bigml-java Documentation

Release stable

Mar 26, 2017

Contents

BigML makes machine learning easy by taking care of the details required to add data-driven decisions and predictive power to your company. Unlike other machine learning services, BigML creates beautiful predictive models that can be easily understood and interacted with.

These BigML Java bindings allow you to interact with BigML.io, the API for BigML. You can use it to easily create, retrieve, list, update, and delete BigML resources (i.e., sources, datasets, models and, predictions).

This module is licensed under the Apache License, Version 2.0.

CHAPTER 1

Support

Please report problems and bugs to our BigML Java Binding issue tracker.

CHAPTER 2

Requirements

JVM 1.6 and above are currently supported by these bindings.

You will also need `maven` to build the package. If you are new to `maven`, please refer to Maven Getting Started Guide.

CHAPTER 3

Installation

To use the latest stable release, include the following `maven` dependency in your project's `pom.xml`.

```
<dependency>
    <groupId>org.bigml</groupId>
    <artifactId>bigml-binding</artifactId>
    <version>1.4.2</version>
</dependency>
```

You can also download the development version of the bindings directly from the Git repository

```
$ git clone git://github.com/bigmlcom/bigml-java.git
```

CHAPTER 4

Authentication

All the requests to BigML.io must be authenticated using your username and API key and are always transmitted over HTTPS.

This module will look for your username and API key in the `src/main/resources/binding.properties` file. Alternatively, you can respectively set the JVM parameters `BIGML_USERNAME` and `BIGML_API_KEY` with `-D`.

With that set up, connecting to BigML is a breeze. First, import `BigMLClient`:

```
import org.bigml.binding.BigMLClient;
```

then:

```
BigMLClient api = BigMLClient.getInstance();
```

Otherwise, you can initialize directly when instantiating the BigMLClient class as follows:

```
BigMLClient api = BigMLClient.getInstance("myusername",
    "ae579e7e53fb9abd646a6ff8aa99d4afe83ac291", false);
```

Note that the 3rd parameter `devMode` determines where your resources will be created, either production or development. You can initialize the library to work in the Sandbox environment by passing `true` for this parameter:

```
BigMLClient api = BigMLClient.getInstance(true);
```

or:

```
BigMLClient api = BigMLClient.getInstance("myusername",
    "ae579e7e53fb9abd646a6ff8aa99d4afe83ac291", true);
```

WARNING: `getInstance` also takes 2-String parameters, and they are NOT your username and API key, but `seed` and `storage`.

For Virtual Private Cloud setups, you can change the remote server URL to the VPC particular one by either setting the `BIGML_URL` or `BIGML_DEV_URL` in `binding.properties` or in the JVM environment. By default, they have the following values:

```
BIGML_URL=https://bigml.io/andromeda/
BIGML_DEV_URL=https://bigml.io/dev/andromeda/
```

If you are in Australia or New Zealand, you can change them to:

```
BIGML_URL=https://au.bigml.io/andromeda/
BIGML_DEV_URL=https://au.bigml.io/dev/andromeda/
```

The corresponding SSL REST calls will be directed to your private domain henceforth.

CHAPTER 5

Quick Start

Imagine that you want to use this csv file containing the Iris flower dataset to predict the species of a flower whose `sepal length` is `5` and whose `sepal width` is `2.5`. A preview of the dataset is shown below. It has 4 numeric fields: `sepal length`, `sepal width`, `petal length`, `petal width` and a categorical field: `species`. By default, BigML considers the last field in the dataset as the objective field (i.e., the field that you want to generate predictions for).

```
sepal length,sepal width,petal length,petal width,species
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
...
5.8,2.7,3.9,1.2,Iris-versicolor
6.0,2.7,5.1,1.6,Iris-versicolor
5.4,3.0,4.5,1.5,Iris-versicolor
...
6.8,3.0,5.5,2.1,Iris-virginica
5.7,2.5,5.0,2.0,Iris-virginica
5.8,2.8,5.1,2.4,Iris-virginica
```

You can easily generate a prediction following these steps:

```
// Create BigMLClient with the properties in binding.properties
BigMLClient api = BigMLClient.getInstance();

JSONObject args = null;

JSONObject source = api.createSource("./data/iris.csv",
    "Iris Source", args);

while (!api.sourceIsReady(source)) Thread.sleep(1000);

JSONObject dataset = api.createDataset(
    (String)source.get("resource"), args, null, null);

while (!api.datasetIsReady(dataset)) Thread.sleep(1000);
```

```
JSONObject model = api.createModel(
    (String)dataset.get("resource"), args, null, null);

while (!api.modelIsReady(model)) Thread.sleep(1000);

JSONObject inputData = new JSONObject();
inputData.put("sepal length", 5);
inputData.put("sepal width", 2.5);

JSONObject prediction = api.createPrediction(
    (String)model.get("resource"), inputData, true,
    args, null, null);
```

You can then get the prediction result:

```
prediction = api.getPrediction(prediction);
```

and print the result:

```
String output = (String)Utils.getJSONObject(
    prediction, "object.output");
System.out.println("Prediction result: " + output);
```

```
Prediction result: Iris-virginica
```

and also generate an evaluation for the model by using:

```
JSONObject testSource = api.createSource("./data/test_iris.csv",
    "Test Iris Source", args);

while (!api.sourceIsReady(source)) Thread.sleep(1000);

JSONObject testDataset = api.createDataset(
    (String)testSource.get("resource"), args, null, null);

while (!api.datasetIsReady(dataset)) Thread.sleep(1000);

JSONObject evaluation = api.createEvaluation(
    (String)model.get("resource"), (String)dataset.get("resource"),
    args, null, null);
```

Setting the `storage` argument in the api client instantiation:

```
BigMLClient api = BigMLClient.getInstance("./storage");
```

or:

```
BigMLClient api = BigMLClient.getInstance("myusername",
    "ae579e7e53fb9abd646a6ff8aa99d4afe83ac291", true, "./storage");
```

all the generated, updated or retrieved resources will be automatically saved to the chosen directory.

You can also find a sample API client code from here.

CHAPTER 6

Fields

BigML automatically generates identifiers for each field. The following example shows how to retrieve the fields, ids, and its types that have been assigned to a source:

```
source = api.getSource(source);
JSONObject fields = (JSONObject) Utils.getJSONObject(source, "object.fields");
```

source `fields` object:


```
{
    "000000":{
        "name":"sepal length",
        "column_number":0,
        "optype":"numeric",
        "order":0
    },
    "000001":{
        "name":"sepal width",
        "column_number":1,
        "optype":"numeric",
        "order":1
    },
    "000002":{
        "name":"petal length",
        "column_number":2,
        "optype":"numeric",
        "order":2
    },
    "000003":{
        "name":"petal width",
        "column_number":3,
        "optype":"numeric",
        "order":3
    },
    "000004":{
        "column_number":4,
```

```
        "name":"species",
        "optype":"categorical",
        "order":4,
        "term_analysis":{
            "enabled":true
        }
    }
}
```

CHAPTER 7

Dataset

If you want to get some basic statistics for each field you can retrieve the `fields` from the dataset as follows to get a dictionary keyed by field id:

```
dataset = api.getDataset(dataset);
JSONOoject fields = (JSONObject) Utils.getJSONObject(dataset, "object.fields");
```

dataset `fields` object:

```
{
    "000000": {
        "column_number": 0,
        "datatype": "double",
        "name": "sepal length",
        "optype": "numeric",
        "order": 0,
        "preferred": true,
        "summary": {
            "bins": [
                [4.3, 1],
                [4.425, 4],

                ...snip...

                [7.9, 1]
            ],
            "kurtosis": -0.57357,
            "maximum": 7.9,
            "mean": 5.84333,
            "median": 5.8,
            "minimum": 4.3,
            "missing_count": 0,
            "population": 150,
            "skewness": 0.31175,
            "splits": [
                4.51526,
```

```
                4.67252,

                ...snip...

                7.64746
            ],
            "standard_deviation": 0.82807,
            "sum": 876.5,
            "sum_squares": 5223.85,
            "variance": 0.68569
        }
    },

    ...snip...

    "000004": {

        ...snip...

    }
}
```

CHAPTER 8

Model

One of the greatest things about BigML is that the models that it generates for you are fully white-boxed. To get the explicit tree-like predictive model for the example above:

```
model = api.getModel(model);
JSONObject tree = (JSONObject) Utils.getJSONObject(model, "object.model.root");
```

model `tree` object:

```
{
    "children":[{
        "children":[{
            "children":[{
                "confidence":0.91799,
                "count":43,
                "id":3,
                "objective_summary":{
                    "categories":[
                        [
                            "Iris-virginica",
                            43
                        ]
                    ]
                },
                "output":"Iris-virginica",
                "predicate":{
                    "field":"000002",
                    "operator":">",
                    "value":4.85
                }
            }, {
                "children":[{
                    "confidence":0.20654,
                    "count":1,
                    "id":5,
                    "objective_summary":{
```

```
                        "categories":[
                            [
                                "Iris-versicolor",
                                1
                            ]
                        ]
                    },
                    "output":"Iris-versicolor",
                    "predicate":{
                        "field":"000001",
                        "operator":">",
                        "value":3.1
                    }
                },

                ...snip...

            },

            ...snip...

        },

        ...snip...

    },

    ...snip...
}
```

(Note that we have abbreviated the output in the snippet above for readability: the full predictive model you'll get is going to contain much more details).

CHAPTER 9

Evaluation

The predictive performance of a model can be measured using many different measures. In BigML these measures can be obtained by creating evaluations. To create an evaluation you need the id of the model you are evaluating and the id of the dataset that contains the data to be tested with. The result is shown as:

```
evaluation = api.getEvaluation(evaluation);
JSONObject result = (JSONObject) Utils.getJSONObject(evaluation, "object.result");
```

evaluation `result` object:


```
{
    "class_names":[
        "Iris-setosa",
        "Iris-versicolor",
        "Iris-virginica"
    ],
    "mode":{
        "accuracy":0.33333,
        "average_f_measure":0.16667,
        "average_phi":0,
        "average_precision":0.11111,
        "average_recall":0.33333,
        "confusion_matrix":[
            [50, 0, 0],
            [50, 0, 0],
            [50, 0, 0]
        ],
        "per_class_statistics":[
            {
                "accuracy":0.3333333333333333,
                "class_name":"Iris-setosa",
                "f_measure":0.5,
                "phi_coefficient":0,
                "precision":0.3333333333333333,
                "present_in_test_data":true,
                "recall":1.0
```

```
            },
            {
                "accuracy":0.6666666666666667,
                "class_name":"Iris-versicolor",
                "f_measure":0,
                "phi_coefficient":0,
                "precision":0,
                "present_in_test_data":true,
                "recall":0.0
            },
            {
                "accuracy":0.6666666666666667,
                "class_name":"Iris-virginica",
                "f_measure":0,
                "phi_coefficient":0,
                "precision":0,
                "present_in_test_data":true,
                "recall":0.0
            }
        ]
    },
    "model":{
        "accuracy":1,
        "average_f_measure":1,
        "average_phi":1,
        "average_precision":1,
        "average_recall":1,
        "confusion_matrix":[
            [50, 0, 0],
            [0, 50, 0],
            [0, 0, 50]
        ],
        "per_class_statistics":[
            {
                "accuracy":1.0,
                "class_name":"Iris-setosa",
                "f_measure":1.0,
                "phi_coefficient":1.0,
                "precision":1.0,
                "present_in_test_data":true,
                "recall":1.0
            },
            {
                "accuracy":1.0,
                "class_name":"Iris-versicolor",
                "f_measure":1.0,
                "phi_coefficient":1.0,
                "precision":1.0,
                "present_in_test_data":true,
                "recall":1.0
            },
            {
                "accuracy":1.0,
                "class_name":"Iris-virginica",
                "f_measure":1.0,
                "phi_coefficient":1.0,
                "precision":1.0,
                "present_in_test_data":true,
```

```
                "recall":1.0
            }
        ]
    },
    "random":{
        "accuracy":0.28,
        "average_f_measure":0.27789,
        "average_phi":-0.08123,
        "average_precision":0.27683,
        "average_recall":0.28,
        "confusion_matrix":[
            [14, 19, 17],
            [19, 10, 21],
            [15, 17, 18]
        ],
        "per_class_statistics":[
            {
                "accuracy":0.5333333333333333,
                "class_name":"Iris-setosa",
                "f_measure":0.2857142857142857,
                "phi_coefficient":-0.06063390625908324,
                "precision":0.2916666666666667,
                "present_in_test_data":true,
                "recall":0.28
            },
            {
                "accuracy":0.4933333333333333,
                "class_name":"Iris-versicolor",
                "f_measure":0.20833333333333331,
                "phi_coefficient":-0.16357216402190614,
                "precision":0.21739130434782608,
                "present_in_test_data":true,
                "recall":0.2
            },
            {
                "accuracy":0.5333333333333333,
                "class_name":"Iris-virginica",
                "f_measure":0.33962264150943394,
                "phi_coefficient":-0.019492029389636262,
                "precision":0.32142857142857145,
                "present_in_test_data":true,
                "recall":0.36
            }
        ]
    }
}
```


where two levels of detail are easily identified. For classifications, the first level shows these keys:

- **class_names**: A list with the names of all the categories for the objective field (i.e., all the classes)
- **mode**: A detailed result object. Measures of the performance of the classifier that predicts the mode class for all the instances in the dataset
- **model**: A detailed result object.
- **random**: A detailed result object. Measures the performance of the classifier that predicts a random class for all the instances in the dataset.

and the detailed result objects include `accuracy`, `average_f_measure`, `average_phi`,

`average_precision`, `average_recall`, `confusion_matrix` and `per_class_statistics`.

For regressions first level will contain these keys:

- **mean**: A detailed result object. Measures the performance of the model that predicts the mean for all the instances in the dataset.
- **model**: A detailed result object.
- **random**: A detailed result object. Measures the performance of the model that predicts a random class for all the instances in the dataset.

where the detailed result objects include `mean_absolute_error`, `mean_squared_error` and `r_squared` (refer to developers documentation for more info on the meaning of these measures.

CHAPTER 10

Cluster

For unsupervised learning problems, the cluster is used to classify in a limited number of groups your training data. The cluster structure is defined by the centers of each group of data, named centroids, and the data enclosed in the group. As for in the model's case, the cluster is a white-box resource and can be retrieved as a JSON:

```
cluster = api.getCluster(cluster);
JSONObject result = (JSONObject) Utils.getJSONObject(cluster, "object");
```

cluster `object` object:

```
{
    "balance_fields":true,
    "category":0,
    "cluster_datasets":{},
    "cluster_models":{},
    "clusters":{
        "clusters":[{
            "center":{
                "000000":6.262,
                "000001":2.872,
                "000002":4.906,
                "000003":1.676,
                "000004":"Iris-virginica"
            },
            "count":100,
            "distance":{
                "bins":[
                    [0.03935, 1],
                    [0.04828, 1],
                    [0.06093, 1 ],

                    ...snip...

                    [0.47935, 1]
                ],
                "maximum":0.47935,
```

```
                "mean":0.21705,
                "median":0.20954,
                "minimum":0.03935,
                "population":100,
                "standard_deviation":0.0886,
                "sum":21.70515,
                "sum_squares":5.48833,
                "variance":0.00785
            },
            "id":"000000",
            "name":"Cluster 0"
        }, {
            "center":{
                "000000":5.006,
                "000001":3.428,
                "000002":1.462,
                "000003":0.246,
                "000004":"Iris-setosa"
            },
            "count":50,
            "distance":{
                "bins":[
                    [0.01427, 1],
                    [0.02279, 1],

                    ...snip...

                    [0.41736, 1]
                ],
                "maximum":0.41736,
                "mean":0.12717,
                "median":0.113,
                "minimum":0.01427,
                "population":50,
                "standard_deviation":0.08521,
                "sum":6.3584,
                "sum_squares":1.16432,
                "variance":0.00726
            },
            "id":"000001",
            "name":"Cluster 1"
        }],
        "fields":{

            ...snip...

        }
    },
    "code":200,
    "columns":5,
    "created":"2016-02-17T08:26:12.583000",
    "credits":0.017581939697265625,
    "credits_per_prediction":0.0,
    "critical_value":5,
    "dataset":"dataset/56c42ea07e0a8d6cca01519b",
    "dataset_field_types":{
        "categorical":1,
        "datetime":0,
```

```
        "effective_fields":5,
        "items":0,
        "numeric":4,
        "preferred":5,
        "text":0,
        "total":5
    },
    "dataset_status":true,
    "dataset_type":0,
    "description":"",
    "dev":true,
    "excluded_fields":[],
    "field_scales":{},
    "fields_meta":{
        "count":5,
        "limit":1000,
        "offset":0,
        "query_total":5,
        "total":5
    },
    "input_fields":[
        "000000",
        "000001",
        "000002",
        "000003",
        "000004"
    ],
    "k":2,
    "locale":"en_US",
    "max_columns":5,
    "max_rows":150,
    "model_clusters":false,
    "name":"Iris Source dataset's cluster",
    "number_of_batchcentroids":0,
    "number_of_centroids":0,
    "number_of_public_centroids":0,
    "out_of_bag":false,
    "price":0.0,
    "private":true,
    "project":null,
    "range":[
        1,
        150
    ],
    "replacement":false,
    "resource":"cluster/56c42ea47e0a8d6cca0151a0",
    "rows":150,
    "sample_rate":1.0,
    "scales":{
        "000000":0.18941532079904913,
        "000001":0.35975000221609077,
        "000002":0.08884141152890178,
        "000003":0.20571391803576422,
        "000004":0.15627934742019414
    },
    "shared":false,
    "size":4609,
    "source":"source/56c42e9f8a318f66df007548",
```

```
    "source_status":true,
    "status":{
        "code":5,
        "elapsed":1213,
        "message":"The cluster has been created",
        "progress":1.0
    },
    "subscription":false,
    "summary_fields":[],
    "tags":[],
    "updated":"2016-02-17T08:26:24.259000",
    "white_box":false
}
```


(Note that we have abbreviated the output in the snippet above for readability: the full predictive cluster you'll get is going to contain much more details).

CHAPTER 11

Anomaly Detector

For anomaly detection problems, BigML uses iforest as an unsupervised kind of model that detects anomalous data in a dataset. The information it returns encloses a `top_anomalies` block that contains a list of the most anomalous points. For each, we capture a `score` from 0 to 1. The closer to 1, the more anomalous. We also capture the `row` which gives values for each field in the order defined by `input_fields`. Similarly we give a list of `importances` which match the `row` values. These importances tell us which values contributed most to the anomaly score. Thus, the structure of an anomaly detector is similar to:

```
anomaly = api.getAnomaly(anomaly);
JSONObject object = (JSONObject) Utils.getJSONObject(anomaly, "object");
```

anomaly `object` object:


```
{
    "anomaly_seed":"2c249dda00fbf54ab4cdd850532a584f286af5b6",
    "category":0,
    "code":200,
    "columns":5,
    "constraints":false,
    "created":"2016-02-17T08:42:26.663000",
    "credits":0.12307357788085938,
    "credits_per_prediction":0.0,
    "dataset":"dataset/56c432657e0a8d6cd0004a2d",
    "dataset_field_types":{
        "categorical":1,
        "datetime":0,
        "effective_fields":5,
        "items":0,
        "numeric":4,
        "preferred":5,
        "text":0,
        "total":5
    },
    "dataset_status":true,
    "dataset_type":0,
```

```
    "description":"",
    "dev":true,
    "excluded_fields":[],
    "fields_meta":{
        "count":5,
        "limit":1000,
        "offset":0,
        "query_total":5,
        "total":5
    },
    "forest_size":128,
    "id_fields":[],
    "input_fields":[
        "000000",
        "000001",
        "000002",
        "000003",
        "000004"
    ],
    "locale":"en_US",
    "max_columns":5,
    "max_rows":150,
    "model":{
        "constraints":false,
        "fields":{

            ...snip...

        },
        "forest_size":128,
        "kind":"iforest",
        "mean_depth":9.557347074468085,
        "sample_size":94,
        "top_anomalies":[{
            "importance":[
                0.22808,
                0.23051,
                0.21026,
                0.1756,
                0.15555
            ],
            "row":[
                7.9,
                3.8,
                6.4,
                2.0,
                "Iris-virginica"
            ],
            "row_number":131,
            "score":0.58766
        },
        {
            "importance":[
                0.21552,
                0.22631,
                0.22319,
                0.1826,
                0.15239
```

```
            ],
            "row":[
                7.7,
                3.8,
                6.7,
                2.2,
                "Iris-virginica"
            ],
            "row_number":117,
            "score":0.58458
        },

        ...snip...


        {
            "importance":[
                0.23113,
                0.15013,
                0.17312,
                0.20304,
                0.24257
            ],
            "row":[
                4.9,
                2.5,
                4.5,
                1.7,
                "Iris-virginica"
            ],
            "row_number":106,
            "score":0.54096
        }],
        "top_n":10,
        "trees":[{
            "root":{
                "children":[{
                    "children":[{
                        "children":[{
                            "children":[{
                                "children":[{
                                    "population":1,
                                    "predicates":[{
                                        "field":"00001f",
                                        "op":">",
                                        "value":35.54357
                                    }]
                                }, {

                                ...snip...

                                }, {
                                    "population":1,
                                    "predicates":[{
                                        "field":"00001f",
                                        "op":"<=",
                                        "value":35.54357
                                    }]
```

```
                                }],
                                "population":2,
                                "predicates":[{
                                    "field":"000005",
                                    "op":"<=",
                                    "value":1385.5166
                                }]
                            }],
                            "population":3,
                            "predicates":[{
                                "field":"000020",
                                "op":"<=",
                                "value":65.14308
                            }, {
                                "field":"000019",
                                "op":"=",
                                "value":0
                            }]
                        }],

                        ...snip...

                        "population":105,
                        "predicates":[{
                            "field":"000017",
                            "op":"<=",
                            "value":13.21754
                        }, {
                            "field":"000009",
                            "op":"in",
                            "value":["0"]
                        }]
                    }],
                    "population":126,
                    "predicates":[true, {
                        "field":"000018",
                        "op":"=",
                        "value":0
                    }]
                },
            },
            "training_mean_depth":11.071428571428571
        }
    },
    "name":"Iris Source dataset's anomaly detector",
    "number_of_anomalyscores":0,
    "number_of_batchanomalyscores":0,
    "number_of_public_anomalyscores":0,
    "ordering":0,
    "out_of_bag":false,
    "price":0.0,
    "private":true,
    "project":null,
    "range":[
        1,
        150
    ],
    "replacement":false,
```

```
    "resource":"anomaly/56c432728a318f66e4012f82",
    "rows":150,
    "sample_rate":1.0,
    "sample_size":94,
    "shared":false,
    "size":4609,
    "source":"source/56c432638a318f66e4012f7b",
    "source_status":true,
    "status":{
        "code":5,
        "elapsed":617,
        "message":"The anomaly detector has been created",
        "progress":1.0
    },
    "subscription":false,
    "tags":[],
    "top_n":10,
    "updated":"2016-02-17T08:42:42.238000",
    "white_box":false
}
```


(Note that we have abbreviated the output in the snippet above for readability: the full anomaly detector you'll get is going to contain much more details).

The `trees` list contains the actual isolation forest, and it can be quite large usually. That's why, this part of the resource should only be included in downloads when needed. Each node in an isolation tree can have multiple predicates. For the node to be a valid branch when evaluated with a data point, all of its predicates must be true.

CHAPTER 12

Additional Information

For additional information about the API, see the BigML developer's documentation.