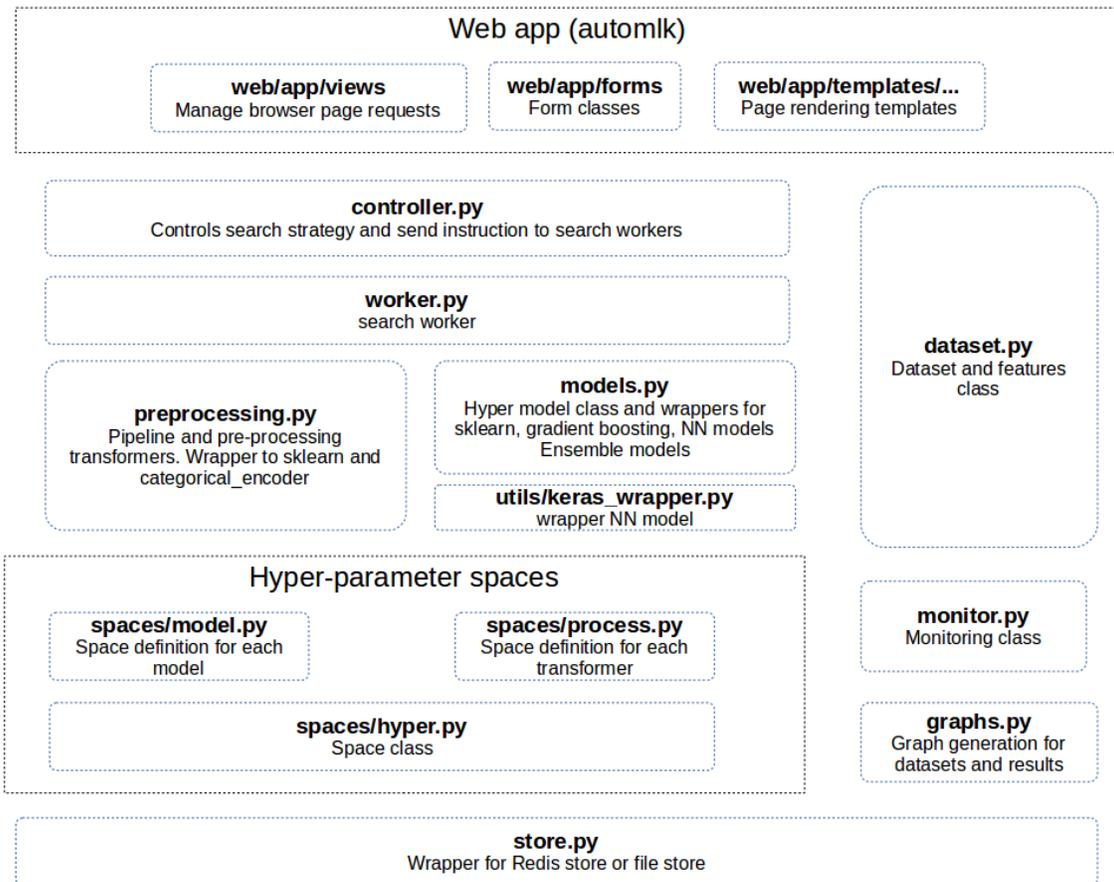


Fig. 1.25: software components of the architecture

We have included a sample of public datasets to start with autoMLk.

To use these datasets, upload the list of datasets or create a dataset in the New dataset from the menu.

the data describing these datasets are located in the csv file 'dataset.csv' in the automlk/datasets folder. You may use the same format to create your own datasets.

1.5 Searching

The automated search will test preprocessing steps and models.

1.6 List of models

The following models are included in autoMLk, with their respective hyper-parameters:

1.6.1 Models level 1

regression:

LightGBM *boosting_type, num_leaves, max_depth, learning_rate, n_estimators, min_split_gain, min_child_weight, min_child_samples, subsample, subsample_freq, colsample_bytree, reg_alpha, reg_lambda, verbose, objective, metric*

XgBoost *max_depth, learning_rate, n_estimators, booster, gamma, min_child_weight, max_delta_step, subsample, colsample_bytree, colsample_bylevel, reg_alpha, reg_lambda, scale_pos_weight, tree_method, sketch_eps, n_jobs, silent, objective, eval_metric*

CatBoost *learning_rate, depth, verbose*

Neural Networks *units, batch_size, batch_normalization, activation, optimizer, learning_rate, number_layers, dropout*

Extra Trees *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, criterion*

Random Forest *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, n_jobs, criterion*

Gradient Boosting *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, learning_rate, loss*

AdaBoost *n_estimators, learning_rate, random_state, loss*

Knn *n_neighbors, weights, algorithm, leaf_size, p, n_jobs*

SVM *C, epsilon, kernel, degree, gamma, coef0, shrinking, tol, max_iter, verbose*

Linear SVR *C, loss, epsilon, dual, tol, fit_intercept, intercept_scaling, max_iter, verbose*

Linear Regression *fit_intercept, normalize, copy_X, n_jobs*

Ridge Regression *alpha, fit_intercept, normalize, copy_X, tol, solver*

Lasso Regression *alpha, fit_intercept, normalize, precompute, copy_X, tol, positive, selection*

Huber Regression *epsilon, alpha, fit_intercept, tol*

classification:

LightGBM *boosting_type, num_leaves, max_depth, learning_rate, n_estimators, min_split_gain, min_child_weight, min_child_samples, subsample, subsample_freq, colsample_bytree, reg_alpha, reg_lambda, verbose, objective, metric*

XgBoost *max_depth, learning_rate, n_estimators, booster, gamma, min_child_weight, max_delta_step, subsample, colsample_bytree, colsample_bylevel, reg_alpha, reg_lambda, scale_pos_weight, tree_method, sketch_eps, n_jobs, silent, objective, eval_metric*

CatBoost *learning_rate, depth, verbose*

Extra Trees *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, n_jobs, criterion, class_weight*

Random Forest *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, n_jobs, criterion, class_weight*

Gradient Boosting *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, learning_rate, criterion, loss*

AdaBoost *n_estimators, learning_rate, random_state, algorithm*

Knn *n_neighbors, weights, algorithm, leaf_size, p, n_jobs*

SVM *C, kernel, degree, gamma, coef0, shrinking, tol, max_iter, verbose, probability*

Logistic Regression *penalty, dual, tol, C, fit_intercept, intercept_scaling, solver, max_iter, multi_class, n_jobs*

Naive Bayes Gaussian ******

Naive Bayes Bernoulli *alpha, binarize, fit_prior*

Neural Networks *units, batch_size, batch_normalization, activation, optimizer, learning_rate, number_layers, dropout*

1.6.2 Ensembles

regression:

Stacking LightGBM *task, boosting, learning_rate, num_leaves, tree_learner, max_depth, min_data_in_leaf, min_sum_hessian_in_leaf, feature_fraction, bagging_fraction, bagging_freq, lambda_l1, lambda_l2, min_gain_to_split, drop_rate, skip_drop, max_drop, uniform_drop, xgboost_dart_mode, top_rate, other_rate, verbose, objective, metric*

Stacking XgBoost *booster, eval_metric, eta, min_child_weight, max_depth, gamma, max_delta_step, sub_sample, colsample_bytree, colsample_bylevel, lambda, alpha, tree_method, sketch_eps, scale_pos_weight, silent, objective*

Stacking Extra Trees *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, criterion*

Stacking Random Forest *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, n_jobs, criterion*

Stacking Gradient Boosting *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, learning_rate, loss*

Stacking Linear Regression *fit_intercept, normalize, copy_X, n_jobs*

classification:

Stacking LightGBM *task, boosting, learning_rate, num_leaves, tree_learner, max_depth, min_data_in_leaf, min_sum_hessian_in_leaf, feature_fraction, bagging_fraction, bagging_freq, lambda_l1, lambda_l2, min_gain_to_split, drop_rate, skip_drop, max_drop, uniform_drop, xgboost_dart_mode, top_rate, other_rate, verbose, objective, metric*

Stacking XgBoost *booster, eval_metric, eta, min_child_weight, max_depth, gamma, max_delta_step, sub_sample, colsample_bytree, colsample_bylevel, lambda, alpha, tree_method, sketch_eps, scale_pos_weight, silent, objective*

Stacking Neural Networks *units, batch_size, batch_normalization, activation, optimizer, learning_rate, number_layers, dropout*

Stacking Extra Trees *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, n_jobs, criterion, class_weight*

Stacking Random Forest *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, n_jobs, criterion, class_weight*

Stacking Gradient Boosting *n_estimators, max_features, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, verbose, random_state, warm_start, learning_rate, criterion, loss*

Stacking Logistic Regression *penalty, dual, tol, C, fit_intercept, intercept_scaling, solver, max_iter, multi_class, n_jobs*

Stacking Neural Networks *units, batch_size, batch_normalization, activation, optimizer, learning_rate, number_layers, dropout*

1.7 Pre-processing steps

The following pre-processing methods are included in autoMLk, with their respective hyper-parameters:

1.7.1 categorical encoding:

No encoding **

Label Encoder **

One hot categorical *drop_invariant*

BaseN categorical *drop_invariant, base*

Hashing categorical *drop_invariant*

1.7.2 text encoding:

Bag of words

Word2Vec

Doc2Vec

1.7.3 imputing missing values:

No missing **

Missing values fixed *fixed*

Missing values frequencies *frequency*

1.7.4 feature scaling:

No scaling **

Scaling Standard **

Scaling MinMax **

Scaling MaxAbs **

Scaling Robust *quantile_range*

1.7.5 feature selection:

No Feature selection **

Truncated SVD *n_components, algorithm*

Fast ICA *n_components, algorithm*

PCA *n_components*

Selection RF *n_estimators*

Selection RF *n_estimators*

Selection LSVR **

CHAPTER 2

Indices

- genindex