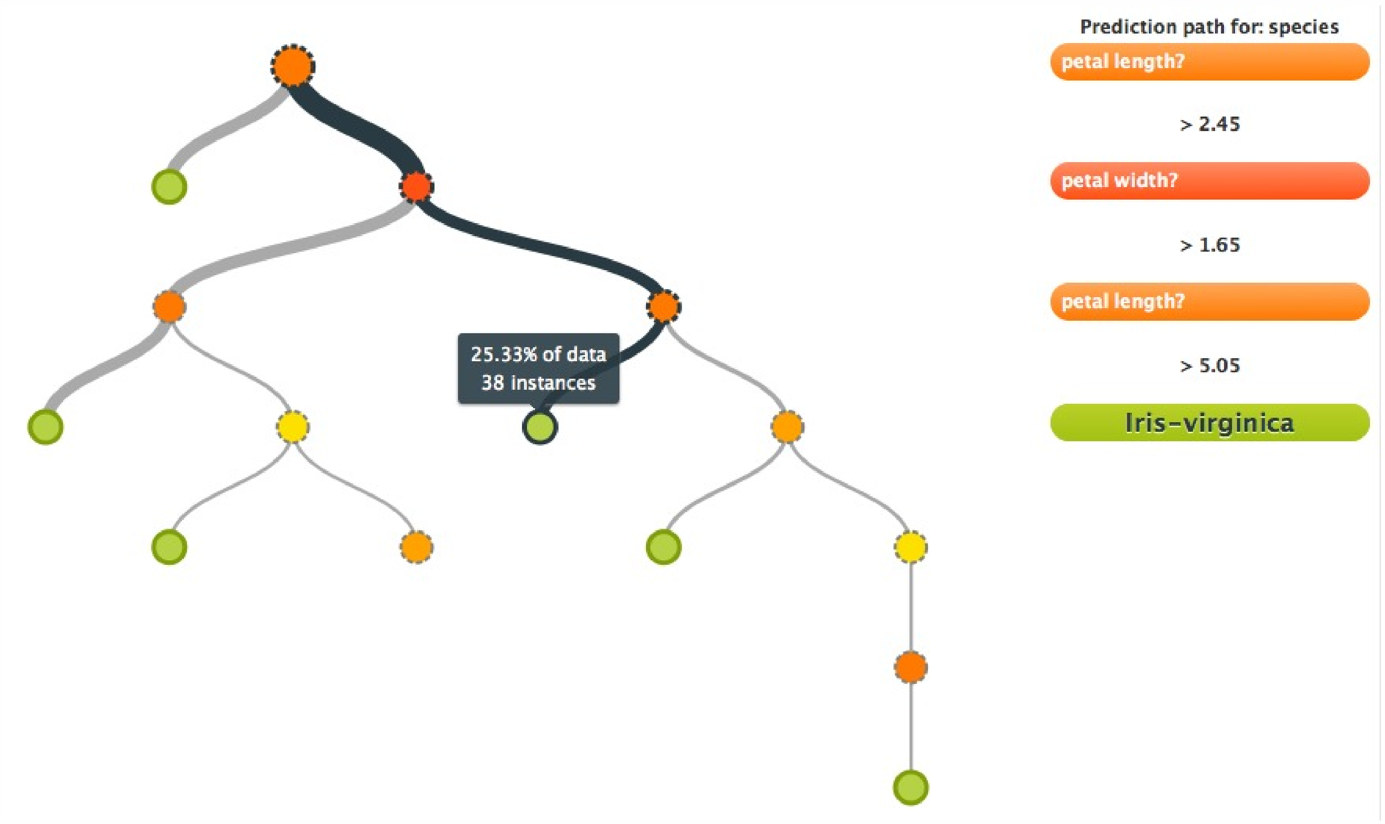
Models

A **model** is a tree-like representation of your **dataset** with predictive power. You can create a **model** selecting which fields from your **dataset** you want to use as **input fields** (or predictors) and which field you want to predict, the **objective field**.

Each node in the **model** corresponds to one of the **input fields**. Each node has an incoming branch except the top node also known as **root** that has none. Each node has a number of outgoing branches except those at the bottom (the "leaves") that have none.

Each branch represents a possible value for the input field where it originates. A leaf represents the value of the **objective field** given all the values for each **input field** in the chain of branches that goes from the root to that leaf.

When you create a new model, **BigML.io** will automatically compute a classification model or regression model depending on whether the **objective field** that you want to predict is categorical or numeric, respectively.



BigML.io allows you to create, retrieve, update, and delete your **model**. You can also list all of your **models**.

Jump to:

Model Base URL

You can use the following base URL to create, retrieve, update, and delete **models**.

```bash
https://bigml.io/andromeda/model
```

All requests to manage your **models** must use HTTPS and be authenticated using your **username** and **API key** to verify your identity. See this section for more details.

Creating a Model

To create a new **model**, you need to POST to the **model** base URL an object containing at least the **dataset/id** that you want to use to create the **model**. The **content-type** must always be **"application/json"**.

```
POST /model?$BIGML_AUTH HTTP/1.1
Host: bigml.io
Content-Type: application/json
```

You can easily create a new **model** using **curl** as follows. All you need is a valid **dataset/id** and your authentication variable set up as shown above.

```curl
curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
    -X POST \
    -H 'content-type: application/json' \
    -d '{"dataset": "dataset/603e20a91f386f43db000004"}'
```

BigML.io will return a newly created **model** document, if the request succeeded.

```json
{
    "boosted_ensemble": false,
    "boosting": {},
    "category": 0,
    "cluster": null,
    "cluster_status": false,
    "code": 201,
    "columns": 5,
    "configuration": null,
    "configuration_status": false,
    "created": "2021-03-04T09:07:17.111162",
    "creator": "alfred",
    "dataset": "dataset/603e20a91f386f43db000004",
    "dataset_field_types": {
```

Model Arguments

In addition to the **dataset**, you can also POST the following arguments.

Argument	Type	Description
		Setting this parameter to **true** will set the

Argument	Type	Description
all_fields_preferred optional	Boolean, default is **false**	preferred flag of all fields at once, instead of doing it one by one in a **fields** map. **Example**: true
balance_objective optional	Boolean, default is **false**	Setting this parameter to **true** will specify weights for a classification objective which are proportional to their category counts. See this Section for more information. **Example**: true
category optional	Integer, default is **the category of the dataset**	The category that best describes the **model**. See the category codes for the complete list of categories. **Example**: 1
centroid optional	String	The **centroid/id** you want the **model** is associated to. **Example**: "000000"
cluster optional	String	The **cluster/id** you want the **model** is associated to. **Example**: "cluster/60363d8e1f386fc650000015"
dataset	String	A valid **dataset/id**. **Example**: dataset/603e20a91f386f43db000004
datasets optional	Array	A list of dataset ids or objects to be used to build the new **model**. See the Section on Multi-Datasets and Section on Resources Accepting Multi-Datasets Input for more details. **Example**: `[{ "id": "dataset/603e20a91f386f43db000004", "sample_rate": 0.5, "out_of_bag": true }, { "id": "dataset/603e20a91f386f43db000005", "sample_rate": 0.8, "replacement": true }]`

Argument	Type	Description
deep optional	Boolean, default is **false**	Clone the dataset used to build the **model** while cloning the original . Must be used along with the **origin** or **shared_hash** option. **Example**: true
default_numeric_value optional	String	It accepts any of the following strings to substitute missing numeric values across all the numeric fields in the dataset: **mean**, **median**, **minimum**, **maximum**, **zero**. **Example**: "median"
depth_threshold optional	Integer, default is **512**	When the depth in the tree exceeds this value, the tree stops growing. It has no effect if it's bigger than the **node_threshold**. **Example**: 128
description optional	String	A description of the **model** up to 8192 characters long. **Example**: "This is a description of my new model"
excluded_fields optional	Array, default is **[], an empty list. None of the fields in the dataset is excluded**.	Specifies the fields that won't be included in the **model**. **Example**: ["000000", "000002"]

fields
optional
	Object,
	default is **{}, an empty dictionary. That is, no names or preferred statuses are changed**.
		This can be used to change the names of the fields in the model with respect to the original names in the dataset or to tell **BigML** that certain fields should be preferred. An entry keyed with the field id generated in the **source** for each field that you want the name updated.
		Example:
		{
		  "000001": {"name": "length_1"},
		  "000003": {"name": "length_2"},
		}

		A dictionary keyed by **dataset/id** and **object**

Argument	Type	Description

fields_maps
optional
	Object	values. Each entry maps fields in the first dataset to fields in the dataset referenced by the key.
		Example:
		{
		 "dataset/603e20a91f386f43db000004": {
		  "000000":"000023",
		  "000001":"000024",
		  "000002":"00003a"},
		 "dataset/603e20a91f386f43db000005": {
		  "000000":"000023",
		  "000001":"000004",
		  "000002":"00000f"
		}

focus_field optional	String	A field name or identifier for a categorical field. If set, the resulting tree will split first, in a cascade, on all categories of the given field. We are still splitting first on the field, but all nodes are kept binary (i.e., having two children). This is for the convenience of clients that don't know how to handle non-binary splits. **Example**: "000001"
include_extracted_features optional	Boolean or Array of IDs	Extracted image features to use as model inputs. Available options are: • **true**: include all extracted features, unless explicitly excluded (this is the default for all non-deepnet models.) • **false**: don't include any extracted features (this is the default for deepnet models, if not given.) • **list of ids**: a explicit list of field ids, corresponding to extracted fields to add to the default set **Example**: true

Argument	Type	Description
input_fields optional	Array, default is **[]. All the fields in the dataset**.	Specifies the fields to be considered to create the **model**. **Example**: ["000001", "000003"]
max_training_time optional	Integer, default is **1800**	The maximum training time allowed for the optimization, in seconds, as a strictly positive integer. Applicable only when **optimize** is set to **true**. **Example**: 3600
missing_splits optional	Boolean, default is **false**	Defines whether to explicitly include missing field values when choosing a split. When this option is enabled, generates predicates whose operators include an asterisk, such as >*, <=*, =*, or !=*. The presence of an asterisk means *"or missing"*. So a split with the operator >* and the value 8 can be read as "*x > 8 or x is missing*". When using missing_splits there may also be predicates with operators = or !=, but with a null value. This means "*x is missing*" and "*x is not missing*" respectively. **Example**: true
name optional	String default is **dataset's name**	The name you want to give to the new **model**. **Example**: "my new model"
node_threshold optional	Integer, default is **512**	When the number of nodes in the tree exceeds this value, the tree stops growing. **Example**: 1000
number_of_model_candidates optional	Integer, default is **128**	The number of model candidates evaluated over the course of the optimization. Applicable only when **optimize** is set to **true**. Maximum 200 candidates. **Example**: 100
objective_field optional	String, default is **dataset's pre-defined**	Specifies the id of the field that you want to predict. **Example**: "000003"

Argument	Type	Description
	objective field	
objective_fields optional	Array, default is **an array with the id of the last field in the dataset**	Specifies the id of the field that you want to predict. Even if this an array **BigML.io** only accepts one **objective field** in the current version. If both **objective_field** and **objective_fields** are specified then, **objective_field** takes preference. **Example**: ["000003"]
objective_weights optional	Array of Pairs	See Section on Weight. Specific weight for each class in classification models. **Example**: [["Iris-versicolor", 2], ["Iris-virginica", 1], ["Iris-setosa", 1]]
operating_point optional	Object	The specification of an operating point for classification problems. See Prediction or Evaluation for more information.
optimize optional	Boolean, default is **false**	Whether the model should be built with the automatic optimization. When it is set to **true**, only the following modeling properties are applied: **default_numeric_value**, **excluded_fields**, **input_fields**, **max_training_time**, **missing_splits**, **number_of_model_candidates**, **objective_field**, **objective_weights**, **sample_rate**, and **weight_field**. **Example**: true
ordering optional	Integer, default is **0 (deterministic)**	Specifies the type of ordering followed to build the **model**. There are three different types that you can specify: • **0** Deterministic • **1** Linear • **2** Random For more information, see the Section on Shuffling your dataset below. **Example**: 1

Argument	Type	Description
origin optional	String	The **model/id** of the gallery model to be cloned. The price of the model must be 0 to be cloned via API. Set **deep** to **true** to clone the dataset used to build the model too. Note that the dataset can be cloned only if it is already in the public gallery and free. If multiple datasets have been used to create the model, only the first dataset will be cloned. **Example**: "model/6040a3451f386f1a8c000000"
out_of_bag optional	Boolean, default is **false**	Setting this parameter to **true** will return a sequence of the out-of-bag instances instead of the sampled instances. See the Section on Sampling for more details. **Example**: true
project optional	String	The **project/id** you want the **model** to belong to. **Example**: "project/603de73d1f386f7360000000"
random_candidate_ratio optional	Float	A real number between 0 and 1. When **randomize** is true and **random_candidate_ratio** is given, BigML randomizes the tree and uses *random_candidate_ratio * total fields* (counting the number of terms in text fields as fields). To get the final number of candidate fields we round down to the nearest integer, but if the result is 0 we'll use 1 instead. If both **random_candidates** and **random_candidate_ratio** are given, BigML ignores **random_candidate_ratio**. **Example**: 0.2
random_candidates optional	Integer, default is **the square root of the total number of input fields.**	Sets the number of random fields considered when **randomize** is **true**. **Example**: 10

Argument	Type	Description
randomize optional	Boolean, default is **false**	Setting this parameter to true will consider only a subset of the possible fields when choosing a split. See the Section on Random Decision Forests below. **Example**: true
range optional	Array, default is **[1, max rows in the dataset]**	The range of successive instances to build the **model**. **Example**: [1, 150]
replacement optional	Boolean, default is **false**	Whether sampling should be performed with or without replacement. See the Section on Sampling for more details. **Example**: true
sample_rate optional	Float, default is **1.0**	A real number between 0 and 1 specifying the sample rate. See the Section on Sampling for more details. **Example**: 0.5
seed optional	String	A string to be hashed to generate deterministic samples. See the Section on Sampling for more details. **Example**: "MySample"
shared_hash	String	The **shared hash** of the shared model to be cloned. Set **deep** to **true** to clone the dataset used to build the model too. Note that the dataset can be cloned only if it is already shared and set clonable. If multiple datasets have been used to create the model, only the first dataset will be cloned. **Example**: "kpY46mNuNVReITw0Z1mAqoQ9ySW"
split_candidates optional	Integer, default is **32**	The number of split points that are considered whenever the tree evaluates a numeric field. Minimum 1 and maximum 1024 **Example**: 128

Argument	Type	Description
split_field optional	String	A field name or identifier for a categorical field. If set, the first split of the decision tree will use this field and have a children per category (i.e., if there are n categories, the first node will have n elements in its children). **Example**: "000001"
stat_pruning optional	Boolean	Activates statistical pruning on your decision tree **model**. **Example**: true
support_threshold optional	Float, default is **0**	The parameter controls the minimum amount of support each child node must contain to be valid as a possible split. So, if it is 3, then a both children of a new split must have 3 instances supporting them. Since instances may have non-integer weights, non-integer values are valid. **Example**: 16
tags optional	Array of Strings	A list of strings that help classify and index your **model**. **Example**: ["best customers", "2021"]

webhook
optional
	Object	A webhook url and an optional secret phrase. See the Section on Webhooks for more details.
		Example:
		{
		  "url": "http://myhost/path/to/webhook",
		  "secret": "mysecret"
		}

weight_field optional	String	See Section on Weight. Numeric **field id** with no negative or missing values. **Example**: "000001"

You can also use **curl** to customize a new **model**. For example, to create a new **model** named "my model", with only certain rows, and with only three fields:

```
curl

curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
        -X POST \
        -H 'content-type: application/json' \
        -d '{"dataset": "dataset/603e20a91f386f43db000004",
             "input_fields": ["000001", "000003"],
             "name": "my model",
             "range": [25, 125]}'
```

If you do not specify a name, **BigML.io** will assign to the new **model** the **dataset**'s name. If you do not specify a **range** of instances, **BigML.io** will use all the instances in the **dataset**. If you do not specify any **input fields**, **BigML.io** will include all the input fields in the **dataset**, and if you do not specify an **objective field**, **BigML.io** will use the last field in your **dataset**.

Shuffling the Rows of Your Dataset

By default, rows from the input dataset are deterministically shuffled before being processed, to avoid inaccurate models caused by ordered fields in the input rows. Since the shuffling is deterministic, i.e., always the same for a given dataset, retraining a model for the same dataset will always yield the same result.

However, you can modify this default behaviour by including the **ordering** argument in the model creation request, where "ordering" here is a shortcut for "ordering for the traversal of input rows". When this property is absent or set to **0, deterministic shuffling** takes place; otherwise, you can set it to:

- **Linear**: If you know that your input is already in random order. Setting "ordering" to **1** in your model request tells BigML to traverse the dataset in a linear fashion, without performing any shuffling (and therefore operating faster).
- **Random**: If you'd like to perform a really random shuffling, most probably different from any other one attempted before. Setting "ordering" to **2** will shuffle the input rows non-deterministically.

Sampling Your Dataset

You can limit the dataset rows that are used to create a model in two ways (which can be combined), namely, by specifying a row range and by asking for a sample of the (alreaday clipped) input rows.

The row range is specified with the **range** argument defined in the Section on Arguments above.

To specify a sample, which is taken over the row range or over the whole dataset if a range is not provided, you can add the following arguments to the creation request:

1. **sample_rate** : A positive number that specifies the sampling rate, i.e., how often we pick a row from the range. In other words, the final number of rows will be the size of the range multiply by the sample_rate,

unless "out_of_bag" is true (see below).

2. **replacement** : A boolean indicating whether sampling should be performed with or without replacement, i.e., the same instance may be selected multiple times for inclusion in the result set. Defaults to **false**.
3. **out_of_bag** : If an instance isn't selected as part of a sampling, it's called *out of bag*. Setting this parameter to true will return a sequence of the out-of-bag instances instead of the sampled instances. This can be useful when paired with "seed". When **replacement** is **false**, the final number of row returned is the size of the range multiply by one minus the **sample_rate**. Out-of-bag sampling with replacement gives rise to variable-size samples. Defaults to **false**.
4. **seed** : Rows are sampled probabilistically using a random string, which means that, in general, two identical samples of the same row range of the same dataset will be different. If you provide a seed (as an arbitrary string), its hash value will be used as the seed, and it'll be possible for you to generate deterministic samples.

Finally, note that the "ordering" of the dataset described in the previous subsection is used on the result of the sampling.

Here's an example of a model request with range and sampling specifications:

```
curl
```

```
curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
     -X POST \
     -H 'content-type: application/json' \
     -d {"dataset": "dataset/603e20a91f386f43db000004",
         "range": [1, 5000],
         "sample_rate": 0.5,
         "replacement": true}
```

Random Decision Forests

A model can be randomized by setting the **randomize** parameter to true. The default is false.

When randomized, the model considers only a subset of the possible fields when choosing a split. The size of the subset will be the square root of the total number of input fields. So if there are 100 input fields, each split will only consider 10 fields randomly chosen from the 100. Every split will choose a new subset of fields.

Although randomize could be used for other purposes, it's intended for growing **random decision forests**. To grow tree models for a random forest, set **randomize** to true and select a sample from the dataset. Traditionally this is a 1.0 sample rate with replacement, but we suggest a 0.63 sample rate without replacement.

Retrieving a Model

Each **model** has a unique identifier in the form **"model/id"** where id is a string of 24 alpha-numeric characters that you can use to retrieve the **model**.

To retrieve a **model** with **curl**:

```curl
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH"
```

You can also use your browser to visualize the **model** using the full **BigML.io** URL or pasting the **model/id** into the BigML labs dashboard.

Model Properties

Once a **model** has been successfully created it will have the following properties.

Property	Type	Description
balance_objective filterable, sortable	Boolean	Whether to balance classes proportionally to their category counts or not.
boosted_ensemble filterable, sortable	Boolean	Whether the model was built as part of an **ensemble** with boosted trees.

boosting	Object	Boosting attribute for the boosted tree. See the Gradient Boosting section for more information.
		Example:
		{
		"objective_field": "000004",
		"objective_class": "Iris-virginica",
		"weight": 0.09984,
		"lambda": 1
		}

category filterable, sortable, updatable	Integer	One of the categories in the table of categories that help classify this resource according to the domain of application.

Property	Type	Description
centroid filterable, sortable	String	The **centroid id** this model was built for.
cluster filterable, sortable	String	The **cluster/id** this model was built for.
cluster_status filterable, sortable	Boolean	Whether the **cluster** is still available or has been deleted.
code	Integer	One of the HTTP status code. This will be 201 upon successful creation of the **model** and 200 afterwards. Make sure that you check the code that comes with the status attribute to make sure that the **model** creation has been completed without errors.
columns filterable, sortable	Integer	The number of fields in the **model**.
composites filterable, sortable	Array of Strings	The list of composite ids that reference this **model**.
created filterable, sortable	ISO-8601 Datetime	This is the date and time in which the **model** was created with microsecond precision. It follows this pattern yyyy-MM-ddThh:mm:ss.SSSSSS. All times are provided in Coordinated Universal Time (UTC).
creator	String	The user that created the **model**.
dataset filterable, sortable	String	The **dataset/id** that was used to build the **model**.
dataset_field_types	Object	A dictionary that informs about the number of fields of each type in the dataset used to create the **model**. It has an entry per each field type (**categorical**, **datetime**, **numeric**, **text**, **image**, **path** and **regions**), an entry for **preferred** fields, and an entry for the **total** number of fields.
dataset_status filterable, sortable	Boolean	Whether the **dataset** is still available or has been deleted.

Property	Type	Description
datasets	Array	A list of dataset ids or objects used to build the **model**.
default_numeric_value	String	Any of the following strings to substitute missing numeric values across all the numeric fields in the dataset: **mean**, **median**, **minimum**, **maximum**, **zero**.
depth_threshold	Integer	The depth, or generation, limit for a tree.
description updatable	String	A text describing the **model**. It can contain restricted markdown to decorate the text.
ensemble filterable, sortable	Boolean	Whether the **model** was built as part of an ensemble of not.
ensemble_id filterable, sortable	String	The ensemble **id**.
ensemble_index filterable, sortable	Integer	The number of order in the ensemble.
excluded_fields	Array	The list of **fields**'s ids that were excluded to build the **model**.
execution_id filterable, sortable	String	The **execution/id** that created the **model**.
execution_status filterable, sortable	Boolean	Whether the **execution** is still available or has been deleted.
fields_meta	Object	A dictionary with meta information about the fields dictionary. It specifies the **total** number of fields, the current **offset**, and **limit**, and the number of fields (**count**) returned.
focus_field	String	Specifies the id of the focus field in the **model**. **Example**: "000001"
focus_field_name	String	The name of the **focus field** in the **model**.

Property	Type	Description
fusions	Array of Strings	The list of fusion ids that reference this **model**.
input_fields	Array	The list of **input fields**' ids used to build the models of the **model**.
locale	String	The **dataset**'s locale.
max_columns filterable, sortable	Integer	The total number of fields in the **dataset** used to build the **model**.
max_rows filterable, sortable	Integer	The maximum number of instances in the **dataset** that can be used to build the **model**.
max_training_time	Integer	The maximum training time allowed for the optimization, in seconds.
missing_splits filterable, sortable	Boolean	Whether to explicitly include missing field values when choosing a split while growing a **model**.
model	Object	All the information that you need to recreate or use the model on your own. It includes a very intuitive description of the tree-like structure that makes the model up and the **field**'s dictionary describing the fields and their summaries.
name filterable, sortable, updatable	String	The name of the **model** as your provided or based on the name of the **dataset** by default.
name_options filterable, sortable	String	Information about the **model**.
node_threshold filterable, sortable	String	The maximum number of nodes that the **model** will grow.
number_of_batchpredictions filterable, sortable	Integer	The current number of **batch predictions** that use this **model**.
number_of_evaluations filterable, sortable	Integer	The current number of **evaluations** that use this **model**.

Property	Type	Description
number_of_model_candidates	Integer	The number of model candidates evaluated over the course of the optimization.
number_of_predictions filterable, sortable	Integer	The current number of **predictions** that use this **model**.
number_of_public_predictions filterable, sortable	Integer	The current number of **public predictions** that use this **model**.
objective_field	String	The id of the field that the **model** predicts.
objective_field_name	String	The name of the field used as the objective for the **model**. **Example**: "species"
objective_field_type	String	The type of the field used as the objective for a **model**. **Example**: "categorical"
objective_fields	Array	Specifies the list of ids of the field that the **model** predicts. Even if this is an array **BigML.io** only accepts one **objective field** in the current version.
operating_point updatable	Object	The specification of an operating point for classification problems.
optimize	Boolean	Whether the **model** was built with the automatic optimization.
optiml filterable, sortable	String	The **optiml/id** that created this **model**.
optiml_status filterable, sortable	Boolean	Whether the **OptiML** is still available or has been deleted.
		The order used to chose instances from the dataset to build the model. There are three different types:

Property	Type	Description
ordering filterable, sortable	Integer	• **0** Deterministic • **1** Linear • **2** Random
origin filterable, sortable	String	The **model/id** of the original gallery model.
out_of_bag filterable, sortable	Boolean	Whether the out-of-bag instances were used to create the **model** instead of the sampled instances.
price filterable, sortable, updatable	Float	The price other users must pay to clone your **model**.
private filterable, sortable, updatable	Boolean	Whether the **model** is public or not.
project filterable, sortable, updatable	String	The **project/id** the resource belongs to.
random_candidate_ratio filterable, sortable	Float	The random candidate ratio considered when **randomize** is **true**.
random_candidates filterable, sortable	Integer	The number of random fields considered when **randomize** is **true**.
randomize filterable, sortable	Boolean	Whether the model splits considered only a random subset of the fields or all the fields available.
range	Array	The **range** of instances used to build the **model**.
replacement filterable, sortable	Boolean	Whether the instances sampled to build the **model** were selected using replacement or not.
resource	String	The **model/id**.
rows filterable, sortable	Integer	The total number of instances used to build the **model**.

Property	Type	Description
sample_rate filterable, sortable	Float	The sample rate used to select instances from the **dataset** to build the **model**.
seed filterable, sortable	String	The string that was used to generate the sample.
selective_pruning filterable, sortable	Boolean	If true, selective pruning throttled the strength of the statistical pruning depending on the size of the **dataset**.
shared filterable, sortable, updatable	Boolean	Whether the **model** is shared using a private link or not.
shared_clonable filterable, sortable, updatable	Boolean	Whether the shared **model** can be cloned or not.
shared_hash	String	The hash that gives access to this **model** if it has been shared using a private link.
sharing_key	String	The alternative key that gives read access to this **model**.
size filterable, sortable	Integer	The number of bytes of the **dataset** that were used to create this **model**.
source filterable, sortable	String	The **source/id** that was used to build the **dataset**.
source_status filterable, sortable	Boolean	Whether the **source** is still available or has been deleted.
split_candidates filterable, sortable	Integer	The number of split points that are considered whenever the tree evaluates a numeric field. Minimum 1 and maximum 1024.
split_field	String	Specifies the id of the split field in the **model**. **Example**: "000001"
split_field_name	String	The name of the **split field** in the **model**.

Property	Type	Description
stat_pruning filterable, sortable	Boolean	Whether statistical pruning was used when building the **model**.
status	Object	A description of the status of the model. It includes a code, a message, and some extra information. See the table below.
subscription filterable, sortable	Boolean	Whether the **model** was created using a subscription plan or not.
support_threshold filterable, sortable	Float	The parameter controls the minimum amount of support each child node must contain to be valid as a possible split.
tags filterable, updatable	Array of Strings	A list of user tags that can help classify and index this resource.
updated filterable, sortable	ISO-8601 Datetime	This is the date and time in which the **model** was updated with microsecond precision. It follows this pattern yyyy-MM-ddThh:mm:ss.SSSSSS. All times are provided in Coordinated Universal Time (UTC).
webhook	Object	A webhook url and an optional secret phrase. See the Section on Webhooks for more details.
weight_field	Boolean	Specifies the id of the weight field in the **model**.
white_box filterable, sortable	Boolean	Whether the **model** is publicly shared as a white-box.

Model Object

Property	Type	Description
depth_threshold	Integer	The depth, or generation, limit for a tree.
		This dictionary gives information about how the training data is distributed across the tree leaves. More concretely, it contains the

Property	Type	Description
distribution	Object	training data distribution with key **training**, and the distribution for the actual prediction values of the tree with key **predictions**. The former is just the **objective_summary** of the tree root (see below), copied for easier individual retrieval, and both have the format of the objective summary in the tree nodes.
fields	Object	A dictionary with an entry per field in the dataset used to build the model. Fields are paginated according to the **field_meta** attribute. Each entry includes the column number in original source, the name of the field, the type of the field, and the **summary**. See this Section for more details.
importance	Array of Arrays	A list of pairs [**field_id**, **importance**]. Importance is the amount by which each field in the model reduces prediction error, normalized to be between zero and one. Note that fields with an importance of zero may still be correlated with the objective; they were just not used in the model.
kind	String	The type of model. Currently, only **stree** is supported.
missing_strategy	String	Default strategy followed by the model when it finds a missing value. Currently, **last_prediction**. At prediction time you can opt for using **proportional**. See this Section for more details.
model_fields	Object	A dictionary with an entry per field used by the model (not all the fields that were available in the dataset). They follow the same structure as the **fields** attribute above except that the summary is not present.
root	Object	A Node Object, a tree-like recursive structure representing the model.
support_threshold	Float	A number between 0 and 1. For a split to be valid, each child's support (instances / total instances) must be greater than this threshold.

Node Object

Property	Type	Description
children	Array	Array of Node Objects.
confidence	Float	For classification models, a number between 0 and 1 that expresses how certain the model is of the prediction. For regression models, a number mapped to the top end of a 95 confidence interval around the expected error at that node (measured using the variance of the output at the node). See the Section on Confidence for more details. Note that for models you might have created using the first versions of **BigML** this value might be **null**.
count	Integer	Number of instances classfied by this node.
objective_summary	Object	An Objective Summary Object summarizes the objective field's distribution at this node.
output	Number or String	Prediction at this node.
predicate	Boolean or Object	Predicate structure to make a decision at this node.

Objective Summary

Property	Type	Description
bins	Array	If the objective field is **numeric** and the number of distinct values is greater than **32**, an array that represents an approximate histogram of the distribution. It consists of value pairs, where the first value is the mean of a histogram bin and the second value is the bin population. For more information, see our blog post or read this paper.
categories	Array	If the objective field is **categorical**, an array of pairs where the first element of each pair is one of the unique categories and the second element is the count for that category.
counts	Array	If the objective field is **numeric** and the number of distinct values is less than or equal to **32**, an array of pairs where the first element of each pair is one of

Property	Type	Description
		the unique values found in the field and the second element is the count.
maximum	Number	The maximum of the objective field's values. Available when 'bins' is present.
minimum	Number	The minimum of the objective field's values. Available when 'bins' is present.

Predicate Object

Property	Type	Description
field	String	Field's id used for this decision.
operator	String	Type of test used for this field.
value	Number or String	Value of the field to make this node decision.

Model Status

Creating **model** is a process that can take just a few seconds or a few days depending on the size of the **dataset** used as input and on the workload of **BigML**'s systems. The **model** goes through a number of states until its fully completed. Through the status field in the **model** you can determine when the model has been fully processed and ready to be used to create predictions. These are the properties that **model**'s **status** has:

Property	Type	Description
code	Integer	A status code that reflects the status of the **resource** creation. It can be any of those that are explained here.
elapsed	Integer	Number of milliseconds that **BigML.io** took to process the **resource**.
message	String	A human readable message explaining the status.
progress	Float, between 0 and 1	How far **BigML.io** has progressed building the **resource**.

Once a **model** has been successfully created, it will look like:

json

```json
{
    "boosted_ensemble": false,
    "boosting": {},
    "category": 0,
    "cluster": null,
    "cluster_status": false,
    "code": 200,
    "columns": 5,
    "configuration": null,
    "configuration_status": false,
    "created": "2021-03-04T09:07:17.111000",
    "creator": "alfred",
    "dataset": "dataset/603e20a91f386f43db000004",
    "dataset_field_types": {
```

Filtering a Model

It is possible to filter the tree returned by a GET to the model location by means of two optional query string parameters, namely **support** and **value**.

Filter by Support

Support is a number from 0 to 1 that specifies the minimum fraction of the total number of instances that a given branch must cover to be retained in the resulting tree. Thus, asking for (minimum) support of 0, is just asking for the whole tree, while something like:

curl

```
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&support=1.0"
```

will return just the root node, that being the only one that covers all instances. If you repeat the **support** parameter in the query string, the last one is used. Non-parseable support values are ignored.

Filter by Values and Value Intervals

Value is a concrete value or interval of values (for regression trees) that a leaf must predict to be kept in the returning tree. For instance:

curl

```
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&value=Iris-setos
```

will return only those branches in the tree whose leaves predict "Iris-setosa" as the value of the (categorical) objective field, while something like:

```curl
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&value=[10,20]"
```

for a regression model will include only those leaves predicting an objective value between 10 and 20. You can also specify sharp values for regression models:

```curl
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&value=23.2"
```

will retrieve only those branches whose predictions are exactly 23.2. It is possible to specify multiple values, as in:

```curl
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&value=Iris-setos

curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&value=(10,20]&va

curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&value=(10.2,20)&
```

in which case the union of the different predicates is used (i.e., the first query will return a tree will all leaves predicting "Iris-setosa" and all leaves predicting "Iris-versicolor".

Intervals can be closed or open in either end. For example, "(-2,10]", "[1,2)" or "(-1.234,0)", and the values of the left or right limits can be omitted, in which case they're taken as negative and positive infinity, respectively; thus "(,3]" denotes all values less or equal to three, as does "[,3]" (infinity not being a valid value for a numeric prediction), while "(0,)" accepts any positive value.

Filter by Confidence / Probability / Expected Error

Confidence is a concrete value or interval of values that a leaf must have to be kept in the returning tree. The specification of intervals follows the same conventions as those of **value**. Since confidences are a continuous value, the most common case will be asking for a range, but the service will accept also individual values. It's also possible to specify both a **value** and a **confidence**. For instance:

curl

```
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&value=Iris-setos
```

asks for a tree with only those leaves that predict "Iris-setosa" with a confidence greater or equal to 0.3, while

curl

```
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&confidence=[,0.
```

returns a model where all leaves with confidence strictly less than 0.25. Confidence filters will work both for classification regression problems, since we call the regression expected error **confidence** in our JSON. If desired (and only for regression), one can specify a filter using **expected_error** instead:

curl

```
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&expected_error=
```

If you specify both **confidence** and **expected_error**, only one of them will be used: **confidence** for classifications, **expected_error** for regressions. If only **confidence** is specified, it will always be used (confidence is an alias for the expected error in regressions). If only **expected_error** is specified, it will only be used if the model is a regression.

Filters by **probability** works exactly as filters by **confidence**, but replacing **probability** for **confidence**. As a consequence, they'll only have an effect on classification problems.

Finally, note that it is also possible to specify **support**, **value**, **confidence**, **probability**, and **expected_error** parameters in the same query.

PMML

The default model output format is JSON. However, the **pmml** parameter allows to include a PMML version of the model. The model will include a XML document that fullfils PMML v4.1. For example:

curl

```
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&pmml=yes"
```

will include the PMML version of the model together with the JSON representation. While:

```curl
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH&pmml=only"
```

will include the PMML version of the model but not all the usual JSON fields. Some fields will be incomplete or not even be returned.

Filtering and Paginating Fields from a Model

A **model** might be composed of hundreds or even thousands of fields. Thus when retrieving a **model**, it's possible to specify that only a subset of fields be retrieved, by using any combination of the following parameters in the query string (unrecognized parameters are ignored):

Parameter	Type	Description
fields optional	Comma-separated list	A comma-separated list of **field** IDs to retrieve. **Example**: "fields=000000,000002"
full optional	Boolean	If false, no information about fields is returned. **Example**: "full=false"
iprefix optional	String	A case-insensitive string to retrieve fields whose name start with the given prefix; It is possible to specify more than one iprefix by repeating the parameter, in which case the union of the results is returned. **Example**: "iprefix=INCOME"
limit optional	Integer	Maximum number of **fields** that you will get in the **fields** field. **Example**: "limit=100"
offset optional	Integer	How far off from the first **field** in your **dataset** is the first **field** in the **fields** field. **Example**: "offset=100"
order_by optional	String	Sorting criteria; possible values are **"count"**, **"max"**, **"min"**, **"name"**, and **"type"**, and their negated values (**"-count"**, **"-name"**, etc.) to specify a descending order. **Example**: "order_by=name"

Parameter	Type	Description
prefix optional	String	A case-sensitive string to retrieve fields whose name start with the given prefix; It is possible to specify more than one prefix by repeating the parameter, in which case the union of the results is returned. **Example**: "prefix=income"

Since **fields** is a map and therefore not ordered, the returned fields contain an additional key, **order**, whose integer (increasing) value gives you their ordering. In all other respects, the source is the same as the one you would get without any filtering parameter above.

The **fields_meta** field can help you paginate fields. Its structure is as follows:

Property	Type	Description
count optional	Integer	Specifies the current number of fields in the resource.
limit optional	Integer	The maximum number of fields that will be returned in the resource.
offset optional	Integer	The current offset in the pagination of fields.
total optional	Integer	The total number of fields in the resource.

Note that paginating fields might only be worth if you are going to deal with really wide (i.e., more than 200 fields).

Updating a Model

To update a **model**, you need to PUT an object containing the fields that you want to update to the **model**'s base URL. The content-type must always be: **"application/json"**. If the request succeeds, **BigML.io** will return with an HTTP 202 response with the updated **model**.

For example, to update **model** with a new name you can use **curl** like this:

```
curl
```

```
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH" \
     -X PUT \
     -H 'content-type: application/json' \
     -d '{"name": "a new name"}'
```

If you want to update **model** with a new **label** and **description** for a specific field you can use **curl** like this:

```curl
curl "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH" \
     -X PUT \
     -H 'content-type: application/json' \
     -d '{"fields": {
            "000000": {
                "label": "a longer name",
                "description": "an even longer description"
            }
        }}'
```

See this section for more details.

Deleting a Model

To delete a **model**, you need to issue a HTTP DELETE request to the **model/id** to be deleted.

Using **curl** you can do something like this to delete a **model**:

```curl
curl -X DELETE "https://bigml.io/andromeda/model/6040a3451f386f1a8c000000?$BIGML_AUTH"
```

If the request succeeds you will not see anything on the command line unless you executed the command in verbose mode. Successful DELETEs will return **"204 no content"** responses with no body.

Once you delete a **model**, it is permanently deleted. That is, a delete request cannot be undone. If you try to delete a **model** a second time, or a **model** that does not exist, you will receive a **"404 not found"** response.

However, if you try to delete a **model** that is being used at the moment, then **BigML.io** will not accept the request and will respond with a **"400 bad request"** response.

See this section for more details.

Listing Models

To list all the **models**, you can use the **model** base URL. By default, only the 20 most recent **models** will be returned. You can see below how to change this number using the **limit** parameter.

You can get your list of **models** using **curl**.

```curl
curl "https://bigml.io/andromeda/model?$BIGML_AUTH"
```

See this section for more details. You can also paginate, filter, and order your **models**.

Weights

BigML.io has added three new ways in which you can use **weights** to deal with imbalanced datasets:

1. **Weight Field**: considering the values one of the fields in the dataset as weight for the instances. This is valid for both regression and classification models.
2. **Objective Weights**: submitting a specific weight for each class in classification models.
3. **Automatic Balancing**: setting the **balance** argument to **true** to let BigML automatically balance all the classes evenly.

Weight Field

A **weight_field** may be declared for either regression or classification models. Any numeric field with no negative or missing values is valid as a weight field. Each instance will be weighted individually according to the weight field's value. See the toy dataset for credit card transactions below.

```bash
online, transaction, pending transactions, days since last transaction, distance, transact
yes, 10, 3, 31, low, 3, -3250, -1500, no, 1
no, 20, 30, 1, high, 0, 0, -300, no, 1
no, 40, 13, 210, low, 1, -19890, -30, no, 1
yes, 500, 0, 1, high, 0, 0, 0, yes, 10
no, 10, 1, 32, low, 0, -2500, -7891, no, 1
```

```
yes, 100, 0, 3, low, 0, -5194, -120, no, 1
yes, 100, 1, 4, low, 0, 0, 1500, no, 1
yes, 1000, 0, 1, high, 0, 0, 0, yes, 10
no, 150, 3, 1, low, 5, -3250, 1500, no, 1
no, 75, 5, 1, high, 1, -3250, 1500, no, 1
yes, 10, 23, 0, low, 1, -3250, 1500, no, 1
yes, 10, 3, 31, low, 3, -3250, -1500, no, 1
```

The last column represents the **weight** for each transaction. We can use it as an input to create a model that will use to weight each instance accordingly. In this case, fraudulent transactions will weight 10 times more than valid transactions in the model building computations.

```curl
curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
     -X POST \
     -H 'content-type: application/json' \
     -d '{"dataset": "dataset/603e20a91f386f43db000004",
          "objective_field": "000008",
          "weight_field": "000009"
         }'
```

With Flatline, you can define arbitrarily complex functions to produce weight fields, making this the most flexible and powerful way to produce weighted models.

For instance, the request below would create a new dataset using the example above that will add a new weight field using the previous and multiplying by two when the amount of the transaction is higher than 500.

```curl
curl "https://bigml.io/andromeda/dataset?$BIGML_AUTH" \
     -X POST \
     -H 'content-type: application/json' \
     -d '{"dataset": "dataset/603e20a91f386f43db000004",
          "new_fields": [{
            "field": "(if (and (= (f fraud) \"yes\") (> (f transaction) 500)) (* (f weight)
            "name": "new weight"}]
         }'
```

This method also works well when you query very large databases that can produce the same row hundreds or thousands of times. You can just use one of the rows and add the corresponding count as a weight field. This will reduce the size of your sources enormously.

Objective Weights

The second method for adding weights only applies to classification models. A set of **objective_weights** may be defined, one per objective class. Each instance will be weighted according to its class weight.

```
curl
```

```
curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
     -X POST \
     -H 'content-type: application/json' \
     -d '{"dataset": "dataset/603e20a91f386f43db000004",
          "objective_field": "000008",
          "excluded_fields": ["000009"],
          "objective_weights": [["yes", 10], ["no", 1]]
         }'
```

If a class is not listed in the objective_weights, it is assumed to have a weight of 1. This means the example below is equivalent to the example above.

```
curl
```

```
curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
     -X POST \
     -H 'content-type: application/json' \
     -d '{"dataset": "dataset/603e20a91f386f43db000004",
          "objective_field": "000008",
          "excluded_fields": ["000009"],
          "objective_weights": [["yes", 10]]
         }'
```

Weights of zero are valid as long as there are some positive valued weights. If every weight does end up zero (this is possible with sampled datasets), then the resulting model will have a single node with a nil output.

Automatic Balancing

Finally, we provide a convenience shortcut for specifying weights for a classification objective which are proportional to their category counts, by means of the **balance_objective** flag.

For instance, if the category counts of the objective field are, say:

```bash
[["Iris-versicolor", 20], ["Iris-virginica", 10], ["Iris-setosa", 5]]
```

the request:

```curl
curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
     -X POST \
     -H 'content-type: application/json' \
     -d '{"dataset": "dataset/603e20a91f386f43db000004",
          "balance_objective": true
         }'
```

would be equivalent to:

```curl
curl "https://bigml.io/andromeda/model?$BIGML_AUTH" \
     -X POST \
     -H 'content-type: application/json' \
     -d '{"dataset": "dataset/603e20a91f386f43db000004",
          "objective_weights": [
            ["Iris-versicolor", 1],
            ["Iris-virginica", 2],
            ["Iris-setosa", 4]]}'
```

The next table summarizes all the available arguments to use weights.

Argument	Type	Description
balance_objective optional	Boolean, default is **false**	Whether to balance classes proportionally to their category counts or not. **Example**: true
		A list of category and weight pairs. One per objective class. **Example**:

Argument	Type	Description
objective_weights optional	Array	[["Iris-versicolor", 2], ["Iris-virginica", 1], ["Iris-setosa", 1]]
weight_field optional	String	Any numeric field with no negative or missing values is valid as a weight field. Each instance will be weighted individually according to the weight field's value. **Example**: "000005"

The nodes for a weighted tree will include a **weight** and **weighted_objective_distribution**, which are the weighted analogs of count and **objective_distribution**. Confidence, importance, and pruning calculations also take weights into account.

```json
{
    "id":0,
    "children":[
        {
            "id":1,
            "children":[
                {
                    "output":"Iris-virginica",
                    "count":10,
                    "objective_summary":{
                        "categories":[
                            [
                                "Iris-virginica",
                                10
```