# datarun Documentation

*Release 0.1*

Camille

April 11, 2016

# CONTENTS

Contents:

# CHAPTER ONE

# MODELS

**class** `runapp.models.`**`RawData`**(**args*, ***kwargs*)

**Parameters**

- **`name`** (*`string`*) – name of the data set
- **`files_path`** (*`string`*) – path of file where data are saved
- **`workflow_elements`** (*`string`*) – list of workflow elements used to solve the RAMP
- **`column`** (*`target`*) – name of the target column

**class** `runapp.models.`**`Submission`**(**args*, ***kwargs*)

**Parameters**

- **`databoard_s_id`** (*`IntegerField(primary_key=True)`*) – id of the submission in the db of databoard
- **`files_path`** (*`CharField(max_length=200, null=True)`*) – path of submitted files
- **`raw_data`** (*`ForeignKey(RawData, null=True, blank=True)`*) – associated raw data

**class** `runapp.models.`**`SubmissionFold`**(**args*, ***kwargs*)

**Parameters**

- **`databoard_sf_id`** (*`IntegerField(primary_key=True)`*) – id of the submission on cv fold in databoard db
- **`databoard_s`** (*`ForeignKey(Submission, null=True, blank=True)`*) – associated submission
- **`train_is`** (*`TextField`*) – train indices
- **`test_is`** (*`TextField`*) – test indices
- **`priority`** (*`CharField, choices.`*) – priority to train-test the fold ('L' for low priority, 'H' for high priority)
- **`full_train_predictions`** (*`TextField`*) – predictions of the entire train dataset
- **`test_predictions`** (*`TextField`*) – predictions of the test dataset
- **`state`** (*`CharField, choices.`*) – TODO, TRAINED, VALIDATED, TESTED, ERROR
- **`log_messages`** (*`TextField`*) – logs recorded during train and test
- **`train_time`** (*`FloatField, default=0.`*) – real clock training time

- **`validation_time`** (*`FloatField, default=0.`*) – real clock validation time
- **`test_time`** (*`FloatField, default=0.`*) – real clock testing time
- **`train_cpu_time`** (*`FloatField, default=0.`*) – training cpu time
- **`train_memory`** – peak memory usage during train and test (in kb)
- **`test_cpu_time`** – test cpu time
- **`test_memory`** (*`FloatField, default=0.`*) – peak memory usage durnig train and test (in kb)
- **`new`** (*`BooleanField, default=True.`*) – True when it has not already been sent by the API

CHAPTER

TWO

# REQUESTS

You can either make direct requests to the datarun API, or use the post_api function.

## 2.1 Direct requests

**class** `runapp.views.`**`GetTestPredictionList`**(***kwargs*)

Get predictions of submissions on cv fold given their ids

**`post`**(*request*, *format=None*)

Retrieve predictions (on the test data set) of SubmissionFold instances among a list of id that have been trained and tested

- Example with curl (on localhost):

  curl -u username:password -H "Content-Type: application/json" -X POST -d '{"list_submission_fold": [1, 2, 10]}' http://127.0.0.1:8000/runapp/testpredictions/list/

  Don't forget double quotes for the json, simple quotes do not work

- Example with the python package requests (on localhost):

  requests.post('http://127.0.0.1:8000/runapp/testpredictions/list/', auth=('username', 'password'), json={'list_submission_fold': [1, 2, 10]})

— parameters:

- name: list_submission_fold description: list of submission on cv fold ids required: true type: list paramType: form

response_serializer: TestPredSubmissionFoldSerializer

**class** `runapp.views.`**`GetTestPredictionNew`**(***kwargs*)

Get predictions of submissions on cv fold that have not been requested

**`post`**(*request*, *format=None*)

Retrieve predictions (on the test data set) of SubmissionFold instances that have been trained and tested and not yet requested. You can specify a given data challenge by posting the raw_data id.

- Example with curl (on localhost):

  curl -u username:password -H "Content-Type: application/json" -X POST -d '{"raw_data_id": 1}' http://127.0.0.1:8000/runapp/testpredictions/new/

  Don't forget double quotes for the json, simple quotes do not work

- Example with the python package requests (on localhost):

requests.post('http://127.0.0.1:8000/runapp/testpredictions/new/', auth=('username', 'password'), json={'raw_data_id': 1})

— parameters:

- name: raw_data_id description: id of the raw dataset from which to get predictions required: false type: integer paramType: form

response_serializer: TestPredSubmissionFoldSerializer

**class** `runapp.views.`**`RawDataList`**`(`***kwargs*`)`

List all data set or submit a new one

**`get`**`(`*request*, *format=None*`)`

List all raw dataset

- Example with curl (on localhost):

curl -u username:password GET http://127.0.0.1:8000/runapp/rawdata/

- Example with the python package requests (on localhost):

requests.get('http://127.0.0.1:8000/runapp/rawdata/', auth=('username', 'password'))

— response_serializer: RawDataSerializer

**`post`**`(`*request*, *format=None*`)`

Create a new dataset

- Example with curl (on localhost):

  curl -u username:password -H "Content-Type: application/json" -X POST -d '{"name": "iris", "target_column": "species", "workflow_elements": "classifier", "files": {"iris.csv": 'blablabla'}}' http://127.0.0.1:8000/runapp/rawdata/

  Don't forget double quotes for the json, simple quotes don't work.

- Example with the python package requests (on localhost):

  requests.post('http://127.0.0.1:8000/runapp/rawdata/', auth=('username', 'password'), json={'name': 'iris', 'target_column': 'species', 'workflow_elements': 'classifier', 'files': {'iris.csv': 'blablabla'}})

— request_serializer: RawDataSerializer response_serializer: RawDataSerializer

**class** `runapp.views.`**`SplitTrainTest`**`(`***kwargs*`)`

Split data set into train and test datasets

**`post`**`(`*request*, *format=None*`)`

Split raw data into train and test datasets

- Example with curl (on localhost):

  **curl -u username:password -H "Content-Type: application/json" -X POST -d '{"random_state": 42, "held_**
  http://127.0.0.1:8000/runapp/rawdata/split/

  Don't forget double quotes for the json, simple quotes do not work

- Example with the python package requests (on localhost):

  requests.post('http://127.0.0.1:8000/runapp/raw_data/split/', auth=('username', 'password'), json={'random_state': 42, 'held_out_test': 0.7, 'raw_data_id': 1})

— parameters:

- name: random_state description: random state used to split data required: false type: integer paramType: form

- •name: held_out_test description: percentage of the dataset kept as test dataset required: true type: float paramType: form
- •name: raw_data_id description: id of the raw dataset required: true type: integer paramType: form

**class** `runapp.views.`**`SubmissionFoldDetail`**(***kwargs*)

Get a submission on CV fold given its id

**`get`**(*request*, *pk*, *format=None*)

Retrieve a SubmissionFold instance to check its state

- •Example with curl (on localhost):

  curl -u username:password GET http://127.0.0.1:8000/runapp/submissionfold/10/

- •Example with the python package requests (on localhost):

  requests.get('http://127.0.0.1:8000/runapp/submissionfold/10/', auth=('username', 'password'))

— parameters:

- •name : pk description: id of the submission on cv fold in the databoard db required: true type: interger paramType: path

response_serializer: SubmissionFoldSerializer

**class** `runapp.views.`**`SubmissionFoldLightList`**(***kwargs*)

To get main info about all submissions on CV fold

**`get`**(*request*, *format=None*)

List main info (id, submission id, state, new) about all submissions on CV fold

- •Example with curl (on localhost):

  curl -u username:password GET http://127.0.0.1:8000/runapp/submissionfold-light/

- •Example with the python package requests (on localhost):

  requests.get('http://127.0.0.1:8000/runapp/submissionfold-light/', auth=('username', 'password'))

— response_serializer: SubmissionFoldLightSerializer

**class** `runapp.views.`**`SubmissionFoldList`**(***kwargs*)

To get all submissions on CV fold

**`get`**(*request*, *format=None*)

List all submission on CV fold

- •Example with curl (on localhost):

  curl -u username:password GET http://127.0.0.1:8000/runapp/submissionfold/

- •Example with the python package requests (on localhost):

  requests.get('http://127.0.0.1:8000/runapp/submissionfold/', auth=('username', 'password'))

— response_serializer: SubmissionFoldSerializer

**`post`**(*request*, *format=None*)

Create a submission on CV fold (and if necessary the associated submission

- •Example with curl (on localhost):

curl -u username:password -H "Content-Type: application/json" -X POST -d '{"databoard_s_id": 1, "files": {"classifier.py": "import sklearn.."}, "train_is": "hgjhg", "raw_data":1, "databoard_sf_id": 11, "test_is": "kdjhLGf2", "priority": "L"}' http://127.0.0.1:8000/runapp/submissionfold/

Don't forget double quotes for the json, simple quotes do not work

•Example with the python package requests (on localhost):

requests.post('http://127.0.0.1:8000/runapp/submissionfold/', auth=('username', 'password'), json={'databoard_sf_id': 10, 'databoard_s_id': 24, 'raw_data': 8, 'train_is': 'GDHRFdfgfd', 'test_is': 'kdjhLGf2', 'priority': 'L' 'files': {'classifier.py': 'import skle...'}})

— request_serializer: SubmissionFoldSerializer response_serializer: SubmissionFoldSerializer

`runapp.views.`**`save_files`**(*dir_data*, *data*)
save files from data['files'] in directory dir_data

## 2.2 post_api module

`test_files.post_api.`**`get_prediction_list`**(*host_url*, *username*, *password*, *list_submission_fold_id*)
Get predictions given a list of submission on cv fold ids

**Parameters**

- **`host_url`** (*`string`*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **`username`** (*`string`*) – username to be used for authentication
- **`password`** (*`string`*) – password to be used for authentication
- **`list_submission_fold_id`** (*`list`*) – list of submission on cv fold ids from which we want the predictions

`test_files.post_api.`**`get_prediction_new`**(*host_url*, *username*, *password*, *raw_data_id*)
Get all new predictions given a raw data id

**Parameters**

- **`host_url`** (*`string`*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **`username`** (*`string`*) – username to be used for authentication
- **`password`** (*`string`*) – password to be used for authentication
- **`raw_data_id`** (*`integer`*) – id of a data set from which we want new predictions

`test_files.post_api.`**`get_raw_data`**(*host_url*, *username*, *password*)
Get all raw data sets

**Parameters**

- **`host_url`** (*`string`*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **`username`** (*`string`*) – username to be used for authentication
- **`password`** (*`string`*) – password to be used for authentication

`test_files.post_api.`**`get_submission_fold`**(*host_url*, *username*, *password*)
Get all submission on cv fold (all attributes)

**Parameters**

- **host_url** (*string*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **username** (*string*) – username to be used for authentication
- **password** (*string*) – password to be used for authentication

test_files.post_api.**get_submission_fold_detail**(*host_url*, *username*, *password*, *submission_fold_id*)
Get details about a submission on cv fold given its id

**Parameters**

- **host_url** (*string*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **username** (*string*) – username to be used for authentication
- **password** (*string*) – password to be used for authentication
- **submission_fold_id** – id of the submission on cv fold
- **submission_fold_id** – integer

test_files.post_api.**get_submission_fold_light**(*host_url*, *username*, *password*)
Get all submissions on cv fold only main info: id, associated submission id, state, and new

**Parameters**

- **host_url** (*string*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **username** (*string*) – username to be used for authentication
- **password** (*string*) – password to be used for authentication

test_files.post_api.**post_data**(*host_url*, *username*, *password*, *data_name*, *target_column*, *workflow_elements*, *data_file*)
To post data to the datarun api. Data are compressed (with zlib) and base64-encoded before being posted.

**Parameters**

- **host_url** (*string*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **username** (*string*) – username to be used for authentication
- **password** (*string*) – password to be used for authentication
- **data_name** (*string*) – name of the raw dataset
- **target_column** (*string*) – name of the target column
- **workflow_elements** (*string*) – workflow elements associated with this dataset, e.g., feature_extractor, classifier
- **data_file** (*string*) – name with absolute of the dataset file

test_files.post_api.**post_submission_fold**(*host_url*, *username*, *password*, *sub_id*, *sub_fold_id*, *train_is*, *test_is*, *priority='L'*, *raw_data_id=None*, *list_submission_files=None*)
To post submission on cv fold and submission (if not already posted). Submission files are compressed (with zlib) and base64-encoded before being posted.

**Parameters**

- **host_url** (*string*) – api host url, such as http://127.0.0.1:8000/ (localhost)
- **username** (*string*) – username to be used for authentication
- **password** (*string*) – password to be used for authentication

- **sub_id** (*integer*) – id of the submission on databoard
- **sub_fold_id** (*integer*) – id of the submission on cv fold on databoard
- **train_is** (*numpy array*) – train indices for the cv fold
- **test_is** (*numpy array*) – test indices for the cv fold
- **priority** (*string*) – priority level to train test the model: L for low and H for high

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX