Skytree REST API

Release 15.4.0

SKYTREE®

Notices

Copyright

Disclaimer

Skytree, Inc. reserves the right to revise this document and to make changes from time to time without any obligation of any kind to notify any person of such revisions or changes.

Trademarks

Company Information:
Skytree, Inc.
1731 Technology Drive, Suite 700
San Jose, CA 95110
408.392.9300
www.skytree.net

Customers with a support contact can contact support using any of the following methods:

- Log in to support.skytree.net.
- Send questions to support@skytree.net.
- Call 408.392.9300 and select option 3.
- You can also view additional Skytree documentation by logging on to support.skytree.net.

Freemium customers can send inquiries to the LinkedIn® Community Group for Skytree:
https://www.linkedin.com/groups/4468542

Revision Date: February 29, 2016

SKYTREE®

Contents

Chapter 8 Models 75

SKYTREE®

Chapter 1 About this Document

Skytree provides a series of REST APIs that enable you to perform the following functions:

- Authenticate users
- Import/upload source data
- Transform raw source data into a dataset
- Train the dataset to create a model
- Test a model
- Make predictions
- Generate plots for models and results

Communication

All Skytree requests and returns are JSON formatted. Most requests must include `application/json` with the following exceptions:

- When uploading a file from a local disk, the communication type is `multipart/form-data`.
- When retrieving PMML output for GBT models in text format, the communication type is `text/xml`.
- When retrieving PMML output for GBT models in a zipped format, the communication type is `application/gzip`.

Assumptions

The examples in this document assume that the API server is available at http://api.skytree.net:8080. Replace this URL with the appropriate address.

HTTP Methods

The Skytree API uses standard HTTP methods to create, update, and retrieve resources.

Table 1: *Skytree API Methods*

Method	Description
POST	Creates a resource, such as a source, dataset, etc
PUT	Updates a resource
GET	Retrieves a resource. This is the default method.
DELETE	Deletes a resource

HTTP Status Codes

The Skytree API uses standard HTTP status codes in each stored response message. The status code definitions are based on RFC 2616.

Note: You can view responses immediately upon entering a command by including the `-verbose` keyword.

Skytree Status Codes

Skytree returns the following status codes in a method request.

Table 2: *Skytree Status Codes*

Status Code	Description
INQUEUE	The resource is ready to be processed.
INPROGRESS	The resource is currently being processed. Note that this status does not apply to training or testing resources.
ETL_INPROGRESS	The training or testing resource is currently undergoing data preparation.
ML_INPROGRESS	The training or testing resource is currently undergoing a machine learning process.
READY	The resource is ready to be used.
ERROR	An error occurred while processing the request.

Chapter 2 Authentication

All users must be authenticated before accessing Skytree. Authenticated users are provided with a unique user ID, also referred to as a token. This information is required in every request automatically.

The authentication APIs allow admins to authorize a new user. The authentication process should be similar to below:

1. The administrator authorizes a new user by providing the user's email address and a default password.
2. The administrator delivers the password and token to the user.
3. The user retains the token and changes the password away from the default. A user can regenerate a new token if the original is lost.

> **Note:** In this version of Skytree, it is currently not possible to retrieve a missing password. If a password is lost or forgotten, contact your administrator to reset your password.

URLs

Use the following base URLs to update user information.

```
'http://api.skytree.net:8080/v1/userPrefs/authenticate'
'http://api.skytree.net:8080/v1/userPrefs/changePassword'
'http://api.skytree.net:8080/v1/userPrefs/about'
```

Authenticate User

Tokens are required in every API request. Use the `authenticate` API to generate a token. Note that this API can also be called to generate new tokens in the event that you lose a token or when a token expires.

Request

Use `POST` to authenticate a user and return a new token. The username (the login name when using LDAP or the e-mail address when not using LDAP) and password are required in this request.

```
curl -X POST -H "Content-type:application/json" \
-d '{"userName":"user1@skytree.net", "password":"Password2"}' \
'http://api.skytree.net:8080/v1/userPrefs/authenticate'
```

Response

The response returns the user's email address and the new token. Be sure to retain this token because it is required in every request. In the event that you lose this token or the token expires, run the `authenticate` request again to generate a new token.

Warning: For security purposes, unused authentication tokens have a lifespan of 6 hours. This means that if you do not use this token, it will expire after 6 hours. When you do use this token, its expiry time will increment by another 6 hours. Run the `authenticate` request to generate a new token.

```
200 (OK)
"Content-type:application/json"
{
 "userName":"user1@skytree.net",
 "token":"dXNlcjFA2t5dHJlZS5uZXQ6cGFz"
}
```

Change Password

Administrators provide you with your initial password that was used when your account was created. After you generate a token, you can call the `changePassword` API to change this password to something more memorable or meaningful. New passwords must be a minimum of eight characters and include at least one uppercase letter, one lowercase letter, and one number. Special characters are permitted.

Note: This REST API is not available for LDAP authentication users.

Request

Use `POST` to change the password for a current user. The user's e-mail address and the original password are required in this request.

```
curl -X POST "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{"userName":"user1@skytree.net", "oldPassword":"Password1",
   "newPassword":"Password2"}' \
'http://api.skytree.net:8080/v1/userPrefs/changePassword'
```

Response

Similar to when the user was created, the API returns the user's e-mail address, old password, and new password upon successful completion. Otherwise, an error will display if any information was entered incorrectly.

```
200 (OK)
"Content-type:application/json"
{
 "userName":"user1@skytree.net",
 "oldPassword":"Password1",
 "newPassword":"Password2"
}
```

Lost Password

In this version of Skytree, it is not currently possible for a user to retrieve a lost password. Contact your administrator if you forget your password. Your administrator reset your password.

Retrieve User Information

User information is supplied by Administrators during the authorization process. The `about` API allows users to retrieve system information about themselves.

Request

Use `GET` to retrieve the system's information about you.

```
curl -X GET -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/userPrefs/about'
```

Response

The response returns the user's full name, ID, username (e-mail address if not using LDAP or login name if using LDAP), login name, and creation and last updated information.

```
200 (OK)
"Content-type:application/json"
{
 "name":"John Smith",
 "id":523452368326,
 "userName":"user1@skytree.net",
 "loginName":"john",
 "updatedAt":1421275938026,
 "createdAt":1421262811001
}
```

Authentication Return Fields

The following table lists the fields that are returned after successful Authentication requests. Note that these can vary depending on the type of request.

Table 3: *Authentication Return Properties*

Return Field	Description
`userName`	The user's login name when using LDAP authentication or the user's e-mail address when not using LDAP authentication
`token`	The user's token
`oldPassword`	The password that the user wants to change
`newPassword`	The user's new password
`name`	The current user's name as defined by the admin during authorization

Table 3: *Authentication Return Properties (continued)*

Return Field	Description
`id`	The current user's randomly generated ID
`loginName`	The current user's login name. For users in a Kerberos Hadoop environment, this tracks the user's primary Kerberos principal. For other users, this field contains the user's Linux username.
`createdAt`	The epoch time when the user was created
`updatedAt`	The epoch time when the user was updated

Authentication Parameters

The following parameters are used in the Authentication APIs.

Table 4: *Authentication API Parameters*

Parameter	Description	Used In
`userName`	The user's login name when using LDAP authentication or the user's e-mail address when not using LDAP authentication	Authenticate user Change password,
`password`	The user's current password	Authenticate user
`oldPassword`	The password that the user wants to change	Change password
`newPassword`	The user's new password	Change password

Chapter 3 Projects

The Skytree API is organized around projects. A project contains the datasets, models, and tests results corresponding to a machine learning task. Each project retains a full audit train of how each dataset, model, or result was created.

> **Note:** A valid `projectID` must be included with every creation request. In addition, all data used when creating models, transforms, and tests must reside in the same project. For example, when creating a model, an error will result if you attempt to reference a dataset that resides in a project other than the project that is specified in the request.

URLs

Use the following base URLs to manage projects:

```
'http://api.skytree.net:8080/v1/projects'
'http://api.skytree.net:8080/v1/projects/<id>'
'http://api.skytree.net:8080/v1/projects/<id>/dependents'
```

Create a Project

This section describes how to create a project. Once created, sources, datasets, and models can be associated with this project by specifying the Project ID when they are created.

Request

Use `POST` to create a create a project. Note that the `description` is optional.

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
-d '{
     "name": "Customer Acquisition",
     "description": "New customers",
    }' 'http://api.skytree.net:8080/v1/projects'
```

Response

Upon successful completion, the response returns the unique ID for the new project. This ID must be specified when creating sources and datasets, when training models, and when performing tests and predictions.

```
200 (OK)
"Content-type:application/json"

{
 "id":"24389108236",
 "name":"Customer Acquisition",
 "description":"New customers",
 "createdAt":1412275232077,
 "updatedAt":1412275232077
}
```

Edit a Project

This section describes how to edit an existing project.

Request

Use PUT to edit an existing project. In the request below, a project whose ID is 24389108236 is edited by changing the description.

```
curl -X PUT -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
-d '{
     "name": "Customer Acquisition",
     "description": "Acquire new customers",
    }' 'http://api.skytree.net:8080/v1/projects/24389108236'
```

Response

Upon successful completion, the response returns information about the updated project. In this case, only the description was changed.

```
200 (OK)
"Content-type:application/json"

{
 "id":"24389108236",
 "name":"Customer Acquisition",
 "description":"Acquire new customers",
 "createdAt":1412275232077,
 "updatedAt":1412278867206
}
```

Retrieve a Project

This section describes how to retrieve a single project.

Request

Use GET along with a project ID to retrieve a single project. The request below shows how to retrieve a project whose ID is 24389108236.

```
curl -X GET -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
```

```
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/projects/24389108236'
```

Response

Upon successful completion, the response returns the name and description (if available) of the specified project.

```
200 (OK)
"Content-type:application/json"

{
 "id":"24389108236",
 "name":"Customer Acquisition",
 "description":"Acquire new customers",
 "createdAt":1412275232077,
 "updatedAt":1412278867206
}
```

Retrieve All Projects

This section describes how to retrieve a list of all projects associated with the current user.

Request

Use GET without an accompanying ID to retrieve all projects.

```
curl -X GET -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/projects
```

Response

Upon successful completion, the response returns the name and description (if available) of all currently available projects along with the creation and update times.

```
200 (OK)
"Content-type:application/json"

[
 {
  "id":"24389108236",
  "name":"Customer Acquisition",
  "description":"Acquire new customers",
  "createdAt":1412275232077,
  "updatedAt":1412278867206
 },
 {
 "id":"68291338234",
  "name":"Fraud",
  "description":"Determine Fraudulent Activity",
  "createdAt":1403045348903,
  "updatedAt":1403045349066
 }
]
```

Retrieve Project Dependents

Use this function to retrieve a list of all datasets, models, results, and plots included in a project.

Request

Use GET along with a project ID to retrieve all files included in the project. The request below shows how to retrieve a list of files in a project whose ID is 24389108236.

```
curl -X GET -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/projects/24389108236/dependents'
```

Response

Upon successful completion, the response returns a list of files included in the project.

```
200 (OK)
"Content-type:application/json"

{
 "id":"24389108236",
 "name":"Prediction",
 "type":"PROJECT",
 "is_deletable":TRUE
},
{
 "id":"4274348704094814824",
 "name":"income.data",
 "type":"DATASET",
 "is_deletable":TRUE
},
{
 "id":"4426696063906522815",
 "name":"income.data.ids",
 "type":"DATASET",
 "is_deletable":TRUE
}
{
 "id":"8613266560949776288",
 "name":"GBT Model",
 "type":"MODEL",
 "is_deletable":TRUE
},
{
 "id":"733579628360679490",
 "name":"GBT Result",
 "type":"TESTRESULT",
 "is_deletable":TRUE
}
```

Delete a Project

This section describes how to delete an existing project. All content will be removed from this project, including all associated sources, datasets, models, and test results

Request

Use `DELETE` to delete an existing project. The request below deletes a project whose ID is 24389108236.

```
curl -X DELETE -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/projects/24389108236'
```

Response

The API returns 204 `No Content` upon successful completion.

Project Parameters

The table below shows parameters that are used in the Project creation, edit, delete, and retrieval APIs.

Table 5: *Project API Parameters*

Parameter	Description	Used In
`name`	The name of a project. Project names must be unique.	Create a project, Edit a project
`description`	Optionally specify a description for the new project	Create a project, Edit a project
`id`	A unique identifier for the project	Retrieve a project, Retrieve dependents, Edit a project, Delete a project

Chapter 4 Sources

A source file represents the raw data that you plan to transform into a dataset. Source files must be in CSV format. They include the data that you want to use for making predictions. The Source API validates the data in the file and then returns a unique source ID. This identifier is then referenced when you create a dataset based on this source.

URLs

Use the following base URLs to manage source files:

```
'http://api.skytree.net:8080/v1/sources'
'http://api.skytree.net:8080/v1/sources/<id>'
'http://api.skytree.net:8080/v1/sources/<id>/sample'
```

Create a Source File

Use one of the following methods to convert raw data into a source file:

- Upload a Local File (page 13)
- Import a File from a URL (page 14)
- Create a Source File by Inputting Raw Data (page 15)
- Create a Source File from a Classification Test File (page 16)
- Create a Source File from a Regression Test File (page 17)

Upload a Local File

CSV files on a local system can be uploaded simply by pointing to that file. Once uploaded, the Skytree API turns the CSV file into a source file.

Request

Use `POST` to upload a local CSV file that you want to convert to a source file. The example below uploads a file named `income.data.with.id`. This file includes a header. The separator in this file is a `,`. A ? indicates missing values in this file.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:multipart/form-data" \
-F name="income.data.with.id" \
```

```
-F file="@/opt/skytree/datasets/income.data.with.id" \
-F "projectId":"24389108236",
   'configuration={
     "hasHeader": true,
     "separator": ",",
     "missingValue": "?"
   }' 'http://api.skytree.net:8080/v1/sources'
```

Response

Upon successful completion, the response returns the unique ID and details of the source file. The response will look similar to the following:

```
200 (OK)
"Content-type:multipart/form-data"
{
 "id":"3980560472",
 "name":"income.data.with.id",
 "createdAt":1412274146288,
 "status":
 {
  "code":"INPROGRESS",
  "message":"Source is being processed"
 },
 "configuration":
 {
  "hasHeader":true,
  "separator":",",
  "missingValue":"?"
 },
 "url":null,
 "type":"Url",
 "projectId":"24389108236"
}
```

Import a File from a URL

Sources files can be imported from a URL. Supported URLs include `http`, `https`, `file` (for example, on the server), and `hdfs` (for example, on a cluster). The type will display in the response.

> **Note:** When specifying a `file` or an `hdfs` path, if the path to the file that you want to register is an absolute path, then be sure to include the beginning "/" (for example, `"url":"hdfs:///path/to/file"`).

Request

Use `POST` to create a source file by importing a CSV file from a URL. Note that the example below includes configuration parameters , which are indicated with the `-d` option.

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE="\
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
```

```
    "name":"income.data.with.id",
    "url":"file:///Users/smith/workspace/income.data.ids",
    "configuration":
    {
     "hasHeader": true,
     "separator": ",",
"missingValue": "?"
    }
   }' 'http://api.skytree.net:8080/v1/sources'
```

Response

Upon successful completion, the response returns the unique ID and details of the source file.

```
200 (OK)
"Content-type:application/json"

{
 "id":"3980560472",
 "name":"income.data.with.id",
 "createdAt":1412274146288,
 "status":
 {
  "code":"INPROGRESS",
  "message":"Source is being processed"
 },
 "configuration":
 {
  "hasHeader":true,
  "separator":",",
  "missingValue":"?"
 },
 "url":"file:///Users/smith/workspace/income.data.ids",
 "type":"Url",
 "projectId":"24389108236"
}
```

Create a Source File by Inputting Raw Data

A source file can be created by inputting raw data in the API request.

Request

Use `POST` to create a source file by inputting values in a data stream. The input values are included in the `configuration` section.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
    "projectId":"24389108236",
    "name": "input",
    "configuration":
    {
     "hasHeader": true,
```

```
     "separator": ",",
     "missingValue": "?"
    },
    "dataPoints":
    {
     "headers":["header","int:2","rating"],
     "data":
     [
      {"point": ["0","473","5"]},
      {"point": ["0","1959","5"]},
      {"point": ["0","946","4"]}
     ]
    }
   }' 'http://api.skytree.net:8080/v1/sources'
```

Response

The response returns the unique source ID along with the source file details.

```
200 (OK)
"Content-type:application/json"
{
 "id":"4144035315",
 "name":"input",
 "createdAt":1412274422546,
 "status":
 {
  "code":"READY",
  "message":"Source is ready to be used"
 },
 "configuration":
 {
  "hasHeader":true,
  "separator":",",
  "missingValue":"?"
 },
 "type":"DataPoints",
 "projectId":"24389108236"
}
```

Create a Source File from a Classification Test File

After a model is trained and tested, create a source from the test output (the predictions), basing this on either the probabilities or objectives output. This allows you to take existing results and further narrow down to a subset of the predictions.

This section describes how to create a source file from a Classification test file. Note that users must specify whether the referenced classification test file is a `Probabilities` file (to source out predicted classification probabilities) or an `Objectives` file (to source out predicted classification labels).

Request

Use `POST` to create a source file by importing a test file. This process requires a test result type defined in the configuration. The example below indicates that test 2089121591 is of type `Objectives`.

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name": "test-from-objective-classification",
     "testResultId": "2089121591",
     "configuration":
     {
      "testResultType":"Objectives"
     }
    }' 'http://api.skytree.net:8080/v1/sources'
```

Response

Upon successful completion, the response returns the unique ID for the new source file.

```
200 (OK)
"Content-type:application/json"

{
 "id":"186564754",
 "name":"test-from-objective-classification",
 "createdAt":1412275232077,
 "status":
 {
  "code":"READY",
  "message":"Source is ready to be used"
 },
 "configuration":
 {
  "testResultType":"Objectives"
 },
 "testResultId":"2089121591",
 "type":"TestResult",
 "projectId":"24389108236"
}
```

Create a Source File from a Regression Test File

After a model is trained and tested, create a source from the regression test file (the predictions), basing this on the objectives output. This allows you to take existing results and further narrow down to a subset of the predictions.

This section describes how to create a source file from a Regression test file. Note that users must specify that the regression test file is an objectives file.

Request

Use POST to create a source file by importing a test file. This process requires the `Objectives` test result type defined in the configuration.

```
curl -X POST -H "x-auth-token: dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
```

```
    "name": "test-from-objective-regression",
    "testResultId": "2196403051",
    "configuration":
    {
     "testResultType":"Objectives"
    }
   }' 'http://api.skytree.net:8080/v1/sources'
```

Response

Upon successful completion, the response returns the unique ID for the new source file.

```
200 (OK)
"Content-type:application/json"

{
 "id":"738972301",
 "name":"test-from-objective-regression",
 "createdAt":1412275622367,
 "status":
 {
  "code":"READY",
  "message":"Source is ready to be used"
 },
 "configuration":
 {
  "testResultType":"Objectives"
 },
 "testResultId":"2196403051",
 "type":"TestResult",
 "projectId":"24389108236"
}
```

Retrieve Sources

You can retrieve information about all sources or for a single source. You can also specify to view a sample of information from a specific source file by specifying the number of top rows to retrieve. Refer to the following sections:

- Retrieve a Single Source (page 18)
- Retrieve All Sources (page 19)
- Retrieve a Sample of Values from a Source (page 19)

Retrieve a Single Source

You can retrieve details about a specific source file by including its unique ID in the API request.

Request

Use GET to retrieve a source file based on its ID. The example below retrieves a source with an ID of 1995278736.

```
curl -X GET -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
```

```
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/sources/1995278736'
```

Response

The response will show information related to the specified source ID.

```
200 (OK)
"Content-type:application/json"
{
 "id":"3980560472",
 "name":"income.data.with.id",
 "createdAt":1403044273249,
 "status":
 {
  "code":"READY",
  "message":"Source is ready to be used"
 },
 "configuration":
 {
  "hasHeader":true,
  "separator":",",
  "missingValue":"?"
 },
 "url":null,
 "type":"Url"
}
```

Retrieve All Sources

This section describes how to retrieve information for all sources.

Request

Use `GET` without any additional parameters to retrieve all sources.

```
curl -X GET -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/sources'
```

Response

The response will show all available sources that were created by the user, with the details of each source included as part of an array.

Retrieve a Sample of Values from a Source

This section describes how to retrieve a subset of values from a source in order to sample the data. Using the `sample` option in the request, the API request includes the top "*n*" rows that you want to retrieve.

Request

Use `GET` to retrieve a list of data points from a source file. The example below retrieves a source with an ID of

3384725639 and shows the top 5 rows of data.

```
curl -X GET -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/sources/3384725639/sample?n=5'
```

Response

The API returns the data that is included in the specified number of rows.

```
200 (OK)
"Content-type:application/json"
{
 "headers":
 [
  "header",
  "int:2",
  "rating"
 ],
 "data":
 [
  {"point":["0","473","5"]},
  {"point":["0","1959","5"]},
  {"point":["0","946","4"]},
  {"point":["0","1024","3"]},
  {"point":["0","106","0"]},
  {"point":["0","1335","2"]},
  {"point":["0","372","0"]},
  {"point":["0","1510","5"]},
  {"point":["0","1719","2"]},
  {"point":["0","94","2"]}
 ]
}
```

Delete a Source

A DELETE request can be used to delete a source file.

Request

Use DELETE to delete a source file. The example below deletes the source file whose ID is 6799717127684506329.

```
curl -X DELETE -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/sources/6799717127684506329'
```

Response

The API returns 204 `No Content` upon successful completion.

Source Parameters

The table below shows parameters that are used in the source creation, retrieval, and delete APIs.

Table 6: *Source API Parameters*

Parameter	Description	Used In
`projectId`	The ID of the project that will contain this source.	Creating sources (all)
`name`	The outputted file name to use when creating a source	Upload a local file, Import a file from a URL
`file`	The name of the file that you want to convert to source. This value is followed by a file path if the source is built from a local file. It is followed by a URL if the source is built from a file imported from a URL.	Import a file from a URL
`id`	The ID of a generated source file	Retrieve a source, Retrieve sample values from a source, Delete a source
`n`	The number of rows to retrieve from a specified source file	Retrieve sample values from a source
`testResultId`	The ID of the test output file that you want to use as the basis for the new source.	Creating a source file from a test result
`testResultType`	When creating source from a classification test result, specify whether the file is `Probabilities` or `Objectives`. When creating source from a regression test result, specify that the file type is `Objectives`.	Creating a source file from a test result
`configuration`	Specifies whether any additional source configurations are required. Configuration options include the following: `hasHeader` - Specifies whether the source includes a header file. This value defaults to TRUE. Note that column headers can be a maximum of 255 characters. `separator` - Specifies the character to use when separating values in the source. This value defaults to ",". `missingValue` - Specifies the token that is used to represent missing values. `dataPoints` - Specifies the raw data to be used when creating a source. When entering data points you can specify headers to be included in the source.	Upload a local file, Import a file from a URL, Creating source from raw data
`url`	The path to use when import files from a URL.	Import a file from a URL

Source Return Fields

A successfully created source response will output with the following properties. Note that these can vary depending on how the source file was created.

Table 7: *Source Return Properties*

Parameter	Description
`name`	The outputted file name used when creating a source
`id`	The ID of a generated source file
`createdAt`	The epoch time when the source was created
`updatedAt`	The epoch time when the source was updated
`status`	The Skytree status (for example, READY) and a message (for example, "Source has been validated and is ready to use.")
`configuration`	Specifies whether any additional source configurations were used when creating the source. Configuration options include the following: `hasHeader` - Specifies whether the source includes a header file. `separator` - Specifies the character used when separating values in the source. `missingValue` - Specifies the token used to represent missing values. `testResultType` - If the origin type was `TestResult`, then this shows whether the result type was `Probabilities` or `Objectives` for Classification tests or just `Objectives` for Regression tests.
`url`	The path used when importing files from a URL
`type`	Shows the origin of the source: `URL` - The origin used for the source was a file referenced by a URL. `DataPonits` - The origin used for the source was a set of data points inputted directly into a request. `TestResult` - The origin used for the source was a test file.
`testResultId`	The ID of the classification or regression test that was used when the source was created

Chapter 5 Datasets

A dataset represents information in tabular form that has been imported into Skytree. Currently, datasets support several types of columns: numerical, categorical, and unique ID columns.

The Dataset APIs transform the source data into a dataset. Upon submission of the create request, the Status is set to INPROGRESS. As soon as the dataset is created, Skytree attempts to validate the dataset. If successful, the Status changes to READY and returns the unique dataset ID.

URLs

Use the following base URLs to manage datasets:

```
'http://api.skytree.net:8080/v1/datasets'
'http://api.skytree.net:8080/v1/datasets/<id>'
'http://api.skytree.net:8080/v1/datasets/<id>/interrupt'
'http://api.skytree.net:8080/v1/datasets/<id>/dependents'
'http://api.skytree.net:8080/v1/datasets/<id>/counts?numberOfBins=<num>'
'http://api.skytree.net:8080/v1/datasets?projectId=<project_id>'
```

Dataset Substatus Codes

After a dataset has been created and its status set to READY, the dataset show an additional substatus (or phase) as described in the following table.

Table 8: *Dataset Substatus Codes*

Substatus Code	Description
`SampleCreated`	Sample data has been created, and a header file has been generated for the sample data.
`Transformed`	The source data has been transformed.
`Converted`	The data has gone through the conversion process and is available in Skytree format.

Create a New Dataset

New datasets can be created by converting an existing source file or by transforming an existing dataset.

Note: When specifying datasets in transforms, models, and test results, be sure that the dataset resides in the current project. A single dataset ID cannot exist in multiple projects.

Request

Use `POST` to create a new dataset from a source file. The following example creates a dataset from a source file with an ID of `2555351788` and specifies that the target column (`idColumn`) is `AnnualIncome`.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["income_data_set"],
     "sourceId":"2555351788",
     "configurations":
     [{
      "idColumn":"AnnualIncome"
     }]
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion, the response returns the unique ID and details about all columns in the dataset. The response will look similar to the following:

```
200 (OK)
"Content-type:application/json"
[
"projectId":"24389108236"
 {
 "id":"234971499",
 "names":["income-data-set"],
 "createdAt":1403045348903,
 "updatedAt":1403045349066,
 "status":
 {
  "code":"READY",
  "message":"Data Set has been validated and is ready to use"
 },
 "sourceId":"2555351788",
 "configurations":
 [{
  "idColumn":"AnnualIncome"
 }],
 "transform":
 {
  "inputs":null,
  "transformSteps":null,
 },
 "columns":
 [
  {
   "id":1,
   "name":"age",
   "type":"Integer",
```

```
   "dictionary":null,
   "min":17.0,
   "max":90.0,
   "mean":38.051,
   "sigma":13.349479675538191,
   "selected":true,
   "categorical":false
  },
  {
   "id":2,
   "name":"workclass",
   "type":"Text",
   "dictionary":
   [
    "self-emp-inc",
    "federal-gov",
    "local-gov",
    "state-gov",
    "self-emp-not-inc",
    "?",
    "private"
   ],
   "min":0.0,
   "max":0.0,
   "mean":0.0,
   "sigma":0.0,
   "selected":true,
   "categorical":true
  },
  {
   "id":3,
   "name":"fnlwgt",
   "type":"Integer",
   "dictionary":null,
   "min":21174.0,
   "max":1033222.0,
   "mean":191904.978,
   "sigma":108125.54441474265,
   "selected":true,
   "categorical":false
  },
...
 ]
"projectId":"24389108236"
}]
```


Commit a Dataset

In most cases, column metadata is generated from a sampling of a dataset during machine learning. Another option is to commit the dataset before performing any machine learning methods. Committing datasets allows characterization of the full dataset. This includes the type information, dictionary, and summary statistics for each column.

Request

Use POST to commit a dataset. The dataset ID is required. In the example below, a dataset with an ID of 2502809161

is committed.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{ "projectId":"24389108236" }' \
'http://api.skytree.net:8080/v1/datasets/2502809161'
```

Response

If the specified dataset ID exists, then the response returns information for the dataset. The output will be similar to the output from the dataset creation process. Refer to *Dataset Return Fields* (page 30) for more information.

Retrieve a Dataset

Use GET to retrieve a specific dataset. The dataset ID is required.

Request

In the example below, a dataset with an ID of 2502809161 is retrieved. Note again that GET is the default behavior.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/datasets/2502809161'
```

Response

If the specified dataset ID exists, then the response returns information for the dataset. The output will be similar to the output from the dataset creation process. Refer to *Dataset Return Fields* (page 30) for more information.

Retrieve All Datasets

If a dataset ID is not specified in a dataset retrieval request, then all datasets from all projects will be returned.

Request

Use GET without an ID to retrieve all available datasets. Note that GET is the default behavior and, thus, is not required.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/datasets'
```

Response

The response returns information for all datasets grouped by Dataset ID.

Retrieve All Datasets in a Project

Use GET along with a project ID to retrieve all available datasets in a project.

Request

The following example is a request to retrieve all datasets in a project whose ID is24389108236. Note that GET is the

default behavior and, thus, is not required.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/datasets?projectId=24389108236'
```

Response

The response returns information for all datasets in the specified project grouped by Dataset ID.

Retrieve Dataset Dependents

Use this function to get a list of all downstream (dependent) files related to a dataset. This includes all models, test results, etc, that were created from the dataset.

Request

Use GET along with a dataset ID to retrieve all files dependent on the dataset. The request below shows how to retrieve a list of files related to a dataset whose ID is 4274348704094814824.

```
curl -X GET -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/datasets/4274348704094814824/dependents'
```

Response

Upon successful completion, the response returns a list of files that are dependent on this dataset.

```
200 (OK)
"Content-type:application/json"

{
 "id":"4274348704094814824",
 "name":"income.data",
 "type":"DATASET",
 "is_deletable":TRUE
},
{
 "id":"4426696063906522815",
 "name":"income.data.ids",
 "type":"DATASET",
 "is_deletable":TRUE
},
{
 "id":"8613266560949776288",
 "name":"GBT Model",
 "type":"MODEL",
 "is_deletable":TRUE
},
{
 "id":"733579628360679490",
 "name":"GBT Result",
 "type":"TESTRESULT",
 "is_deletable":TRUE
}
```

Retrieve Column Statistics

This API call retrieves column statistics for a specified dataset. In this request, the dataset ID and `numberOfBins` are required. You can optionally specify `column="<column_name>"` to retrieve stats for a specific column.

Request

Use `GET` to retrieve column statistics for a dataset. The dataset ID is required as is `numberOfBins`. In the example below, column statistics for a dataset with an ID of 2502809161 is retrieved. The number of bins is specified as 10. Note again that `GET` is the default behavior.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json"
'http://api.skytree.net:8080/v1/datasets/2502809161/counts?numberOfBins=10'
```

Response

The response provides count information for all columns or for a single column if one was specified in the request.

```
{
  "status": {
  "code": "READY"
},
  "numberOfBins": 10,
  "counts": [
    { "name": "X",
      "type": "numeric",
      "min": 0,
      "max": 100,
      "bins": [
        {"count": 19, "range": "[0,10)"}
        {"count": 22, "range": "[10,20)"}
        {"count": 38, "range": "[20,30)"}
        {"count": 41, "range": "[30,40)"}
        {"count": 52, "range": "[40,50)"}
        {"count": 46, "range": "[50,60)"}
        {"count": 33, "range": "[60,70)"}
        {"count": 21, "range": "[70,80)"}
        {"count": 11, "range": "[80,90)"}
        {"count": 10, "range": "[90,100]"}
      ],
      "totalCount": 1000,
      "missingValueCount", 5
    },
    { "name": "Y",
      "type": "categorical",
      "bins": [
        {"count": 100, "category": "K"},
        {"count": 90, "category": "F"},
      ],
      "totalCount": 1000,
      "missingValueCount": 10
    },
    ...
  ]
}
```

Interrupt a Dataset

You may find it necessary to interrupt a dataset that is currently INPROGRESS. Note that you cannot interrupt datasets that have a status of READY.

Request

Use POST to interrupt the process running on a dataset. The dataset ID is required. In the example below, a dataset with an ID of 234971499 is interrupted.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http:/api.skytree.net:8080/v1/datasets/234971499/interrupt'
```

Response

The API returns "`true`" if the interrupt completed successfully. Otherwise the API returns "`false`" (for example, if the dataset was already created).

Delete a Dataset

A DELETE request can be used to delete a dataset along with all downstream files that were generated from this dataset (for example, models and results).

> **Note:** You cannot delete datasets that were built from test results.

Request

Use DELETE to delete a dataset. The dataset ID is required. In the example below, a dataset with an ID of 6799717127684506329 is deleted.

```
curl -X DELETE -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/datasets/6799717127684506329'
```

Response

The API returns 204 `No Content` upon successful completion.

Dataset Parameters

The following table summarizes the parameters that are used in the Dataset APIs.

Table 9: *Dataset Parameters*

Parameter	Description	Used In
`projectId`	The ID of the project that will contain this dataset.	Create dataset, Commit dataset, Retrieve dataset from specific project

Table 9: *Dataset Parameters (continued)*

Parameter	Description	Used In
`names`	Specify a name for the dataset that you're creating. Users specify a new name when creating a new dataset from a source file and when transforming an existing dataset.	Create dataset
`sourceId`	The ID of the source file on which to base the new dataset. This is not required when transforming a dataset.	Create dataset
`<id>`	The unique value of an existing dataset	Commit dataset, Retrieve dataset, Retrieve dataset dependents, Retrieve column statistics, Delete dataset, Interrupt dataset
`numberOfBins`	The top N categories to be returned	Retrieve column statistics
`column`	Optionally specify to return counts for a specific column. If not specified, then counts are returned for all columns.	Retrieve Column Statistics
`configuration`		
`idColumn`	When a dataset will be used to test a model, specify the column name to be used as the index. If your training dataset already contains an ID Column, then that column must be specified with this option. Users will receive an error if they attempt to test a dataset that does not include this configuration setting.	Create dataset

Note: A `transform` parameter is also available. Dataset transformations are described in *Chapter 6 Transformations* on page 33.

Dataset Return Fields

A successfully generated dataset will output with the following properties:

Table 10: *Dataset Return Properties*

Return Field	Description
`id`	The ID of the dataset that was just created
`name`	The name of the dataset
`createdAt`	The epoch time when the dataset was created
`updatedAt`	The epoch time when the dataset was updated
`status`	The Skytree status (for example, READY) and a message (for example, "Data Set has been validated and is ready to use.")

Table 10: *Dataset Return Properties (continued)*

Return Field	Description
`sourceId`	The ID of the source that was used to create this dataset if created from a source.
`transform`	Includes "`null`" values if the dataset was newly created.
`Columns`: The fields below describe each column in the dataset.	
`id`	The column ID
`name`	The column name
`type`	The column type
`dictionary`	This value is `null` if the column is numerical. Otherwise (if categorical), this shows all the unique values in the column.
`min`	For numeric values, this shows the minimum value of the column. For text, this shows `0.0`.
`max`	For numeric values, this shows the maximum value of the column. For text, this shows `0.0`.
`mean`	For numeric values, this shows the mean value of the column. For text, this shows `0.0`.
`sigma`	For numeric values, this shows the sigma value of the column. For text, this shows `0.0`.
`selected`	"`true`" if the column is selected when the dataset was being used; "`false`" if not
`categorical`	"`true`" if the data is categorical; "`false`" if the data is not categorical
`Configurations`: The fields below describe the `Configurations` section in a dataset.	
`blobColumns`	An array of column names. Dictionaries are not currently created for these columns.
`idColumn`	The ID of the column used as the target column.

Chapter 6 Transformations

After you create a dataset, Skytree allows you to apply transformations to the dataset for feature engineering.

> **Note:** When applying transformations to a dataset, the original dataset and the transformed dataset must reside in the same project.

Transform Types

Below are the currently available transform types.

- **Add Unique ID Column** - ID columns are required for datasets that will be used to test models. This transform allows you to specify an existing column in a dataset and set it as the ID column.
- **Categorize** - This operation converts categorical data into a form acceptable for machine learning by treating each entry in a column as text and giving each unique entry a distinct value or by treating each entry in a column as part of a numeric category.
- **Clamp Values** - Given a set of minimum/maximum values, this transform clamps values that are beyond the specified range.
- **Extract Text Features** - Extracts text features in a specified blob column. This can only be performed after a dataset has been transformed using Ingest Self Contained Text or Ingest Text.
- **Filter Rows** - Datasets can be filtered by rows to include a subset of data. The filter is based on a given Lambda expression.
- **Horizontalize** - This transforms categorical data into a form acceptable for machine learning by creating a new column for each different value that a categorical column can take. A value of `0.707` is assigned to the right column, and the columns for all other values get 0.
- **Ingest Self Contained Text** - Transforms a dataset by specifying the blob column or columns in a text file.
- **Ingest Text** - For datasets that contain a list of URLs, this transforms the dataset by ingesting the text the from URL links.
- **Join** - Combine portions of two datasets into a single dataset. You can perform an Inner Join, Right Outer Join, or Left Outer Join.
- **New Column** - This adds a new column to a dataset and populates the data based on a supplied Lambda expression.
- **Normalize** - Normalize columns containing numeric data.
- **Remove Columns** - Datasets can be also be filtered to exclude certain columns based on the column name.

- **Split** - Split an existing dataset into multiple datasets.
- **Tokenize Text** - Tokenizes the blob column in a dataset.
- **Vectorize Text** - Vectorizes the features in a dataset. This transform requires a dataset that includes a categorized label column and a dataset with extracted text features.

Note: During transformations, columns that include only missing values will be set a categorical type "text" and will have an empty dictionary. The transformation will then complete with an error indicating that a dictionary is not available for the column and that continuing any machine learning with the dataset is not possible.

Note: Due to constraints with Spark, you cannot chain more than 400 total operations when transforming a dataset. This constraint applies to the entire life cycle of a dataset (i.e., from source, through multiple transforms, to final dataset). A workaround is to create an intermediate dataset and continue additional transform steps from there on.

You can also apply multiple transforms to a dataset. Refer to *Specifying Multiple Transforms* (page 56) for more details. Finally, Skytree accepts scripts that you can specify in custom transform. Refer to *Custom Transforms* (page 61) for more information.

The sections that follow provide additional information about each of the above transforms.

URLs

Use the following base URLs when transforming datasets:

```
'http://api.skytree.net:8080/v1/datasets'
```

Add Unique ID Column

This transform transform adds an ID column as the last column of a dataset. This column serves as a row identifier. The new column is guaranteed to be unique but not necessarily sequential.

Add Unique ID Column Transform Parameters

The Add Unique ID Column transform accepts the following parameters:

Table 11: *Add Unique ID Column Transform Parameters*

Parameter	Description
`columnName`	Specify a name for the new column.

Request

Use `POST` to add a new column to an existing dataset. The example below adds a new column named `IDs` to a dataset whose ID is 298952448.

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-Type:application/json" \
-d '{
```

```
    "projectId":"24389108236",
    "names":["income-data-ids"]
    "sourceId":null,
    "transform":
    {
     "inputs":["298952448"],
     "transformSteps":
     [
      {
       "type":"AddUniqueIdColumn",
       "columnName":"IDs",
      }
     ]
    }
   }' 'http://api.skytree.net:8080/v1/datasets/'
```

Response

```
{
 "id":"867431201",
 "name":"income-data-ids",
 "createdAt":1412101348351,
 "updatedAt":1412101348351,
 "status": {
   "code": "INPROGRESS",
   "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform": {
   "inputs": ["298952448"],
   "transformSteps": [
    {
      "type":"AddUniqueIdColumn",
      "columnName":"IDs",
   }
  ],
  "blobColumns": null,
  "subStatus": null
 },
 "columns": null,
 "projectId":"24389108236"
}
```

Categorize Datasets

The Categorize transform API allows you to convert categorical data into a form acceptable for machine learning by replacing the categorical value in the specified column with a unique number identifying the value.

Consider example data as follows:

```
0,1,3
00,4,5
10,2,7
```

By default, the data will be treated as a continuous value.

```
0.0,1,3
0.0,4,5
10.0,2,7
```

In some cases, you might want this to be treated as categorical (for example, if the above are status codes). If `"categorizationType":"text"` is specified, then each entry is treated as text, and each unique entry would be a distinct value. So 0, 00, and 10 above would be replaced with 1, 2, 3 (a unique value for each unique entry). Note that "00" is no longer treated the same as "0."

```
1,1,3
2,4,5
3,2,7
```

If `"categorizationType":"number"` is specified, then each entry is treated as part of a numeric category. With numeric categories, for example, `1.0`, `1.00`, and `01` are considered to be in the same category. In the above example, 0 and 00 would be in the same category, so the output would be similar to the following:

```
1,1,3
1,4,5
2,2,7
```

Note: For columns with more than 2048 categorical values, the column is converted to non-categorical text in order to limit the dictionary size.

Categorize Parameters

Categorize transforms accept the following parameters:

Table 12: *Categorize Transform Parameters*

Parameter	Description
`categorizationType`	Specify the type of data to categorize. `categorizationType` is a 1-based index. Accepted values include: • `Number` • `Text`
`categorizationColumn`	Specify the column name to be used during the transformation. If the dataset does not contain headers, columns are named "`Column_<index>`" where `<index>` is a 1-based index.

Request

Use `POST` to transform a new dataset from an existing dataset. Unlike in the dataset creation process, the Source ID is not required. Only the ID of the dataset that you want to use as the basis for the transform ("`inputs`") is required. In the example below, a dataset with an ID of `907137553` is specified. As a result, all entries in the second column will be treated as part of a numeric category.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
```

```
    "names":["Categorized_Dataset"],
    "sourceId":null,
    "transform":
    {
     "inputs":["907137553"],
     "transformSteps":
     [
      {
       "categorizationType":"Number",
       "categorizationColumn":"Column_2",
       "type":"Categorize"
      }
     ]
    }
   }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion of the transform, the API returns a new dataset. The response will be similar to below. Refer to *Transform Return Fields* (page 57) for information about the fields that display in this response.

```
200 (OK)
"Content-type:application/json"

{
 "id": "1003277440",
 "name": "Categorized_Dataset",
 "createdAt": 1401161445310,
 "updatedAt": 1401161445310
 "status":
 {
  "code": "INPROGRESS",
  "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform":
 {
  "inputs": ["907137553"],
  "transformSteps":
  [
   {
    "categorizationType": "Number",
    "categorizationColumn": "Column_2",
    "type": "Categorize"
   }
  ],
  "blobColumns": null,
  "subStatus": null
 },
 "columns": null,
 "projectId":"24389108236"
}
```

Clamp Values

The Clamp Values transform allows you to substitute a new value for values that are out of a specified range. The new value will be either the specified maximum value or the specified minimum value, depending on whether it falls above or below the range.

For example, given a set of data with columns A and B as below:

```
A,B
1,20
2,5
3,0
4,-5
5,-20
```

A clamp transform that specifies minimum and maximum values of column B as -10 and 10, respectively, will transform the data as follows:

```
A,B
1,10
2,5
3.0
4,-5
5,-10
```

Notice that the values that fall outside of the -10,10 range are clamped with these updated values.

Clamp Values Parameters

The Clamp Values transform accepts the following parameters:

Table 13: *Clamp Values Transform Parameters*

Parameter	Description
clampColumn	Specify the column name or number in the header to be considered when clamping values.
maxValue	When clamping values, specify the maximum value to be included in the transform. Values in the header that are larger than this value will be represented with this value instead.
minValue	When clamping values, specify the minimum value to be included in the transform. Values in the header that are smaller than this value will be represented with this value instead.

Request

Use POST to transform a new dataset using clamping. In the example below, a dataset with an ID of 32184553 is specified. Column 2 is specified as the column that includes the values to be clamped. The maximum and minumum values for this column are specified as 100 and 0.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
```

```
    "projectId":"24389108236",
    "names":["My_Clamped_Dataset"],
    "sourceId":null,
    "transform":
    {
     "inputs":["32184553"],
     "transformSteps":
     [
      {
       "clampColumn":2,
       "maxValue":"100",
       "minValue":"0",
       "type":"ClampValues"
      }
     ]
    }
   }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion of the transform, the API returns a new dataset. The response will be similar to below. Refer to *Transform Return Fields* (page 57) for information about the fields that display in this response.

```
200 (OK)
"Content-type:application/json"

{
 "id": "1029487442",
 "name": "My_Clamped_Dataset",
 "createdAt": 2014161445310,
 "updatedAt": 201461445310
 "status":
 {
  "code": "INPROGRESS",
  "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform":
 {
  "inputs": ["32184553"],
  "transformSteps":
  [
   {
    "clampColumn": "2",
    "maxValue": "100",
    "minValue": "0",
    "type": "ClampValues"
   }
  ],
  "blobColumns": null,
  "subStatus": null
 },
 "projectId":"24389108236",
 "columns": null
}
```

Extract Text Features

This transforms a dataset by extracting the text features in a specified blob column. These features can be extracted using:

- Term Frequency - Inverse Document Frequency (tf-idf): The IDF value increases proportionally to the number of times a word shows up in the dataset and also diminishes the weight of terms that occur frequently (such as "the" or "and").
- Term Frequency - Bi-Normal Separation (tf-bns): Computes the BNS score for each feature by scaling the magnitude of feature values

Users can also optionally specify the minimum feature weight and the maximum feature size when transforming the dataset.

The resulting dataset will include the featurized words in a new Token column and the weight given to each word in a new Weight column. This new dataset can then be referenced in a Vectorize Text transform. (See Vectorize Text (page 56).)

Extract Text Features Parameters

The Extract Text Features transform accepts the following parameter:

Table 14: *Extract Text Features Transform Parameters*

Parameter	Description
`targetBlobColumn`	Specify the blob column to be featurized in the new dataset
`labelColumnName`	Specify the label column that was set as categorical text in the Set as Catetorical Text transform.
`featureWeightingScheme`	Specify whether to featurize using inverse document frequency (`tf_idf`) or bi-normal separation (`tf_bns`).
`minFeatureWeight`	Optionally specify the minimum weight to be given to a word.
`maxFeatureSize`	Optionally specify the maximum size of a word

Request

Use `POST` to extract text features in a dataset. In the example below, a dataset with an ID of 907137553 is specified. This is the dataset that includes the blob column. Another input 83224872, includes the labels column that was specified as categorical text. The features are extracted using `tf-idf`.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["Extracted Text Features"]
     "sourceId":null,
     "transform":
     {
      "inputs":["907137553","83224872"],
      "transformSteps":
      [
```

```
      {
       "targetBlobColumn":"doc",
       "labelColumnName":"label",
       "featureWeightingScheme":"td-idf",
       "type":"TextFeatureExtraction"
      }
     ]
    }
   }' 'http://api.skytree.net:8080/v1/datasets'
```

Filter Rows

The Filter Rows transform allows you to filter rows in datasets to include only a subset of the original data in the new, transformed dataset. Filtering rows evaluates a Lambda expression (see also Lambda Expressions in The Python Tutorial) on each row of data. The Lambda expression must return a Boolean value. If the returned value is True, then the row is included in the new dataset. If the returned value is False, then the row is filtered out.

For example, given a dataset with a text-type column named "color" the following Lambda expression can be used to filter the data:

This will include only rows where color is red.

```
"lambda color: color == 'red'"
```

This will include all rows except red rows.

```
"lambda color: color != 'red'"
```

Filter Parameters

The Filter Rows transform accepts the following parameter:

Table 15: *Filter Rows Transform Parameters*

Parameter	Description
`filterExpression`	Evaluates a Lambda expression on each row of data.

Request

Use `POST` to transform a new dataset by filtering an existing dataset. In the example below, a dataset with an ID of `907137553` is specified. This transform generates a dataset that remove rows in the `workclass` column that include the `Private` value.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["Filtered_Dataset"],
     "sourceId":null,
     "transform":
     {
      "inputs":["907137553"],
```

```
     "transformSteps":
     [
      {
       "filterExpression":"lambda workclass: workclass != 'Private'"",
       "type":"Filter"
      }
     ]
    }
   }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion of the transform, the API returns a new dataset. The response will be similar to below. Refer to *Transform Return Fields* (page 57) for information about the fields that display in this response.

```
200 (OK)
"Content-type:application/json"

{
 "id": "1003277442",
 "name": "Filtered_Dataset",
 "createdAt": 1401161445310,
 "updatedAt": 1401161445310
 "status":
 {
  "code": "INPROGRESS",
  "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform":
 {
  "inputs": ["907137553"],
  "transformSteps":
  [
   {
    "filterExpression": "lambda workclass: workclass != 'Private'",
    "type": "Filter"
   }
  ],
  "blobColumns": null,
  "subStatus": null
 },
 "projectId":"24389108236",
 "columns": null
}
```

Horizontalize Datasets

This transform creates a new column for each different value that a categorical column can take. A value of 0.707 is assigned to the right column, and the columns for all other values get 0.

Consider the following example data:

```
blue,1,3
red,4,5
```

```
blue,2,7
...
```

When horizontalize is applied to the first column, the following transformation take place (assuming that column 1 can only take values of blue and red):

```
0.7071067811865475,0,1,3
0,1,4,5
0.7071067811865475,0,2,7
```

Horizontalize Parameters

Horizontalize transforms accept the following parameter:

Table 16: *Horizontalize Transform Parameters*

Parameter	Description
`horizontalizeColumn`	Specify the column name that contains the data to be horizontalized.
`horizontalizeDictionary`	Optionally specify a comma-separated list of keys to be used when testing against a model trained on a different dataset. This option allows you to force the transform to use the specified dictionary instead of generating it from the column.

Request

Use `POST` to horizontalize an existing dataset. In the example below, a dataset with an ID of 907137553 is specified, and a column named "`column2`" is the targeted column.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["Horizontalize"]
     "sourceId":null,
     "transform":
     {
      "inputs":["907137553"],
      "transformSteps":
      [
       {
        "horizontalizeColumn":"column2",
        "type":"Horizontalize"
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion of the transform, the API returns a new dataset. The response will be similar to below. Refer to *Transform Return Fields* (page 57) for information about the fields that display in this response.

```
200 (OK)
"Content-type:application/json"
```

```
{
 "id": "1003277443",
 "name": "Horizontalize",
 "createdAt": 1401161445310,
 "updatedAt": 1401161445310,
 "status":
 {
  "code": "INPROGRESS",
  "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform":
 {
  "inputs": ["907137553"],
  "transformSteps":
  [
   {
    "horizontalizeColumn": "column2",
    "type": "Horizontalize"
   }
  ],
  "blobColumns": null,
  "subStatus": null
 },
 "projectId":"24389108236",
 "columns": null
}
```


Ingest Self Contained Text

This transform defines the blob column (column containing text) in a dataset to be ingested in a new dataset. This transform can only be used with datasets that have one or more blob columns already defined. The resulting dataset will include a new Error column for rows that encounter errors during the transform. If no errors exist, then this transformed dataset will be ready for tokenization. (See Tokenize Text (page 55).)

Ingest Self Contained Text Parameters

The Ingest Self Contained Text transform accepts the following parameter:

Table 17: *Ingest Self Contained Text Transform Parameters*

Parameter	Description
`targetBlobColumn`	Specify the column in the dataset that includes the blob (free form) text.

Request

The request below uses `POST` to define a blob column in a dataset whose ID is 907137553. This is the dataset that includes the blob column named "doc".

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
```

```
    "projectId":"24389108236",
    "names":["Ingest Self Contained Text"]
    "sourceId":null,
    "transform":
    {
     "inputs":["907137553"],
     "transformSteps":
     [
      {
       "targetBlobColumn":"doc",
       "type":"IngestSelfContainedFormattedDoc"
      }
     ]
    }
   }' 'http://api.skytree.net:8080/v1/datasets'
```

Ingest Text

For datasets that contain a list of URLs, this transforms the dataset by ingesting the text the from the URL links in the new dataset. This transform can only be used with datasets that have one or more blob columns already defined. The resulting dataset will include a new Error column for rows that encounter errors during the transform. If no errors exist, then this transformed dataset will be ready for tokenization. (See Tokenize Text (page 55).)

Ingest Text Parameters

The Ingest Text transform accepts the following parameter:

Table 18: *Ingest Text Transform Parameters*

Parameter	Description
`targetBlobColumn`	Specify the column in the dataset that includes the blob (free form) text.

Request

The request below uses `POST` to define a blob column in a dataset whose ID is 9071846553. This dataset includes URLs that point to a variety of PDF files. The blob column named containing these URLs is named "doc".

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
    "projectId":"24389108236",
    "names":["Ingest Formatted Text"]
    "sourceId":null,
    "transform":
    {
     "inputs":["9071846553"],
     "transformSteps":
     [
      {
       "targetBlobColumn":"doc",
       "type":"IngestFormattedDoc"
      }
```

```
    ]
   }
}' 'http://api.skytree.net:8080/v1/datasets'
```

Join Transforms

The Join transforms allow you to join portions of two datasets and combine those into a single dataset. For example, you might have two datasets that include household income information, and you want to build models based solely on gender information. You can use the Join transforms to join the gender columns and combine those into a single dataset.

> **Note:** Joins are only supported on two datasets in a single transform. Create multiple transforms to join more than two datasets.

The following types of Join transforms are available:

- `InnerJoin` - Combines like entries from two datasets into a single dataset.
- `RightOuterJoin` - Adds values from Dataset1 into Dataset2. Every row from Dataset2 will appear in the joined table at least once. If a matching row from Dataset1 does not exist, then those values will be empty (or NULL).
- `LeftOuterJoin` - Adds values from Dataset2 into Dataset1. The resulting dataset returns all values from an inner join plus values in Dataset1 that do not match Dataset2, including rows with NULL values

Join Transform Parameters

The Join transforms accept the following parameters:

Table 19: *Join Transform Parameters*

Parameter	Description
`joinColumns`	Specify the column name in each file that you want to join.

`type`	The following join types are available:
	`InnerJoin`
	`LeftOuterJoin`
	`RightOuterJoin`

Request

Use `POST` to join columns from two datasets. The following example joins the "age" columns from datasets 3794113559 and 3090760026. The type is specified as `RightOuterJoin`.

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-Type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["join_1],
```

```
    "sourceId":null,
    "transform":
    {
     "inputs":["3794113559","3090760026"],
     "transformSteps":
     [
      {
       "joinColumns":["age","age"],
       "type":"RightOuterJoin"
      }
     ]
    }
   }' 'http://api.skytree.net:8080/v1/datasets/'
```

Response

```
{
 "id": "3794113559",
 "name": "join_1",
 "createdAt": 1410793054869,
 "updatedAt": 1410793054869,
 "status": {
   "code": "INPROGRESS",
   "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform":
 {
  "inputs": ["2754676572"],
  "transformSteps":
  [
    {
      "joinColumns": ["age","age"],
      "type": "RightOuterJoin"
    }
  ],
  "blobColumns": null,
  "subStatus": null
 },
 "projectId":"24389108236",
 "columns": null
}
```

New Column Transform

Use the New Column transform to create a new column based on an existing column. After specifying a column name and the Lambda expression (see also Lambda Expressions in The Python Tutorial) to use when calculating values for the new column, the new column will appear as the last column in the dataset. The data within this new column can then be manipulated while information from the original column remains intact.

New Column Transform Parameters

The New Column transform accepts the following parameters:

Table 20: *New Column Transform Parameters*

Parameter	Description
newColumnName	Specify a name for the new column. The column name length cannot exceed 255 characters.
newColumnExpression	The Lambda expression that will calculate the value of the new column, such as "lambda yearly-income: yearly-income + 1000". This can include constants or variables. A variable must represent a column name.

Request

Use POST to add a new column to an existing dataset. The example below adds a new column named age_plus_20 to a dataset whose ID is 298952448.

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-Type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["newcolumn_1"]
     "sourceId":null,
     "transform":
     {
      "inputs":["298952448"],
      "transformSteps":
      [
       {
        "type":"NewColumn",
        "newColumnName":"age_plus_20",
        "newColumnExpression":"lambda age: age + 20"
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets/'
```

Response

```
{
 "id":"867431201",
 "name":"newcolumn_1",
 "createdAt":1412101348351,
 "updatedAt":1412101348351,
 "status": {
   "code": "INPROGRESS",
   "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform": {
   "inputs": ["298952448"],
   "transformSteps": [
    {
      "type":"NewColumn",
      "newColumnName":"age_plus_20",
      "newColumnExpression":"lambda age: age + 20"
    }
  ],
```

```
  "blobColumns": null,
  "subStatus": null
 },
 "projectId":"24389108236",
 "columns": null
}
```

Normalize Columns

The normalize functions are only applicable to columns containing numeric data. These allow you to normalize columns containing numeric data. Two types of normalization are available:

- **Unit**: his form makes the data range between 0 and 1. Min and max values can be provided when performing a Unit normalization. When these are provided, the column is recalculated using [(*value* - *min_value* / *max_value*].
- **Standard**: This gives the data a mean value of 0 and unit variance. Standard deviation and mean values can be provided when performing Standard normalization. When these are provided, the column is recalculated using [(*value* - *stdev_value* / *mean_value*].

Normalize Parameters

The Normalize transform accepts the following parameters:

Table 21: *Normalize Transform Parameters*

Parameter	Description
`normalizeColumn`	Specify the column to be normalized. Note that only columns that contain numeric data can be normalized.
`normalizationType`	Specify one of the following: • `Unit`: This makes the data range between 0 and 1. • `Standard`: This gives the data a mean value of 0 and unit variance.
`max`	When normalizing , optionally specify the maximum value to be included in the transform. Values in the header that are larger than this value will be represented with this value instead. Note that this cannot be used with `mean` or `stdev`.
`min`	When normalizing, optionally specify the minimum value to be included in the transform. Values in the header that are smaller than this value will be represented with this value instead. Note that this cannot be used with `mean` or `stdev`.
`mean`	When normalizing, optionally specify the mean value to be used in the transform. This option must be used with `stdev` and cannot be used with `max` or `min`.
stdev	When normalizing based on a `mean` value, specify the standard deviation value to be used in the transform. This cannot be used with `max` or `min`.

Request

Use `POST` to transform a dataset using normalization. In the example below, a dataset with an ID of 94284546 is specified. Column 2 is specified as the column that includes the values to be normalized. The normalization type is specified as `Unit`. The maximum and minimum values to be included in the transform are 100 and 0.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["Normalized_Dataset"],
     "sourceId":null,
     "transform":
     {
      "inputs":["94284546"],
      "transformSteps":
      [
       {
        "normalizationType":"Unit",
        "normalizationColumn":"2",
        "max":"100",
        "min":"0",
        "type":"Normalize"
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion of the transform, the API returns a new dataset. The response will be similar to below. Refer to *Transform Return Fields* (page 57) for information about the fields that display in this response.

```
200 (OK)
"Content-type:application/json"

{
 "id": "6582826995",
 "name": "Normalized Dataset",
 "createdAt": 201416237310,
 "updatedAt": 201416237310,
 "status":
 {
  "code": "INPROGRESS",
  "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform":
 {
  "inputs": ["94284546"],
  "transformSteps":
  [
   {
    "normalizationColumn": "2",
    "normalizationType": "Unit",
    "maxValue":"100",
    "minValue":"0",
    "mean":null,
    "stdev":null,
    "type": "Normalize"
   }
  ],
```

```
  "blobColumns": null,
  "subStatus": null
 },
 "projectId":"24389108236",
 "columns": null
}
```

Remove Columns

The Remove Columns transform excludes certain columns based on the column name. Only datasets that have header rows (column names) can be referenced in this transform. Note that the "removeColumns" and "type" expressions are required for each column that you want to filter.

Remove Columns Parameters

The Remove Columns transform accepts the following parameter:

Table 22: *Remove Columns Transform Parameters*

Parameter	Description
removeColumns	Specify a single column name or a list of column names that you want to remove from the transformed dataset.

Request

Use POST to transform a new dataset by removing a column or columns from an existing dataset. In the example below, a dataset with an ID of 907137553 is specified. This transform generates a dataset that excludes columns "age" and "gender."

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["removecolumns"],
     "sourceId":null,
     "transform":
     {
      "inputs":["2396326717"],
      "transformSteps":
      [
       {
        "type":"RemoveColumns",
        "removeColumns":"age"
       },
       {
        "type":"RemoveColumns",
        "removeColumns":"gender"
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion of the transform, the API returns a new dataset. The response will be similar to below. Refer to *Transform Return Fields* (page 57) for information about the fields that display in this response.

```
200 (OK)
"Content-type:application/json"

{
 "id": "3736676442",
 "name": "removecolumns",
 "createdAt": 1407870442671,
 "updatedAt": 1407870442671
 "status":
 {
  "code": "INPROGRESS",
  "message": "Data Set is currently being validated"
 },
 "sourceId": null,
 "transform":
 {
  "inputs": ["2396326717"],
  "transformSteps":
  [
   {
    "removeColumns": "age",
    "type": "RemoveColumns"
   },
   {
    "removeColumns": "gender",
    "type": "RemoveColumns"
   }
  ],
  "blobColumns": null,
  "subStatus": null,
 },
 "projectId":"24389108236",
 "columns": null
}
```

Split-by-Ratio Transform

The Split-by-Ratio transform allows you to split an existing dataset into multiple datasets.

Note: You must specify an output file for each split.

Split-by-Ratio Transform Parameters

The Split-by-Ratio transform accepts the following parameters:

Table 23: *Split Transform Parameters*

Parameter	Description
`names`	Specify names for the resulting split datasets. The number of names must equal the number specified by `splitRatios`.
`splitRatios`	Specify the ratio of each split. For example, if you want to specify a file be split into two equal parts, then this value would be `[1,1]`. Similarly, if you want a file to be split in half, with the second half split in two equal parts, then this value would be `[2,1,1]`. Floating values are also accepted. For example, to perform an 80%/20% split, you can specify `[0.8,0.2]`.
`splitSeed`	Specify a seed value for the random shuffle.

Request

Use `POST` to split a dataset. The following example splits a dataset into three equal parts, resulting in three files.

```
curl -X POST -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-Type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["split_1","split_2","split_3"],
     "sourceId":null,
     "transform":
     {
      "inputs":["2754676572"],
      "transformSteps":
      [
       {
        "splitRatios":[1,1,1],
        "splitSeed":234221,
        "type":"SplitByRatio"
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets/'
```

Response

```
[
  {
    "id": "3794113559",
    "name": "split_1",
    "createdAt": 1410793054869,
    "updatedAt": 1410793054869,
    "status":
    {
      "code": "INPROGRESS",
      "message": "Data Set is currently being validated"
    },
    "sourceId": null,
    "transform":
```

```
  {
   "inputs": ["2754676572"],
   "transformSteps":
    [
      {
        "splitRatios": [ 1, 1, 1 ],
        "splitSeed": 234221,
        "type": "SplitByRatio"
      }
    ],
    "blobColumns": null,
    "subStatus": null
  },
  "columns": null
},
{
  "id": "3090760026",
  "name": "split_2",
  "createdAt": 1410793055289,
  "updatedAt": 1410793055289,
  "status":
    {
      "code": "INPROGRESS",
      "message": "Data Set is currently being validated"
    },
  "sourceId": null,
  "transform":
  {
   "inputs": ["2754676572"],
   "transformSteps":
   [
     {
       "splitRatios": [ 1, 1, 1 ],
       "splitSeed": 234221,
       "type": "SplitByRatio"
     }
   ],
   "blobColumns": null,
   "subStatus": null
  },
  "columns": null
},
{
  "id": "190364645",
  "name": "split_3",
  "createdAt": 1410793055709,
  "updatedAt": 1410793055709,
  "status":
    {
      "code": "INPROGRESS",
      "message": "Data Set is currently being validated"
    },
  "sourceId": null,
  "transform":
  {
   "inputs": ["2754676572"],
```

```
    "transformSteps":
    [
      {
        "splitRatios": [ 1, 1, 1 ],
        "splitSeed": 234221,
        "type": "SplitByRatio"
      }
    ],
    "blobColumns": null,
    "subStatus": null
   },
   "columns": null
  },
  "projectId":"24389108236",
]
```

Tokenize Text

This transformation tokenizes the blob column in a text dataset. Note that this can only be used on datasets that have been transformed already with ingested text. After this completes successfully, you can set this column as categorical and then extract the text features.

Tokenize Text Parameters

The Tokenize Text transform accepts the following parameter:

Table 24: *Tokenize Text Transform Parameters*

Parameter	Description
`targetBlobColumn`	Specify the blob column to be tokenized in the new dataset

Request

The request below uses `POST` to specify that a column in the 907137553 dataset should be tokenized. This dataset includes a blob column named "doc".

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
    "projectId":"24389108236",
    "names":["Tokenize Text"]
    "sourceId":null,
    "transform":
    {
     "inputs":["907137553"],
     "transformSteps":
     [
      {
       "targetBlobColumn":"doc",
       "type":"TokenAnnotation"
      }
     ]
    }
```

```
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Vectorize Text

This transform vectorizes the features in a dataset. This transform is done on a dataset that includes a label column. A dataset that was transformed using Extracted Text Features is also required. (See Extract Text Features (page 40).)

Vectorize Text Parameters

The Vectorize Text transform accepts the following parameter:

Table 25: *Vectorize Text Transform Parameters*

Parameter	Description
`targetBlobColumn`	Specify the blob column to be vectorized in the new dataset.
`labelColumnName`	Specify the label column to be vectorized in the new dataset.

Request

The following example use POST to a vectorized dataset using the labels from the 83224872 dataset and the features the 907137553 dataset.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["Vectorize Text"]
     "sourceId":null,
     "transform":
     {
      "inputs":["83224872","907137553"],
      "transformSteps":
      [
       {
        "targetBlobColumn":"doc",
        "labelColumnName":"labels",
        "type":"TextVectorSpace"
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Specifying Multiple Transforms

You can apply multiple transform types to a dataset. For example, you might to filter a dataset based on an age while ignoring the top 100 rows in a dataset.

Request

This example specifies the following transforms.

- Do not include any rows that include a year earlier than 1993
- Categorize text in column 2
- Horizontalize column 3
- Do not include the "Gender" column

The response will be similar to those described previously in this section. Refer to *Transform Return Fields* (page 57) for information about the fields that display in this response.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["Filtered Dataset"],
     "sourceId":null,
     "transform":
     {
      "inputs":["907137553"],
      "transformSteps":
      [
       {
        "filterExpression":"lambda year: year not < 1993",
        "type":"Filter"
       },
       {
        "categorizationType":"Text",
        "categorizationColumn":"Column_2",
        "type":"Categorize"
       },
       {
        "horizontalizeColumn":"Column_3",
        "type":"Horizontalize"
       },
       {
        "removeColumns":"Gender",
        "type":"RemoveColumns"
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Transform Return Fields

A successfully transformed dataset will output with the following properties:

Table 26: *Transform Return Properties*

Return Field	Description
`id`	The ID of the dataset that was just created
`name`	The name of the dataset
`createdAt`	The epoch time when the dataset was created

Table 26: *Transform Return Properties (continued)*

Return Field	Description
`updatedAt`	The epoch time when the dataset was updated
`status`	The Skytree status (for example, READY) and a message (for example, "Data Set has been validated and is ready to use.")
`sourceId`	The ID of the source that was used to create this dataset
`transformSteps`: The fields below define the options that were used to transform this dataset.	
`inputs`	The unique dataset ID that was used as the basis for this transform
`filterExpression`	Defines the filter when filtering rows. Otherwise, this value is "`null`".
`filterColumn`	Defines the column name that was excluded. Otherwise, this value is "`null`".
`horizontalizeColumn`	Shows the column name that was horizontalized if the transform type is Horizontalize. Otherwise, this value is "`null`".
`categorizationType`	Shows the type of data that was categorized if the transform type is Categorize. Otherwise, this value is "`null`". `categorizationType` is a 1-based index. Accepted values include: • `Number` • `Text`
`categorizationColumn`	Shows the column name to be used during the transform if the transform type is Categorize. If the dataset does not contain headers, columns are named "`Column_<index>`" where `<index>` is a 1-based index. If the transform type is not Categorize, then this value is "`null`".
`clampColumn`	The column name or number to be considered when clamping values.
`max`	When normalizing , this shows the maximum value to be included in the transform. Values in the header that are larger than this value will be represented with this value instead. Note that this is not available with `mean` or `stdev`.
`min`	When normalizing, this shows the minimum value to be included in the transform. Values in the header that are smaller than this value will be represented with this value instead. Note that this is not available with `mean` or `stdev`
`mean`	When normalizing, this shows the mean value to be used in the transform. This option is used with `stdev`. It is not used with `max` or `min`.
`stdev`	When normalizing based on a mean value, this shows the standard deviation value to be used in the transform. This is not used with `max` or `min`.
`normalizeColumn`	Shows the column that was normalized.
`normalizationType`	Shows the specified normalization type. This can be one of the following: • `Unit` • `Standard`
`maxValue`	When normalizing, this represents the maximum value represented in the transform.
`minValue`	When normalizing, this represents the minimum value represented in the transform.

Table 26: *Transform Return Properties (continued)*

Return Field	Description
splitRatios	Shows the ratio of each split in the transform. For example, a value of [1,1,1] indicates that a file was split in three equal parts.
splitSeed	Shows seed value for the random shuffle in a Split transform.
labelColumnName	The label column that was set as categorical text in the Set as Catetorical Text transform.
featureWeightingScheme	Featurizes using inverse document frequency (tf_idf) or bi-normal separation (tf_bns).
minFeatureWeight	The minimum weight to be given to a word.
maxFeatureSize	The maximum size of a word.
targetBlobColumn	Shows the column in the dataset that includes the blob (free form) text.
type	Shows the type of transform that was used. Values include the following: • AddUniqueIDColumn • Categorize • ClampValues • Custom • Extract Text Features • Filter • FilterColumn • Horizontalize • InjestSelfContainedText • InjestText • InnerJoin • LeftOuterJoin • NewColumn • Normalize • RightOuterJoin • SplitByRatio • TokenizeText • VectorizeText If multiple transforms were applied to a dataset, then the type will display once for each transform type that was used.

Chapter 7 Custom Transforms

The Custom transform allows you to reference a Spark script, written in Python, within a transform. This allows you to access many additional Spark functions not directly encompassed in Skytree transforms.

Applying a custom transform is done in two steps:

1. Uploading a script file
2. Creating a custom transform

These steps are described in the sections that follow. An additional section is also included that provides a list of functions and variables available for custom scripts. Refer to *Custom Script Objects* (page 64) for a list of supported variables.

> **Note:** When applying transformations to a dataset, the source dataset and the transformed dataset must reside in the same project.

URLs

Use the following base URLs to manage custom transforms:

```
'http://api.skytree.net:8080/v1/scripts'
'http://api.skytree.net:8080/v1/datasets'
```

Uploading a Script File

When calling scripts in a custom transform, the scripts must first be added to API project.

Script Parameters

The Script function accepts the following parameters:

Table 27: *Script Transform Parameters*

Parameter	Description
name	Specify a unique name for this script file.

Table 27: *Script Transform Parameters (continued)*

Parameter	Description
`scriptFiles`	Specify the path to the script that you want to use for the custom transform. This parameter can be specified multiple times to include multiple scripts. The scripts will be run in the order that they are arranged in the request.

Example Script

The following example shows a script that specifies to exclude rows with a number of columns different from the number of columns in the header. Note that this assumes the dataset has no extra or trailing separators. The name of this script is called `exclude_inconsistent_rows.py`. This file will be called in the Script request that follows.

```
numberOfColumns = sc.broadcast(len(inputMetadata[0].header.columns))

def consistentRow(line):
    return len(line) == numberOfColumns.value()

output =   input[0].filter(consistentRow)
```

Request

Use `POST` to upload a script file. The following example uploads the `exclude_inconsistent_rows.py` file. The name for the uploaded script is "`UploadedScript`."

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" -F name="UploadedScript" \
-F scriptFiles=@/usr/scripts/exclude_inconsistent_rows.py \
'http://api.skytree.net:8080/v1/scripts'
```

Response

Upon successful completion, the response returns the unique ID of the script. A READY script can then be referenced in a custom transform. The response will look similar to the following:

```
200 (OK)
"Content-type:multipart/form-data"
{
 "id":"788532485",
 "name":"UploadedScript",
 "createdAt":1412708896502,
 "updatedAt": 1412708896524,
 "status":
 {
  "code":"READY",
  "message":"Script is ready to be used"
 },
```

Creating a Custom Transform

This section describes how to configure a custom transform using a file that was previous uploaded.

Custom Transform Parameters

The Custom transform accepts the following parameters:

Table 28: *Custom Transform Parameters*

Parameter	Description
`customScriptName`	Specify the name and path to the script that you want to use for the custom transform. This parameter can be specified multiple times to include multiple scripts. The scripts will be run in the order that they are arranged in the request.
`customScriptArgs`	Specify an array of arguments

Request

Use `POST` to use a script to create a custom transform. The following example references script `788532485` in a custom transform. This is the ID of the script previously created. The resulting dataset will exclude inconsistent rows.

```
curl -X POST -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "names":["custom-dataset"],
     "sourceId":null,
     "transform":
     {
      "inputs":["2396326717"],
      "transformSteps":
      [
       {
        "type":"Custom",
        "customScriptName":788532485,
        "customScriptArgs":null
       }
      ]
     }
    }' 'http://api.skytree.net:8080/v1/datasets'
```

Response

Upon successful completion, the response returns the unique ID of the script. A READY script can then be referenced in a custom transform. The response will look similar to the following:

```
200 (OK)
"Content-type:multipart/form-data"
{
 "id":"1704667445",
 "name":"custom-dataset",
 "createdAt":1412709803845,
 "updatedAt":1412709803845,
 "status":
 {
  "code":"INPROGRESS",
```

```
 "message":"Data Set is currently being validated"
},
"sourceId": null,
"transform":
{
 "inputs":["2396326717"],
 "transformSteps":
 [
  {
   "type":"Custom",
   "customScriptName":788532485,
   "customScriptArgs":null
  },
 ]
},
"configuration":
{
 "blobColumns": null,
 "idColumn": null,
},
"projectId":"24389108236",
"columns": null
}
```

Custom Script Objects

This section describes the objects that Skytree provides for the user to use in custom scripts. Consult the Spark documentation for further details on behavior and functions of the Spark objects and methods.

sc

A restricted version of SparkContext.

Methods

Method signatures are provided for the underlying Scala functions. In a Skytree custom script, these would be accessed through Python.

Create an Accumulable shared variable of the given type, to which tasks can "add" values with add.

```
<T,R> Accumulable<T,R> accumulable(T initialValue, AccumulableParam<T,R> param)
```

Create an Accumulator variable of a given type, which tasks can "add" values to using the add method.

```
<T> Accumulator<T> accumulator(T initialValue, AccumulatorParam<T>
accumulatorParam)
```

Create an Accumulator integer variable, which tasks can "add" values to using the add method.

```
Accumulator<Integer> accumulator(Integer initialValue)
```

Create an Accumulator double variable, which tasks can "add" values to using the add method.

```
Accumulator<Double> accumulator(double initialValue, String name)
```

Application name

```
String appName()
```

Broadcast a read-only variable to the cluster, returning a Broadcast object for reading it in distributed functions.

```
<T> Broadcast<T> broadcast(T value)
```

Default min number of partitions for Hadoop RDDs when not given by user

```
Integer defaultMinPartitions()
```

Default level of parallelism to use when not given by user (e.g. parallelize).

```
Integer defaultParallelism()
```

Create an Accumulator double variable, which tasks can "add" values to using the add method.

```
Accumulator<Double> doubleAccumulator(double initialValue)
```

Get a local property set in this thread, or null if it is missing.

```
String getLocalProperty(String key)
```

Get Spark's home location from either a value set through the constructor, or the spark.home Java property, or the SPARK_HOME environment variable (in that order of preference).

```
String getSparkHome()
```

Create an Accumulator integer variable, which tasks can "add" values to using the add method.

```
Accumulator<Integer> intAccumulator(int initialValue)
```

Is Spark running in Local mode

```
Boolean isLocal()
```

Distribute a local Scala collection to form an RDD.

```
<T> JavaRDD<T>  parallelize(java.util.List<T> list)
```

Distribute a local Scala collection to form an RDD.

```
<T> JavaRDD<T>  parallelize(java.util.List<T> list, int numSlices)
```

Distribute a local Scala collection to form an RDD.

```
JavaDoubleRDD   parallelizeDoubles(java.util.List<Double> list)
```

Distribute a local Scala collection to form an RDD.

```
JavaDoubleRDD   parallelizeDoubles(java.util.List<Double> list, int numSlices)
```

Distribute a local Scala collection to form an RDD.

```
<K,V> JavaPairRDD<K,V> parallelizePairs(java.util.List<scala.Tuple2<K,V>> list)
```

Distribute a local Scala collection to form an RDD.

```
<K,V> JavaPairRDD<K,V> parallelizePairs(java.util.List<scala.Tuple2<K,V>> list, int
numSlices)
```

Context start time.

```
Long startTime()
```

Input

Description:

Array of input RDDs. An input RDD is of type `SkytreeJavaRDD<String[]>`.

Methods:

Method signatures are provided for the underlying Scala functions. In a Skytree custom script, these would be accessed through Python.

Gets a list of column names for the RDD
```
String[] getColumnNames()
```

Gets a list of Column objects
```
List<Column> getColumns()
```

Gets the Column object for a given column name
```
Column getColumn(String columnName)
```

Output

Description:

RDD or Array of RDDs sent back to the system

Methods:

Method signatures are provided for the underlying Scala functions. In a Skytree custom script, these would be accessed through Python

Sets the column names for the output RDD
```
void setColumnNames(String[] columnNames)
```

Sets the type of the given column to Text
```
void setColumnTypeText(String columnName)
```

Sets the type of the given column to Integer
```
void setColumnTypeInteger(String columnName)
```

Sets the type of the given column to Double
```
void setColumnTypeFloat(String columnName)
```

Sets the given column's categorical flag
```
void setColumnCategorical(String columnName, boolean isCategorical)
```

Sets the given column's blob flag
```
void setColumnTypeBlob(String columnName)
```

Column

Methods:

Method signatures are provided for the underlying Scala functions. In a Skytree custom script, these would be accessed through Python.

Gets the name of the column

```
string getName()
```

Gets the 1-based index of the column

```
long getIndex()
```

Returns True if the column is marked as categorical

```
boolean isCategorical()
```

Returns the type of the column. One of Text, Integer, Float

```
type getType()
```

RDD

Methods

Return a new RDD containing only the elements that satisfy a predicate function `f`, which takes as input an RDD row.

```
filter(f)
```

Aggregate the elements of each partition, and then the results for all the partitions, using given combine functions and a neutral "zero value".

```
aggregate(zeroValue, seqOp, combOp)
```

Return a new RDD by first applying a function to all elements of this RDD, and then flattening the results.

```
flatMap(f)
```

Return a new Pair-RDD by first applying a function to all elements of this RDD, and then flattening the results.

```
flatMapToPair(f)
```

Aggregate the elements of each partition, and then the results for all the partitions, using a given associative function and a neutral "zero value".

```
fold(zeroValue, f)
```

Applies a function f to all elements of this RDD.

```
foreach(f)
```

Applies a function f to each partition of this RDD.

```
foreachPartition(f)
```

Return a Pair-RDD of grouped elements.

```
groupBy(f)
```

Return a Pair-RDD of grouped elements with the given number of partitions.

```
groupBy(f, numPartitions)
```

Creates a Pair-RDD of tuples of the elements in this RDD by applying f.

```
keyBy(f)
```

Return a new RDD by applying a function to all elements of this RDD.

```
map(f)
```

Return a new RDD by applying a function to each partition of this RDD.

```
mapPartitions(f)
```

Return a new RDD by applying a function to each partition of this RDD. Set preservesPartitioning to True to keep the same partitions.

```
mapPartitions(f, preservesPartitioning)
```

Return a new Pair-RDD by applying a function to each partition of this RDD.

```
mapPartitionsToPair(f)
```

Return a new RDD by applying a function to each partition of this RDD, while tracking the index of the original partition. Set preservesPartitioning to True to preserve the original partitioning.

```
mapPartitionsWithIndex(f, preservesPartitioning)
```

Return a new Pair-RDD by applying a function to all elements of this RDD.

```
mapToPair(f)
```

Return a new RDD that is reduced into numPartitions partitions.

```
coalesce(numPartitions)
```

Return an array that contains all of the elements in this RDD.

```
collect()
```

Return the number of elements in the RDD.

```
count()
```

Return an array that contains all of the elements in specific partitions of this RDD.

```
collectPartitions(partitionIds)
```

Return the count of each unique value in this RDD as a map of (value, count) pairs.

```
countByValue()
```

Return a new RDD containing the distinct elements in this RDD.

```
distinct()
```

Return the first element in this RDD.

```
first()
```

Return the number of partitions in this RDD.

```
getNumPartitions()
```

Return an RDD created by coalescing all elements within each partition into an array.

```
glom()
```

Return the intersection of this RDD and another one.

```
intersection(rdd)
```

Returns the maximum element from this RDD as defined by the specified comparator f.

```
max(f)
```

Returns the minimum element from this RDD as defined by the specified comparator f.

```
min(f)
```

Reduces the elements of this RDD using the specified commutative and associative binary operator f.

```
reduce(f)
```

Return a new RDD that has exactly numPartitions partitions.

```
repartition(numPartitions)
```

Randomly selects a fraction of the items of a RDD and returns them in a new RDD.

```
sample(withReplacement, fraction)
```

Randomly selects a fraction of the items of a RDD and returns them in a new RDD.

```
sample(withReplacement, fraction, seed)
```

Set of partitions in this RDD.

```
splits()
```

Take the first num elements of the RDD.
```
take(num)
```

Return an RDD with the elements from this that are not in other.
```
subtract(rdd)
```

Return an RDD with the elements from this that are not in other.
```
subtract(rdd, numPartitions)
```

Get the N elements from a RDD ordered in ascending order.
```
takeOrdered(num)
```

Return a fixed-size sampled subset of this RDD (currently requires numpy).
```
takeSample(withReplacement, num)
```

Return a fixed-size sampled subset of this RDD.
```
takeSample(withReplacement, num, seed)
```

Returns the top num elements from this RDD using the natural ordering for T.
```
top(num)
```

Return the union of this RDD and another one.
```
union(rdd)
```

Zips this RDD with another one, returning key-value pairs (Pair-RDD) with the first element in each RDD, second element in each RDD, etc.
```
zip(rdd)
```

Zip this RDD's partitions with one (or more) RDD(s) and return a new RDD by applying a function to the zipped partitions.
```
zipPartitions(rdd, f)
```

Zips this RDD with its element indices. (Requires a pass through the data to determine partition lengths)
```
zipWithIndex()
```

Zips this RDD with generated unique Long ids.
```
zipWithUniqueId()
```

Return a new Double-RDD by first applying a function to all elements of this RDD, and then flattening the results.
```
flatMapToDouble(f)
```

Return a new Double-RDD by applying a function to each partition of this RDD.
```
mapPartitionsToDouble(f)
```

Return a new Double-RDD by applying a function to each partition of this RDD.
```
mapPartitionsToDouble(f, preservesPartitioning)
```

Return a new Double-RDD by applying a function to all elements of this RDD.
```
mapToDouble(f)
```

Double RDD

Methods

Compute a histogram using the provided buckets.
```
histogram(buckets)
```

Compute the mean of this RDD's elements.
```
mean()
```

Return a StatCounter object that captures the mean, variance and count of the RDD's elements in one operation.

```
stats()
```

Compute the variance of this RDD's elements.

```
variance()
```

Add up the elements in this RDD.

```
sum()
```

Compute the standard deviation of this RDD's elements.

```
stdev()
```

Compute the sample standard deviation of this RDD's elements (which corrects for bias in estimating the standard deviation by dividing by N-1 instead of N).

```
sampleStdev()
```

Compute the sample variance of this RDD's elements (which corrects for bias in estimating the standard variance by dividing by N-1 instead of N).

```
sampleVariance()
```

Return the count of each unique value in this RDD as a map of (value, count) pairs. The final combine step happens locally on the master, equivalent to running a single reduce task.

```
countByValue()
```

Return the union of this RDD with another.

```
union(rdd)
```

Applies a function f to all elements of this RDD.

```
foreach(f)
```

Return an array that contains all of the elements in this RDD.

```
collect()
```

Return a new Double-RDD containing only the elements that satisfy a predicate.

```
filter(f)
```

Return the first element in this RDD.

```
first()
```

Return the intersection of this RDD and another one.

```
intersection(rdd)
```

Return a new Double-RDD that has exactly numPartitions partitions.

```
repartition(numPartitions)
```

Return an RDD with the elements from this that are not in other.

```
subtract(rdd, numPartitions)
```

Return a new RDD by applying a function to all elements of this RDD.

```
map(f)
```

Pair RDD

Methods

Aggregate the elements of each partition, and then the results for all the partitions, using given combine functions and a neutral "zero value".

```
  aggregate(zeroValue, seqOp, combOp)
```

Return the number of elements in the RDD.

```
count()
```

Return an array that contains all of the elements in specific partitions of this RDD.

```
collectPartitions(partitionIds)
```

Generic function to combine the elements for each key using a custom set of aggregation functions.

```
combineByKey(createCombiner, mergeValue, mergeCombiners)
```

Generic function to combine the elements for each key using a custom set of aggregation functions.

```
combineByKey(createCombiner, mergeValue, mergeCombiners, numPartitions)
```

Return the count of each unique value in this RDD as a map of (value, count) pairs.

```
countByValue()
```

Return a new RDD containing the distinct elements in this RDD.

```
distinct()
```

Return the first element in this RDD.

```
first()
```

Return the number of partitions in this RDD.

```
getNumPartitions()
```

Return an RDD created by coalescing all elements within each partition into an array.

```
glom()
```

Return the intersection of this RDD and another one.

```
intersection(rdd)
```

Sort the RDD by key, so that each partition contains a sorted range of the elements.

```
sortByKey()
```

Sort the RDD by key, so that each partition contains a sorted range of the elements. Set ascending to True to get an ascending sort.

```
sortByKey(ascending)
```

Return an RDD with the values of each tuple.

```
values()
```

Return an RDD with the keys of each tuple.

```
keys()
```

Return a new RDD by applying a function to all elements of this RDD.

```
map(f)
```

Return a new RDD that is reduced into numPartitions partitions.

```
coalesce(numPartitions)
```

Return a new RDD that is reduced into numPartitions partitions.

```
coalesce(numPartitions, shuffle)
```

For each key k in this RDD or the other RDD, return a resulting Pair-RDD that contains a tuple with the list of values for that key in this as well as other.

```
cogroup(other)
```

For each key k in this or other1 or other2, return a resulting Pair-RDD that contains a tuple with the list of values for that key in this, other1 and other2.

```
cogroup(other1, other2)
```

Return an array that contains all of the elements in this RDD.

```
collect()
```

Return the key-value pairs in this RDD to the master as a Map.

```
collectAsMap()
```

Count the number of elements for each key, and return the result to the master as a Map.

```
countByKey()
```

Return a new RDD containing the distinct elements in this RDD.

```
distinct()
```

Merge the values for each key using an associative function "func" and a neutral "zeroValue" which may be added to the result an arbitrary number of times, and must not change the result (e.g., 0 for addition, or 1 for multiplication.).

```
foldByKey(zeroValue, f)
```

Merge the values for each key using an associative function "func" and a neutral "zeroValue" which may be added to the result an arbitrary number of times, and must not change the result (e.g., 0 for addition, or 1 for multiplication.).

```
foldByKey(zeroValue, f, numPartitions)
```

Applies a function f to all elements of this RDD.

```
foreach(f)
```

Return a Pair-RDD of grouped elements.

```
groupBy(f)
```

Alias for cogroup but with support for multiple RDDs.

```
groupWith(other)
```

Alias for cogroup but with support for multiple RDDs.

```
groupWith(other1, other2)
```

Group the values for each key in the RDD into a single sequence.

```
groupByKey(numPartitions)
```

Creates a Pair-RDD of tuples of the elements in this RDD by applying f.

```
keyBy(f)
```

Return an RDD containing all pairs of elements with matching keys in this and other.

```
join(rdd)
```

Return an RDD containing all pairs of elements with matching keys in this and other.

```
join(red, numPartitions)
```

Perform a left outer join of this and other.

```
leftOuterJoin(rdd)
```

Perform a left outer join of this and other.

```
leftOuterJoin(rdd, numPartitions)
```

Perform a right outer join of this and other.

```
rightOuterJoin(rdd)
```

Perform a right outer join of this and other.

```
rightOuterJoin(rdd, numPartitions)
```

Return the list of values in the RDD for key key.

```
lookup(key)
```

Merge the values for each key using an associative reduce function.

```
reduceByKey(f)
```

Return a new Pair-RDD that has exactly numPartitions partitions.

```
repartition(numPartitions)
```

Randomly selects a fraction of the items of a RDD and returns them in a new RDD.

```
sample(withReplacement, fraction)
```

Randomly selects a fraction of the items of a RDD and returns them in a new RDD.

```
sample(withReplacement, fraction, seed)
```

Set of partitions in this RDD.

```
splits()
```

Take the first num elements of the RDD.

```
take(num)
```

Return an RDD with the elements from this that are not in other.

```
subtract(rdd)
```

Return an RDD with the elements from this that are not in other.

```
subtract(rdd, numPartitions)
```

Return each (key, value) pair in this rdd that has no pair with matching key in other.

```
subtractByKey(other)
```

Return each (key, value) pair in this rdd that has no pair with matching key in other.

```
subtractByKey(other, numPartitions)
```

Get the N elements from a RDD with the ordering specified using comparator function f.

```
takeOrdered(num, f)
```

Return a fixed-size sampled subset of this RDD (currently requires numpy).

```
takeSample(withReplacement, num)
```

Return a fixed-size sampled subset of this RDD.

```
takeSample(withReplacement, num, seed)
```

Returns the top num elements from this RDD using the natural ordering for T.

```
top(num)
```

Return the union of this RDD and another one.

```
union(rdd)
```

Zips this RDD with another one, returning key-value pairs (Pair-RDD) with the first element in each RDD, second element in each RDD, etc.

```
zip(rdd)
```

Chapter 8 Models

Skytree allows the user to train a variety of different model types on their datasets, tune the parameters of those models, test those models, and use them to predict values.

Classification vs. Regression

Skytree methods allow you to perform classification and regression problems.

Binary Classification

Given two sets of points, one for class A and one for class B, users want to find a model that can separate efficiently the points of classes A and B. Classification models are trained on a dataset where the correct class labels are known. They are used to predict, for unlabeled data, the probabilities that each point is of class A or B, and therefore predict a class label.

Regression

Regression problems involve predicting a continuous value rather than a categorical class label. The training dataset needs to have correct objective values for each rows, and the resulting model predicts, for unlabeled data, the value of the objective.

URLs

Use the following base URLs to manage modeling resources:

```
'http://api.skytree.net:8080/v1/gbt/train'
'http://api.skytree.net:8080/v1/gbt/test'
'http://api.skytree.net:8080/v1/gbtr/train'
'http://api.skytree.net:8080/v1/gbtr/test'
'http://api.skytree.net:8080/v1/glmr/train'
'http://api.skytree.net:8080/v1/glmr/test'
'http://api.skytree.net:8080/v1/glmc/train'
'http://api.skytree.net:8080/v1/glmc/test'
'http://api.skytree.net:8080/v1/rdf/train'
'http://api.skytree.net:8080/v1/rdf/test'
'http://api.skytree.net:8080/v1/rdfr/train'
'http://api.skytree.net:8080/v1/rdfr/test'
'http://api.skytree.net:8080/v1/svm/train'
'http://api.skytree.net:8080/v1/svm/test'
'http://api.skytree.net:8080/v1/automodel/train'
'http://api.skytree.net:8080/v1/automodel/test'
'http://api.skytree.net:8080/v1/models'
```

```
'http://api.skytree.net:8080/v1/models/<model_id>'
'http://api.skytree.net:8080/v1/models/<model_id>/dependents'
'http://api.skytree.net:8080/v1/models/<model_id>/interrupt'
'http://api.skytree.net:8080/v1/models?projectId=<project_id>'
'http://api.skytree.net:8080/v1/models/<model_id>/pmml'
'http://api.skytree.net:8080/v1/models/<model_id>/pmml/gzip'
```

Methods

Skytree allows you to build models using one of the following learning methods:

- AutoModel
- Gradient Boosted Trees (GBT)
- Gradient Boosted Trees Regression (GBTR)
- Generalized Linear Models (GLM)
- Generalized Linear Models Regression (GLMR)
- Random Decision Forests (RDF)
- Random Decision Forests Regression (RDFR)
- Support Vector Machines (SVM)

These methods are summarized in the sections that follow. In addition, refer to *Model Parameters* (page 86) to see the configuration options that can be used with each method.

AutoModel Models

Skytree AutoModel automatically determines the best model and parameter settings to solve a particular problem. It explores, in an intelligent way, all possible models and their parameter settings and picks the model that is most effective at making the required prediction. AutoModel selects whether to use classification or regression based on the type of the objective column; continuous values are predicted with regression models, and categorical values are predicted with classification models.

Gradient Boosted Trees

Gradient Boosted Trees (GBT) is a machine learning algorithm combining decision trees with boosting. The latter is a numerical gradient optimization method that minimizes a loss function, in this case, the deviance.

The optimization is achieved by successively adding trees that best reduce the loss function. The first tree reduces the loss function to the greatest extent possible, while subsequent trees focus on a residual that represents the poorest responses of the model.

As the boosting iteration proceeds, the algorithm adjusts the relative weighting of each tree without modifying the previous trees themselves. The resulting model is a linear combination of all trees.

GBTR Models

Gradient Boosted Trees Regression (GBTR) is a machine learning algorithm combining regression trees with boosting. The latter is a numerical gradient optimization method that minimizes a loss function, in this case, the deviance.

The optimization is achieved by successively adding trees that best reduce the loss function. The first tree reduces the loss function to the greatest extent possible, while subsequent trees focus on a residual that represents the poorest responses of the model.

As the boosting iteration proceeds, the algorithm adjusts the relative weighting of each tree without modifying the previous trees themselves. The resulting model is a linear combination of all trees.

Generalized Linear Models - Classification and Regression

Linear Regression models the relationship between a group of regressors (predictors) $X_1, \cdots, X_p$ and the prediction values y using a linear fit. The Generalized Linear Model method is an extension of ordinary Linear Regression and consists of the following components:

- A probability distribution family f in the exponential family with expected value $\mu = E_f(y)$
- A linear model $\eta = \mathbf{x}^T\beta$
- A link function g relating $g(\mu) = \eta$

Accordingly, in order to specify a GLM problem, you must choose family function f, link function g, and any parameters needed to train the linear model weights β.

Note: A family function cannot be specified in GLMC because only a single family (Binomial) is available.

Skytree supports the following Family/Link Function combinations:

Table 29: *Family/Link Function Combinations*

Family	Link Function					
	Identity	Square Root	Inverse	Log	Logit	Cloglog
Gaussian	X					
Poisson	X	X		X		
Gamma			X	X		
Tweedie				X		
Binomial					X	X

Note: Certain link functions, such as Log and Square-Root, are only applicable to non-negative responses.

RDF Models

The Random Decision Forests (RDF) algorithm is an ensemble classifier composed of a collection of decision trees. The ensemble returns classification based on the aggregate results of the individual trees. Each tree in the forest is a weak classifier, trained by selecting a subset of the training data, usually with replacement, and a random selection of the available features.

Each tree is grown using a sampling of the original training data. Nodes of the tree are split by considering a subset of all features of the data and are grown without pruning.

Each ensemble classification requires classification by each tree in the forest. The class with a plurality of votes determines the final classification.

Several factors affect the error of the model. Strong individual trees in the ensemble decrease the error rate and occur with a larger number of trees. However, a large number of trees also introduces correlated trees which increase the error rate. For these reasons, among others, it's important that the parameters of the Random Decision Forest be tuned to provide the best possible error rate.

RDFR Models

The Random Decision Forest Regression (RDFR) algorithm is an ensemble regression model comprising a collection of regression trees. The ensemble returns predictions based on the average of the results of the individual trees. Each tree in the forest is a weak regression model, trained by selecting a subset of the training data, usually with replacement, and a random selection of the available features.

Each tree is grown using a sampling of the original training data. Nodes of the tree are split by considering a subset of all features of the data and are grown without pruning.

Each ensemble regression prediction requires a regression prediction by each tree in the forest. The average of the predictions of the individual trees determines the final prediction.

Several factors affect the error of the model. Strong individual trees in the ensemble decrease the error rate and occur with a larger number of trees. However, a large number of trees also introduces correlated trees which increase the error rate. For these reasons, among others, it's important that the parameters of the Random Decision Forest Regression model be tuned to provide the best possible error rate.

SVM Models

The Support Vector Machine (http://en.wikipedia.org/wiki/Support_vector_machine) (SVM) is one of the most successful classification algorithms. It is based on the revolutionary theory of kernel learning. The main concept is that SVM tries to fit a hyperplane that optimally separates different classes of data. The hyperplane can be linear or nonlinear.

- Find the best boundary to separate points into two classes
- Maps points onto a high-dimensional space so that it can learn complex nonlinear decision boundaries
- If no perfect separation, maximizes the margin
- An advantage is no local-minimum situations
- Naively scales as N^2-N^3; Skytree scaling is data dependent

Train a Model

This section describes how to train a model. The process is the same for each method, though the URL and configuration options will vary.

> **Note:** The training dataset specified during model creation must reside in the specified project. You will get an error if you attempt to reference a dataset ID that is in a different project.

Request

Use POST to train a dataset and produce a model using.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"gbt_train_model",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[2],
      "objectiveColumn":"yearly-income",
      "samplingRatio":[[1.0],[2.3]],
      "classweight":[1.0,2.0]
     }
    }' 'http://api.skytree.net:8080/v1/gbt/train'
```

Response

After a model is trained, Skytree stores the model using a unique identifier, which is a string of a maximum of 200 alpha-numeric characters. This identifier can be used to retrieve a model and can be specified when producing predictions.

```
200 (OK)
"Content-type:application/json"

{
 "id":"1850758990",
 "name":"gbt_train_model",
 "dataSetId":"1483908776",
 "status":
 {
  "code":"READY",
  "message":"Model has been validated and is ready to use"
 },
 "createdAt":1401202557204,
 "updatedAt":1401202557204,
 "configuration":
 {
  "holdoutRatio":null,
  "numFolds":null,
  "objectiveColumn":"yearly-income",
  "sampleWithReplacement":null,
  "treeDepth":null,
  "numTrees":
```

```
 {
  [2]
 },
 "numDimensions":null,
 "holdoutSeed":null,
 "categoricalSelectionMethod":null,
 "categoricalRandomSplitTries":null,
 "categoricalRandomSplitThreshold":null,
 "dimensionSamplingSeed":null,
 "tableSamplingSeed":null,
 "categoricalSamplingSeed":null,
 "testingObjective":null,
 "maxSplits":null,
 "minNodeWeight":null,
 "cardinalityBasedDimensionSampling":null,
 "imbalance":null,
 "imbalanceScale":null,
 "learningRate":null,
 "ensembleSize":null,
 "samplingRatio":
 {
  {
   [1.0]
  },
  {
   [2.3]
  }
 },
 "classweight":
 {
  [1.0,2.0]
 },
 "regularizationBins":null,
 "regularization":null,
 "trim":null,
 "trimAlpha":null,
 "classificationObjective":null,
 "kForPrecision":null,
 "probabilityThreshold":null,
 "lossFunction":null,
 "smartSearch":null,
 "smartSearchIterations":null
},
"tuningresults":[
 {
  "gini": 0.774425957,
  "captureDeviation": 0.357559351,
  "precision": 0.84123771,
  "fScore": 0.680541246,
  "accuracy": 0.845201695,
  "giniProbThreshold": 0.29509603,
  "captureDeviationProbThreshold": 0.29509603,
  "precisionProbThreshold": 0.5744932,
  "fScoreProbThreshold": 0.29509603,
  "accuracyProbThreshold": 0.36550762,
  "samplingRatio": null,
```

```
   "numTrees": 10,
   "treeDepth": 3,
   "numDimensions": null,
   "leafNodes": 0,
   "learningRate": 0.1,
   "ensembleSize": null,
   "regularizationBins": -1
  },
  {
   "gini": 0.802989695,
   "captureDeviation": 0.216214067,
   "precision": 0.880878819,
   "fScore": 0.691668864,
   "accuracy": 0.851538036,
   "giniProbThreshold": 0.279287021,
   "captureDeviationProbThreshold": 0.279287021,
   "precisionProbThreshold": 0.674622582,
   "fScoreProbThreshold": 0.279287021,
   "accuracyProbThreshold": 0.419841869,
   "samplingRatio": null,
   "numTrees": 20,
   "treeDepth": 3,
   "numDimensions": null,
   "leafNodes": 0,
   "learningRate": 0.1,
   "ensembleSize": null,
   "regularizationBins": -1
  }
 ]
 "projectId":"24389108236",
 "method":"GBT"
}
```


Test a Model

This section show how to test a trained model against a dataset.

> **Note:** The dataset ID and model ID specified when testing a model must reside in the specified project. You will get an error if you attempt to reference datasets and/or models that are in a different project.

Request

Use POST to test a model against a dataset. In the example below, the model that we just trained, "1850798990," will be tested against dataset "4176208087."

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"gbt_model_test",
     "modelId":"1850798990",
     "dataSetId":"4176208087"
    }' 'http://api.skytree.net:8080/v1/gbt/test/'
```

Response

The API returns the ID of the newly created test. The return message will be similar to the following:

```
200 (OK)
"Content-type:application/json"

{
 "id:" "831718717",
 "modelId": "1850798990",
 "name": "gbt_model_test",
 "status":
 {
  "code": "READY",
  "message": "Test Result has been validated and is ready to use"
 },
 "dataSetId": "4176208087",
 "createdAt": 1401280660404,
 "updatedAt": 1401280660404,
 "testResult":[
  {
   "score": null,
   "points": 1,
   "confusionMatrix": 1,
   "accuracy": 0.994205,
   "recall": 0.974207,
   "falsePositiveRate": 0.115288,
   "precision": 0.978001,
   "fScore": 0.985067,
   "gini": 0.995849,
   "captureDeviation": 0.023764
 },
 ]
 "projectId":"24389108236",
 "method":"GBT",
 "type":"Classification"
}
```

Interrupt a Model

You may find it necessary to interrupt a model that is currently INPROGRESS. Note that you cannot interrupt models that have a status of READY.

Request

Use POST to interrupt the process running on a model. The model ID is required.

```
curl -X POST -H "x-auth-token:dXNlcjFAccGFzc3dvcmQy" \
-H "Content-type:application/json"  \
'http://api.skytree.net:8080/v1/models/1398605668/interrupt'
```

Response

The API returns "`true`" if the interrupt completed successfully. Otherwise the API returns "`false`" (for example, if the model was already created).

Retrieving a Single Model

Use GET to retrieve a single model based on the Model ID.

Request

The example below is a request to retrieve a model whose ID is 1398605668.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/models/1398605668'
```

Response

The response returns information about the specified model.

Retrieve All Models

Use GET without an ID to retrieve all available models.

Request

The following example is a request to retrieve all models. Note that GET is the default behavior and, thus, is not required.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/models'
```

Response

The response returns information for all models grouped by Model ID.

Retrieve All Models of a Specific Type

Use GET to retrieve all models generated by a specific method. Specify the method using the `type` parameter. If `type` is not specified, then all models will be returned.

Request

The example below shows a request to retrieve all models produced using RDF.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/models?type=rdf'
```

Response

The response returns information about all models created using RDF.

```
[
 {
  "id":"1398605668",
```

```
"name":"rdf_train_model",
"dataSetId":"1487422371",
"status":
{
 "code":"READY",
 "message":"Model has been validated and is ready to use"
},
"createdAt":1401198775613,
"updatedAt":1401198780667,
"configuration":
{
 "holdoutRatio":null,
 "numFolds":null,
 "objectiveColumn":"yearly-income",
 "sampleWithReplacement":null,
 "numTrees":
 {
  [2]
 },
 "treeDepth":null,
 "numDimensions":null,
 "holdoutSeed":null,
 "categoricalSelectionMethod":null,
 "categoricalRandomSplitTries":null,
 "categoricalRandomSplitThreshold":null,
 "dimensionSamplingSeed":null,
 "tableSamplingSeed":null,
 "categoricalSamplingSeed":null,
 "testingObjective":null,
 "maxSplits":null,
 "minNodeWeight":null
 "cardinalityBasedDimensionSampling":null,
 "smoothing":null,
 "imbalance":null,
 "imbalanceScale":null,
 "samplingRatio":null,
 "impurity":null,
 "splitCriterion":null,
 "probAggregationMethod":null,
 "classificationObjective":null,
 "kForPrecision":null,
 "probabilityThreshold":null,
 "classweight":null
},
"tuningresults":[
{
 "gini": 0.774425957,
 "captureDeviation": 0.357559351,
 "precision": 0.84123771,
 "fScore": 0.680541246,
 "accuracy": 0.845201695,
 "giniProbThreshold": 0.29509603,
 "captureDeviationProbThreshold": 0.29509603,
 "precisionProbThreshold": 0.5744932,
 "fScoreProbThreshold": 0.29509603,
 "accuracyProbThreshold": 0.36550762,
```

```
   "samplingRatio": null,
   "numTrees": 10,
   "treeDepth": 3,
   "numDimensions": null,
   "smoothing": null,
  },
  ...
  "method":"RDF"
 }
]
```

Retrieve All Models in a Project

Use GET along with a project ID to retrieve all available models in a project.

Request

The following example shows a request to retrieve all models in a project whose ID is 24389108236. Note that GET is the default behavior and, thus, is not required.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/models?projectId=24389108236'
```

Response

The response returns information for all models in the specified project grouped by Model ID.

Retrieve Model Dependents

Use this function to get a list of all downstream (dependent) files related to a model. This includes all test results and pots that were referenced using this model.

Request

Use GET along with a model ID to retrieve all files dependent on the model. The request below shows how to retrieve a list of files related to a model whose ID is 8613266560949776288.

```
curl -X GET -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/models/8613266560949776288/dependents'
```

Response

Upon successful completion, the response returns a list of files that are dependent on this dataset.

```
200 (OK)
"Content-type:application/json"

{
 "id":"8613266560949776288",
 "name":"GBT Model",
 "type":"MODEL",
 "is_deletable":TRUE
```

```
},
{
 "id":"733579628360679490",
 "name":"GBT Result",
 "type":"TESTRESULT",
 "is_deletable":TRUE
}
```

Delete a Model

Use `DELETE` to delete a single model based on the model ID.

Request

The example below is a request to delete a model with an ID of 1398605668.

```
curl -X DELETE -H "x-auth-token:dXNlcjFAccGFzc3dvcmQy" \
-H "Content-type:application/json" \
 'http://api.skytree.net/v1/models/1398605668'
```

Response

The API returns 204 `No Content` upon successful completion.

Model Parameters

A `projectId` is required for every model train request. You can also specify to retrieve all models in a specific project by including the `projectId` in the URL. The remaining options vary depending on the method being used. Refer to the following sections for a list of available configuration options:

- AutoModel Configuration Parameters (page 86)
- GBT Configuration Parameters (page 89)
- GBTR Configuration Parameters (page 93)
- GLMC Configuration Parameters (page 98)
- GLMR Configuration Parameters (page 101)
- RDF Configuration Parameters (page 104)
- RDFR Configuration Parameters (page 108)
- SVM Configuration Parameters (page 110)

AutoModel Configuration Parameters

The tables below describe the parameters that can be used when training and tuning models using AutoModel.

Table 30: *AutoModel Model Validation Parameters*

Parameter	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numbFolds`. If omitted, a time-based seed will be used.
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. For example, if the number of folds is 10 and the holdout ratio is 0.2, then a random 20% of the training data would be heldout in each fold as the validation set. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. When performing a classification problem, this column must be a categorical column with two unique values. When performing a regression problem, this column must include only numeric values.

Table 31: *Non-Tunable AutoModel Parameters*

Parameter	Description
`classificationObjective`	Optionally specify either "FSCORE" (default) or "ACCURACY" as the objective for classification threshold tuning.
`enablePartialDependencies`	Specify whether to export the partial dependencies of the relevant features and/or feature pairs in JSON format. These can be used to generate partial dependence plots.

`enablePmmlOut`	Specify whether to export trained models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`kForPrecision`	Specify *k* as a decimal value between 0 and 1 for precision at the top 100*k*-th percentile. This value defaults to `0.1`.

Table 31: *Non-Tunable AutoModel Parameters (continued)*

Parameter	Description
`limitParameters`	This option restricts the parameter space (for example, in `numTrees`, `treeDepth`, etc.) in order to reduce the training time. This option is enabled by default. Note that when GBT or GBTR is the method used to generate the model, `limitParameters` will set the following upper bound values: • `numTrees` <= 3000 • `treeDepth` <= 9 • `maxSplits` <= 256
`outputWithIds`	If enabled, the per-point output for labels, probabilities, and target points will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "0" is set as the input "`id`" field (for example, "`0,a,b,c`").
`smartSearchIterations`	Optionally specify the number of search rounds to try for tuning. This value must be greater than 0 and defaults to 100.
`smartSearchSeed`	Optionally specify the search seed. If omitted, a time-based seed will be used.
`testingObjective`	Specify one of the following objectives for selecting the test model from a set of tuned models. **Classification problem Options** ▪ `GINI` (default for Classification) ▪ `FSCORE` ▪ `ACCURACY` ▪ `CAPTURE_DEV` ▪ `PRECISION_AT_K` ▪ `YIELD` **Regression problem options** ▪ `NORMALIZED_GINI` ▪ `MEAN_ABSOLUTE_ERROR` (default for Regression) ▪ `MEAN_SQUARED_ERROR` ▪ `COEFF_DETERMINATION`

GBT Configuration Parameters

The tables below describe the parameters that can be used when training and testing models using GBT.

Table 32: *GBT Model Validation Parameters*

Parameter	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numFolds`. If omitted, a time-based seed will be used.
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. For classification problems, this column must be a categorical column with two unique values.
`smartSearch`	Specify to use an intelligent tuning technique (instead of naive grid search). When this flag is set to `True`, no other tuning parameters are required; only model validation options are expected. GBT will automatically search for the best parameter setting. Because Smart Search performs intelligent tuning, this option cannot be used with `probabilityThreshold`.
`smartSearchIterations`	If `smartSearch` is enabled, optionally specify the number of search rounds to try for tuning. This value must be greater than 0 and defaults to 100.

Table 33: *Tunable GBT Parameters specified as an array*

Parameter	Description
`classweight`	Specify decimal values for class weights. One class weight must be specified for each class in the dataset. The values are used to artificially inflate the impact of the corresponding class. If omitted, class weights of 1 are assumed for every class.
`ensembleSize`	Specify a list of integers. The ensemble GBT algorithm applies the concept of bootstrap aggregation, or “bagging”, to GBTs. The training table is sampled a number of times, and a new GBT is built for each sample.This option specifies the number of GBTs to build.

Table 33: *Tunable GBT Parameters specified as an array (continued)*

Parameter	Description
`imbalanceScale`	Specify a list of decimal values for the relative size of all classes with respect to the smallest class. This option is only available if imbalance is enabled. In this case, this value defaults to `1.0`.
`learningRate`	Specify a list of decimal values value for the learning rate for each iteration. This value defaults to `0.1` and must be > 0 and <= 1.
`maxSplits`	Specify the number of splits in the tree. This value must be >= 0. A value of 0 disables this option. Either this value or `minNodeWeight` must be > 0 if `treeDepth` is explicitly set to 0. If both `treeDepth` and are > 0, then Skytree finds the best splits that exist within the depth level, up to `maxSplits`.
`minNodeWeight`	Specify an array for the minimum bound on the total size in each leaf during tree building. This value defaults to 0. Either this value or `maxSplits` must be > 0 if `treeDepth` is explicitly set to 0.
`numDimensions`	Specify a list of integers for the number of dimensions to sample at each tree node. Only used in conjunction with `ensembleSize`. If omitted, all dimensions are used.
`numTrees`	This is required except when using smart search. In that case, this is optional. Specify a list of numbers to indicate the number of trees in the ensemble. This value must be > 0.s
`regularizationBins`	Specify a list of integers for the number of bins to use per dimension/attribute of the data. This option is only used with regularization. If `regularization` is enabled, then this value defaults to 200.
`samplingRatio`	Specify a list of doubles to indicate the per-class sampling ratio. You can use a single `samplingRatio` that will be used for each class, or specify `samplingRatio` values for each class. If a value is not specified and `sampleWithReplacement` is enabled, then full bootstrapping will be used.

Table 33: *Tunable GBT Parameters specified as an array (continued)*

Parameter	Description
`treeDepth`	Specify a list of integers for the depth to which each ensemble is built. If set to 0, trees will be built to the fullest extent. In this case, then you must also specify a positive value for either `minNodeWeight` or `maxSplits`. If this value is not set, and values for `maxSplits` and `minNodeWeight` are likewise not set, then this value defaults to 3. If this value and `maxSplits` are both > 0, then Skytree finds the best splits that exist within the depth level, up to `maxSplits`.

Table 34: *Non-Tunable GBT Parameters*

Parameter	Description
`cardinalityBasedDimensionSampling`	Categorical dimensions with larger number of unique values in a given node will have a proportionately higher chance of being selected when sampling dimensions. Continuous features are treated as having 1 unique value. When `False`, all dimensions have an equal chance of being selected in dimension sampling
`categoricalRandomSplitThreshold`	If the number of categorical values exceeds this integer value, then random splitting is attempted. Larger values will slow down training time considerably for exhaustive searches of all combinations of $O(2^{q-1})$. For binary classification problems, though, it is O(q) and, therefore, can be set to larger values. This value defaults to 4.
`categoricalRandomSplitTries`	Specify an integer for the number of times to try random categorical splits. This value defaults to 10.
`categoricalSamplingSeed`	Specify a long value for the categorical values sampling random number seed. If omitted, a time-based seed will be used.
`categoricalSelectionMethod`	Specify one of the following methods. • RANDOM • RANDOM_EXACT (default) • ONE_VS_ALL • ONE_VS_ALL_RANDOM • ONE_VS_ALL_EXACT • EXACT

Table 34: *Non-Tunable GBT Parameters (continued)*

Parameter	Description
`classificationObjective`	Optionally specify either "FSCORE" (default) or "ACCURACY" as the objective for classification threshold tuning. This option cannot be used with `probabilityThreshold`.
`dimensionSamplingSeed`	Specify a long value for the dimension sampling random number seed. If omitted, a time-based seed will be used.
`enablePartialDependencies`	Specify whether to export the partial dependencies of the relevant features and/or feature pairs in JSON format. These can be used to generate partial dependence plots. This option is available when: • Training along with (model save OR testing) • Tuning followed by training + (model save OR testing)

`enablePmmlOut`	Specify whether to export trained models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`imbalance`	When enabled, the API will attempt to improve classification for imbalanced classes. Note that this option cannot be used with `samplingRatio`.
`kForPrecision`	Specify *k* as a decimal value between 0 and 1 for precision at the top 100*k*-th percentile. This value defaults to `0.1`.
`lossFunction`	Specify the method to use when performing ranking for non-ensemble GBT. Options include the following: • `LOGISTIC` - Logistic ranking • `NDCG` - Normalized discounted cumulative gain • `MAP` - Mean Average Precision • `MRR` - Mean Reciprocal Ranking
`outputWithIds`	If enabled, the per-point output for labels and probabilities will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "0" is set as the input "`id`" field (for example, "`0,a,b,c`").
`probabilityThreshold`	Specify a decimal value for the probability to be used as the threshold for classification. This option cannot be used with `classificationObjective` or `testingObjective`. Similarly, because `probabilityThreshold` cannot be used during tuning, this option cannot be specified with `smartSearch`.

Table 34: *Non-Tunable GBT Parameters (continued)*

Parameter	Description
`quantiles`	When performing classification scoring, specify an integer value for the quantiles to use for the computation of the capture deviation. Defaults to deciles (10).
`regularization`	Specify to use the fast decision tree construction heuristic. This option defaults to `True` for `gbt`/`gbtr` and defaults to `False` for ensemble `gbt`/`gbtr`. When regularization is enabled, `regularizationBins` defaults to 200.
`sampleWithReplacement`	For each tree, sample the training points using replacement (bootstrap). This option is enabled by default. If set to `False`, then the `samplingRatio` must be provided.
`smartSearchSeed`	When `smartSearch` is enabled, optionally specified a seed value for smart search. If omitted, a time-based seed will be used by default.
`tableSamplingSeed`	Specify a long value for the data sampling random number seed. If omitted, a time-based seed will be used.
`testingObjective`	Specify one of the following objectives for selecting the test model from a set of tuned models. ▪ `GINI` (default) ▪ `FSCORE` ▪ `ACCURACY` ▪ `CAPTURE_DEV` ▪ `PRECISION_AT_K` ▪ `YIELD` This option cannot be used with `probabilityThreshold`.
`trim`	If enabled, observations with low importance in the model will be pruned in successive iterations. This can lead to significant speedup, but accuracy can suffer. This option defaults to `False`.
`trimAlpha`	Specify a decimal value > 0 and < 1. Higher values lead to more aggressive pruning of observations per iterations. If trim is enabled, then this value defaults to `0.2`.

GBTR Configuration Parameters

The tables below describe the parameters that can be used when training and testing models using GBTR.

Table 35: *GBTR Model Validation Parameters*

Parameter	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numFolds`. If omitted, a time-based seed will be used.
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. For example, if the number of folds is 10 and the holdout ratio is 0.2, then a random 20% of the training data would be held out in each fold as the validation set. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. For regression problems, this column must include only numeric values.
`smartSearch`	Specify to use an intelligent tuning technique (instead of naive grid search). When this flag is enabled, no other tuning parameters are required; only model validation options are expected. Skytree will automatically search for the best parameter setting. Because Smart Search performs intelligent tuning, this option cannot be used with `probabilityThreshold`.
`smartSearchIterations`	If `smartSearch` is enabled, optionally specify the number of search rounds to try for tuning. This value must be greater than 0 and defaults to 100.

Table 36: *Tunable GBTR Parameters specified as an array*

Parameter	Description
`ensembleSize`	Specify a list of integers. The ensemble GBT algorithm applies the concept of bootstrap aggregation, or “bagging”, to GBTs. The training table is sampled a number of times, and a new GBT is built for each sample.This option specifies the number of GBTs to build.
`learningRate`	Specify a list of decimal values value for the learning rate for each iteration. This value defaults to `0.1` and must be > 0 and <= 1.

Table 36: *Tunable GBTR Parameters specified as an array (continued)*

Parameter	Description
`logClamp`	When `lossFunction` is `PSLOG`, `GMLOG`, or `TDLOG`, optionally specify the maximum absolute log value for any terminal node. This value must be > 0 and defaults to 20.
`maxSplits`	Specify the number of splits in the tree. This value must be >= 0. A value of 0 disables this option. Either this value or `minNodeWeight` must be > 0 if `treeDepth` is explicitly set to 0. If both `treeDepth` and `maxSplits` are > 0, then Skytree finds the best splits that exist within the depth level, up to `maxSplits`.
`minNodeWeight`	Specify an array for the minimum bound on the total size in each leaf during tree building. This value defaults to 0. Either this value or `maxSplits` must be > 0 if `treeDepth` is explicitly set to 0.
`numDimensions`	Specify a list of integers for the number of dimensions to sample at each tree node. Only used in conjunction with `ensembleSize`. If omitted, all dimensions are used.
`numTrees`	This is required except when using smart search. In that case, this is optional. Specify a list of numbers to indicate the number of trees in the ensemble. This value must be > 0.
`regularizationBins`	Specify a list of integers for the number of bins to use per dimension/attribute of the data. This option is only used with regularization. If `regularization` is enabled, then this value defaults to 200.
`samplingRatio`	Specify a list of doubles to indicate the sampling ratio. You can use a single `samplingRatio` that will be used for each class, or specify `samplingRatio` values for each class. If a value is not specified and `sampleWithReplacement` is enabled, then full bootstrapping will be used.
`treeDepth`	Specify a list of integers for the depth to which each ensemble is built. If set to 0, trees will be built to the fullest extent. In this case, then you must also specify a positive value for either `minNodeweight` or `maxSplits`. If this value is not set, and values for `maxSplits` and `minNodeweight` are likewise not set, then this value defaults to 3. If this value and `maxSplits` are both > 0, then Skytree finds the best splits that exist within the depth level, up to `maxSplits`.

Table 36: *Tunable GBTR Parameters specified as an array (continued)*

Parameter	Description
`tweedieExponent`	When `lossFunction` is `TDLOG`, specify a value for the exponent. This option has no default value and, therefore, must be explicitly specified when configuring the Tweedie loss function. This value must be > `1.0` and < `2.0`. If users desire a value of 1, they should use Poisson distribution (`PSLOG`). Similarly, if users desire an exponent equal to `2.0`, they should specify Gamma distribution (`GMLOG`).

Table 37: *Non-Tunable GBTR Parameters*

Parameter	Description
`cardinalityBasedDimensionSampling`	Categorical dimensions with larger number of unique values in a given node will have a proportionately higher chance of being selected when sampling dimensions. Continuous features are treated as having 1 unique value. When set to `False`, all dimensions have an equal chance of being selected in dimension sampling
`categoricalRandomSplitThreshold`	If the number of categorical values exceeds this integer value, then random splitting is attempted. Larger values will slow down training time considerably for exhaustive searches of all combinations of $O(2^{q-1})$. For binary classification problems, though, it is O(q) and, therefore, can be set to larger values. This value defaults to 4.
`categoricalRandomSplitTries`	Specify an integer for the number of times to try random categorical splits. This value defaults to 10.
`categoricalSamplingSeed`	Specify a long value for the categorical values sampling random number seed. If omitted, a time-based seed will be used.
`categoricalSelectionMethod`	Specify one of the following methods. • RANDOM • RANDOM_EXACT (default) • ONE_VS_ALL • ONE_VS_ALL_RANDOM • ONE_VS_ALL_EXACT • EXACT
`dimensionSamplingSeed`	Specify a long value for the dimension sampling random number seed. If omitted, a time-based seed will be used.

Table 37: *Non-Tunable GBTR Parameters (continued)*

Parameter	Description
`enablePartialDependencies`	Specify whether to export the partial dependencies of the relevant features and/or feature pairs in JSON format. These can be used to generate partial dependence plots. This option is available when: • Training along with (model save OR testing) • Tuning followed by training + (model save OR testing)

`enablePmmlOut`	Specify whether to export trained models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`huberLossQuantile`	When `lossFunction` is `HUBER`, specify the top percentile of error that should be considered as outliers. This value must be between `0` and `1` and defaults to `0.9`.
`imbalance`	When set to `True`, the API will attempt to improve classification for imbalanced classes. Note that this option cannot be used with `samplingRatio`.
`lossFunction`	Optionally specify a loss function method to use during regression. Available values include the following: • `LAD` - Perform least absolute deviation (LAD) regression • `LS` - Perform least-squares regression • `HUBER` - Perform Huber loss regression • `PSLOG` - Perform Poisson-log regression with positive integral targets (such as counts) • `GMLOG` - Perform Gamma-log regression with positive continuous targets • `GMINV` - Perform Gamma-inverse regression with positive continuous targets • `TDLOG` - Perform Tweedie-log regression If `HUBER` is specified, then you must also define a value for `huberLossQuantile`. Similarly, if `TDLOG` is specified, then you must also define a value for `tweedieExponent`.
`outputWithIds`	If enabled, the per-point output for target will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "`0`" is set as the input "`id`" field (for example, "`0,a,b,c`").

Table 37: *Non-Tunable GBTR Parameters (continued)*

Parameter	Description
`regularization`	Specify to use the fast decision tree construction heuristic. This option defaults to `True` for `gbt`/`gbtr` and defaults to `False` for ensemble `gbt`/`gbtr`. When regularization is enabled, `regularizationBins` defaults to 200.
`sampleWithReplacement`	For each tree, sample the training points using replacement (bootstrap). This option is enabled by default. If turned off, then the `samplingRatio` must be provided.
`smartSearchSeed`	When `smartSearch` is enabled, optionally specified a seed value for smart search. If omitted, a time-based seed will be used by default.
`tableSamplingSeed`	Specify a long value for the data sampling random number seed. If omitted, a time-based seed will be used.
`testingObjective`	Specify one of the following objectives for selecting the test model from a set of tuned models. ▪ `NORMALIZED_GINI` ▪ `MEAN_ABSOLUTE_ERROR` (default) ▪ `MEAN_SQUARED_ERROR` ▪ `COEFF_DETERMINATION`

GLMC Configuration Parameters

The tables below describe the parameters that can be used when training and testing models using GLMC.

Table 38: *GLMC Model Validation Parameters*

Parameter	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numFolds`. If omitted, a time-based seed will be used.

Table 38: *GLMC Model Validation Parameters (continued)*

Parameter	Description
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. For example, if the number of folds is 10 and the holdout ratio is 0.2, then a random 20% of the training data would be held out in each fold as the validation set. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. For classification problems, this column must be a categorical column with two unique values.
`smartSearch`	Specify to use an intelligent tuning technique (instead of naive grid search). When this flag is enabled, no other tuning parameters are required; only model validation options are expected. Skytree will automatically search for the best parameter setting. Because Smart Search performs intelligent tuning, this option cannot be used with `probabilityThreshold`.
`smartSearchIterations`	If `smartSearch` is enabled, optionally specify the number of search rounds to try for tuning. This value must be greater than 0 and defaults to 100.

Table 39: *Tunable GLMC Parameters specified as an array*

Parameter	Description
`l1Penalty`	Specify the penalty value for L1 (LASSO) regularization. This value must be >= 0.0. Specifying both L1 and L2 values > 0 will result in Elastic Net regularization.
`l2Penalty`	Specify the penalty value for L2 (Ridge) regularization. This value must be > = 0.0 and defaults to 1.0. Specifying both L1 and L2 values > 0 will result in Elastic Net regularization.

Table 40: *Non-Tunable GLMC Options*

Command	Description
`classificationObjective`	Optionally specify either "FSCORE" (default) or "ACCURACY" as the objective for classification threshold tuning. This option cannot be used with `probabilityThreshold`.

Table 40: *Non-Tunable GLMC Options (continued)*

Command	Description
`enablePartialDependencies`	Specify whether to export the partial dependencies of the relevant features and/or feature pairs in JSON format. These can be used to generate partial dependence plots. This option is available when: • Training along with (model save OR testing) • Tuning followed by training + (model save OR testing)

`enablePmmlOut`	Specify whether to export trained models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`epsilon`	Specify the numerical accuracy to which to train the GLMC model. This value must be > 0.0 and defaults to 0.001.
`excludeBiasTerm`	If set, the GLMC formulation will not have a bias term. A bias term is used by default.
`kForPrecision`	Specify *k* as a decimal value between 0 and 1 for precision at the top 100*k*-th percentile. This value defaults to `0.1`.
`link`	Specify the link function to use for the GLM model. Options include: • `LOGIT` (default): logistic • `CLOGLOG`: complementary log-log
`maxCacheMemory`	Specify the maximum allowed memory, in megabytes, used to cache the rows of the kernel matrix (to reduce recomputation). This value defaults to 500MB.
`maxIterations`	Specify the maximum allowed number of iterations for the training algorithm. This value must be > 0. For GLMC training, this corresponds to the number of passes over the data.
`outputWithIds`	If enabled, the per-point output for target points will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "0" is set as the input "`id`" field (for example, "`0,a,b,c`").
`probabilityThreshold`	Specify a decimal value for the probability to be used as the threshold for classification. This option cannot be used with `classificationObjective` or `testingObjective`. Similarly, because `probabilityThreshold` cannot be used during tuning, this option cannot be specified with `smartSearch`.
`tableSamplingSeed`	Specify the data sampling random number seed. If omitted, a time-based seed will be used.

Table 40: *Non-Tunable GLMC Options (continued)*

Command	Description
`testingObjective`	Specify one of the following objectives for selecting the test model from a set of tuned models. ▪ `GINI` (default) ▪ `FSCORE` ▪ `ACCURACY` ▪ `CAPTURE_DEV` ▪ `PRECISION_AT_K` ▪ `YIELD` This option cannot be used with `probabilityThreshold`.

GLMR Configuration Parameters

The tables below describe the parameters that can be used when training and testing models using GLMR.

Table 41: *GLMR Model Validation Parameters*

Parameter	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numFolds`. If omitted, a time-based seed will be used.
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. For example, if the number of folds is 10 and the holdout ratio is 0.2, then a random 20% of the training data would be held out in each fold as the validation set. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. For regression problems, this column must include only numeric values.

Table 41: *GLMR Model Validation Parameters (continued)*

Parameter	Description
`smartSearch`	Specify to use an intelligent tuning technique (instead of naive grid search). When this flag is enabled, no other tuning parameters are required; only model validation options are expected. Skytree will automatically search for the best parameter setting.
`smartSearchIterations`	If `smartSearch` is enabled, optionally specify the number of search rounds to try for tuning. This value must be greater than 0 and defaults to 100.

Table 42: *Tunable GLMR Parameters specified as an array*

Parameter	Description
`l1Penalty`	Specify the penalty value for L1 (LASSO) regularization. This value must be >= 0.0. Specifying both L1 and L2 values > 0 will result in Elastic Net regularization.
`l2Penalty`	Specify the penalty value for L2 (Ridge) regularization. This value must be > = 0.0 and defaults to 1.0. Specifying both L1 and L2 values > 0 will result in Elastic Net regularization.
`tweedieExponent`	When `family=TWEEDIE`, specify a value for the exponent. This option has no default value and, therefore, must be explicitly specified when configuring the Tweedie loss function. This value must be > `1.0` and < `2.0`. If users desire a value of 1, they should specify the Poisson family and Log link function. Similarly, if users desire an exponent equal to `2.0`, they should specify Gamma family along with the Log link function. Note that users can also tune over this option with values > `1.0` and < `2.0`. When used for tuning, the output CSV will have an extra column for this value.

Table 43: *Non-Tunable GLMR Options*

Command	Description
`enablePartialDependencies`	Specify whether to export the partial dependencies of the relevant features and/or feature pairs in JSON format. These can be used to generate partial dependence plots. This option is available when: • Training along with (model save OR testing) • Tuning followed by training + (model save OR testing)

Table 43: *Non-Tunable GLMR Options (continued)*

Command	Description

`enablePmmlOut`	Specify whether to export trained models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`epsilon`	Specify the numerical accuracy to which to train the GLMR model. This value must be > 0.0 and defaults to 0.001.
`excludeBiasTerm`	If set, the GLMR formulation will not have a bias term. A bias term is used by default. **Note**: This option cannot be used with the following family-link pairs: Gamma-Inverse, Gamma-Canonical, Poisson-Identity, or Poisson-Sqrt.
`family`	Specify the family function to use for a GLMR model. Options include: • `GAMMA` • `GAUSSIAN` • `POISSON` • `TWEEDIE`
`link`	Specify the link function to use for the GLMR model. Options include: • `CANONICAL`. This can be use with any `family` option. • `IDENTITY`. This can be specified if `family` is `GAUSSIAN` or `POISSON`. • `INVERSE`. This can be specified if `family` is `GAMMA`. • `LOG`. This can be specified if `family` is `GAMMA`, `POISSON`, or `TWEEDIE`. • `SQRT`. This can be specified if `family` is `POISSON`.
`maxCacheMemory`	Specify the maximum allowed memory, in megabytes, used to cache the rows of the kernel matrix (to reduce recomputation). This value defaults to 500MB.
`maxIterations`	Specify the maximum allowed number of iterations for the training algorithm. This value must be > 0. For GLMR training, this corresponds to the number of passes over the data.
`outputWithIds`	If enabled, the per-point output for target points will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "0" is set as the input "`id`" field (for example, "`0,a,b,c`").
`tableSamplingSeed`	Specify the data sampling random number seed. If omitted, a time-based seed will be used.

Table 43: *Non-Tunable GLMR Options (continued)*

Command	Description
`testingObjective`	Optionally specify the objective for selecting the test model from a set of tuned models. Specify one of the following: • `NORMALIZED_GINI` • `MEAN_ABSOLUTE_ERROR` (default) • `MEAN_SQUARED_ERROR` • `COEFF_DETERMINATION`

RDF Configuration Parameters

The tables below describe the parameters that can be used when training and testing models using RDF.

Table 44: *RDF Model Validation Parameters*

Parameter	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numFolds`. If omitted, a time-based seed will be used.
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. For example, if the number of folds is 10 and the holdout ratio is 0.2, then a random 20% of the training data would be heldout in each fold as the validation set. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. For classification problems, this column must be a categorical column with two unique values.

Table 45: *Tunable RDF Parameters specified as an array*

Parameter	Description
`classweight`	Specify decimal values for class weights. One class weight must be specified for each class in the dataset. The values are used to artificially inflate the impact of the corresponding class. If omitted, class weights of 1 are assumed for every class.

Table 45: *Tunable RDF Parameters specified as an array (continued)*

Parameter	Description
`imbalanceScale`	Specify a list of decimal values for the relative size of all classes with respect to the smallest class. This option is only available if imbalance is enabled. In this case, this value defaults to `1.0`.
`numDimensions`	Specify a list of integers for the number of dimensions to sample at each node. If omitted, (log2(*D*) + 1) is used, where *D* is the number of attributes.
`numTrees`	Required. Specify a list of numbers to indicate the number of trees in the ensemble. This value must be > 0.
`samplingRatio`	Specify a list of doubles to indicate the per-class sampling ratio. You can use a single `samplingRatio` that will be used for each class, or specify `samplingRatio` values for each class. If a value is not specified and `sampleWithReplacement` is enabled, then full bootstrapping will be used.
`smoothing`	Specify the probability smoothing parameter. This value must be >= 0. Note that specifying 0 disables this feature.
`treeDepth`	Specify the depth to which each ensemble is built. If set to 0, trees will be built to their fullest extent.

Table 46: *Non-Tunable RDF Parameters*

Parameter	Description
`cardinalityBasedDimensionSampling`	Categorical dimensions with larger number of unique values in a given node will have a proportionately higher chance of being selected when sampling dimensions. Continuous features are treated as having 1 unique value. When `False`, all dimensions have an equal chance of being selected in dimension sampling
`categoricalRandomSplitThreshold`	If the number of categorical values exceeds this integer value, then random splitting is attempted. Larger values will slow down training time considerably for exhaustive searches of all combinations of $O(2^{q-1})$. For binary classification problems, though, it is O(q) and, therefore, can be set to larger values. This value defaults to 4.
`categoricalRandomSplitTries` `categorical_random_split_tries`	Specify an integer for the number of times to try random categorical splits. This value defaults to 10.
`categoricalSamplingSeed`	Specify a long value for the categorical values sampling random number seed. If omitted, a time-based seed will be used.

Table 46: *Non-Tunable RDF Parameters (continued)*

Parameter	Description
`categoricalSelectionMethod`	Specify one of the following methods. • RANDOM • RANDOM_EXACT (default) • ONE_VS_ALL • ONE_VS_ALL_RANDOM • ONE_VS_ALL_EXACT • EXACT
`classificationObjective`	Optionally specify either "FSCORE" (default) or "ACCURACY" as the objective for classification threshold tuning. This option cannot be used with `probabilityThreshold`.
`curve`	When performing classification scoring, this defines the type of curve used for AUC/GINI calculation. Specify one of the following: • `roc`: Receiver Operations Curve (ROC) (default): x-axis: false positive rate y-axis: true positive rate (capture rate) • `lorenz`: Lorenz curve: x-axis: percentile y-axis: true positive rate (capture rate)
`dimensionSamplingSeed`	Specify a long value for the dimension sampling random number seed. If omitted, a time-based seed will be used.

`enablePmmlOut`	Specify whether to export trained models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`imbalance`	When set to `True`, the API will attempt to improve classification for imbalanced classes. Note that this option cannot be used with `samplingRatio`.
`impurity`	Specify the type of impurity to use when generating the split. Available options include `ENTROPY` (default), `GINI`, and `MISCLASSIFICATION`.
`kForPrecision`	Specify *k* as a decimal value between 0 and 1 for precision at the top 100*k*-th percentile. This value defaults to `0.1`.

Table 46: *Non-Tunable RDF Parameters (continued)*

Parameter	Description
`outputWithIds`	If enabled, the per-point output for labels and probabilities will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "0" is set as the input "`id`" field (for example, "`0,a,b,c`").
`probAggregationMethod`	Specify the method used for aggregation of individual tree predictions. Specify either "VOTE" or "AVERAGE" (default).
`probabilityThreshold`	Specify a decimal value for the probability to be used as the threshold for classification. This option cannot be used with `classificationObjective` or `testingObjective`. Similarly, because `probabilityThreshold` cannot be used during tuning, this option cannot be specified with `smartSearch`.
`quantiles`	When performing classification scoring, specify an integer value for the quantiles to use for the computation of the capture deviation. Defaults to deciles (10).
`sampleWithReplacement`	For each tree, sample the training points using replacement (bootstrap). This option is set to `True` by default. If set to `False`, then the `samplingRatio` must be provided.
`splitCriterion`	Specify the method to use as the measure for split criterion. Optional values include the following: • `INFORMATION_GAIN` • `INFORMATION_GAIN_RATIO` (default)
`tableSamplingSeed`	Specify a long value for the data sampling random number seed. If omitted, a time-based seed will be used.
`testingObjective`	Specify one of the following objectives for selecting the test model from a set of tuned models. ▪ `GINI` (default) ▪ `FSCORE` ▪ `ACCURACY` ▪ `CAPTURE_DEV` ▪ `PRECISION_AT_K` ▪ `YIELD` This option cannot be used with `probabilityThreshold`.

RDFR Configuration Parameters

The tables below describe the parameters that can be used when training and testing models using RDFR.

Table 47: *RDFR Model Validation Parameters*

Parameter	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numFolds`. If omitted, a time-based seed will be used.
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. For example, if the number of folds is 10 and the holdout ratio is 0.2, then a random 20% of the training data would be held out in each fold as the validation set. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. For regression problems, this column must include only numeric values.

Table 48: *Tunable RDFR Parameters specified as an array*

Parameter	Description
`minNodeWeight`	Specify an array for the minimum bound on the total size in each leaf during tree building. This value defaults to 2.
`numDimensions`	Specify a list of integers for the number of dimensions to sample at each node. . By default, RDFR used (D/3) dimensions, where D is the total number of attributes in the dataset.
`numTrees`	Required. Specify a list of numbers to indicate the number of trees in the ensemble. This value must be > 0.
`samplingRatio`	Specify a list of doubles to indicate the per-class sampling ratio. You can use a single `samplingRatio` that will be used for each class, or specify `samplingRatio` values for each class. If a value is not specified and `sampleWithReplacement` is enabled, then full bootstrapping will be used.
`treeDepth`	Specify a list of integers for the depth to which each ensemble is built. If set to 0, trees will be built to the fullest extent.

Table 49: *Non-Tunable RDFR Parameters*

Parameter	Description
`cardinalityBasedDimensionSampling`	Categorical dimensions with larger number of unique values in a given node will have a proportionately higher chance of being selected when sampling dimensions. Continuous features are treated as having 1 unique value. When set to `False`, all dimensions have an equal chance of being selected in dimension sampling
`categoricalRandomSplitThreshold`	If the number of categorical values exceeds this integer value, then random splitting is attempted. Larger values will slow down training time considerably for exhaustive searches of all combinations of $O(2^{q-1})$. For binary classification problems, though, it is O(q) and, therefore, can be set to larger values. This value defaults to 4.
`categoricalRandomSplitTries`	Specify an integer for the number of times to try random categorical splits. This value defaults to 10.
`categoricalSamplingSeed`	Specify a long value for the categorical values sampling random number seed. If omitted, a time-based seed will be used.
`categoricalSelectionMethod`	Specify one of the following methods. • RANDOM • RANDOM_EXACT (default) • ONE_VS_ALL • ONE_VS_ALL_RANDOM • ONE_VS_ALL_EXACT • EXACT
`dimensionSamplingSeed`	Specify a long value for the dimension sampling random number seed. If omitted, a time-based seed will be used.

`enablePmmlOut`	Specify whether to export trained models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`outputWithIds`	If enabled, the per-point output for target points will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "0" is set as the input "`id`" field (for example, "`0,a,b,c`").
`tableSamplingSeed`	Specify a long value for the data sampling random number seed. If omitted, a time-based seed will be used.

Table 49: *Non-Tunable RDFR Parameters (continued)*

Parameter	Description
`testingObjective`	Specify one of the following objectives for selecting the test model from a set of tuned models. ▪ `NORMALIZED_GINI` ▪ `MEAN_ABSOLUTE_ERROR` (default) ▪ `MEAN_SQUARED_ERROR` ▪ `COEFF_DETERMINATION`

SVM Configuration Parameters

The tables below describes the parameters that can be specified when training and testing models using SVM.

Table 50: *SVM Model Validation Parameters*

Command	Description
`holdoutRatio`	Specify a decimal value for the fraction of the table to be held out for tuning. If this option is specified, a collection of models will be built and evaluated on the held-out tuning set, then the parameters from the best model will be used to train a model on the entire training dataset.
`holdoutSeed`	Optionally specify a long value for the holdout sampling random number seed. This value also serves as the seed for `numFolds`. If omitted, a time-based seed will be used.
`numFolds`	Specify an integer for the number of folds for k-fold cross-validation. If `holdoutRatio` is also specified, then Monte Carlo cross-validation (with randomly drawn holdout sets) is instead performed. If this option is specified, a collection of models will be built and evaluated, then the parameters from the best model will be used to train a model on the entire training dataset.
`objectiveColumn`	Specify the column in the dataset that you are attempting to predict. For classification problems, this column must be a categorical column with two unique values.
`smartSearch`	Specify to use an intelligent tuning technique (instead of naive grid search). When this flag is set to `True`, no other tuning parameters are required; only model validation options are expected. SVM will automatically search for the best parameter setting.
`smartSearchIterations`	If `smartSearch` is enabled, optionally specify the number of search rounds to try for tuning. This value must be greater than 0 and defaults to 100.

Table 51: *SVM Tunable Parameters*

Command	Description
`lambda`	The regularization parameter in the SVM formulation. This option is required when training or tuning without smart search. If `smartSearch` is enabled, then this value is optional.
`polynomialDegree`	Specify the degree of the polynomial kernel. This option is required when training or tuning with a kernel type specified as `POLYNOMIAL`.
`polynomialOffset`	Specify the offset of the polynomial kernel. This can only be used when the kernel type is specified as `POLYNOMIAL`. Must be >= 0. This value defaults to 0.
`polynomialScale`	Specify the scale of the polynomial kernel. This can only be used when the kernel type is specified as `POLYNOMIAL`. Must be >= 0. This value defaults to `1.0`.
`rbfBandwidth`	Specify the bandwidth of the RBF kernel. Must be >= 0. This option is required when training or tuning with a kernel type specified as `RBF`.

Table 52: *SVM Non-Tunable Parameters*

Command	Description
`classificationObjective`	Optionally specify either "`FSCORE`" (default) or "`ACCURACY`" as the objective for classification threshold tuning. This option cannot be used with `probabilityThreshold`.
`enablePartialDependencies`	Specify whether to export the partial dependencies of the relevant features and/or feature pairs in JSON format. These can be used to generate partial dependence plots. This option is available when: • Training along with (model save OR testing) • Tuning followed by training + (model save OR testing)

`enablePmmlOut`	Specify whether to export trained SVM models into PMML (Predictive Model Markup Language) format. This allows the use of an external code for scoring. When set to TRUE, this creates an endpoint that can be accessed using:
	`/<id>/pmml (content-type: text/xml)`
	`/<id>/pmml/gzip (content-type:application/gzip)`

`epsilon`	Specify the numerical accuracy to which to train the SVM. This defaults to 1e-3.
`excludeBiasTerm`	If enabled, the SVM formulation will not have a bias term. A bias term is used by default.
`kForPrecision`	Specify *k* as a decimal value between 0 and 1 for precision at the top 100*k*-th percentile. This value defaults to `0.1`.

Table 52: *SVM Non-Tunable Parameters (continued)*

Command	Description
`kernel`	Specify the kernel function to be used for the SVM. Specify `LINEAR` (default), `RBF` (Gaussian), or `POLYNOMIAL`. **Note**: An SVM smart search run is based on a specified kernel type. If `kernel` is not specified, smart search will run using type `LINEAR`. If a different kernel function is desired in smart search, then it must be explicitly specified.
`maxCacheMemory`	Specify the maximum allowed memory, in megabytes, used to cache the rows of the kernel matrix (to reduce recomputation). This value defaults to 500MB for nonlinear SVM.
`maxIterations`	Specify the maximum allowed number of iterations for the training algorithm. This value must be > 0. For linear SVM training, this corresponds to the number of passes over the data.
`outputWithIds`	If enabled, the per-point output for label points will be prepended with the input file's "`id`" meta field followed by a comma. For example, if the original output is "`a,b,c`" then the new output would be "`ID,a,b,c`" where "`ID`" is an integer (`1, -1`). If an "`id`" meta field is not available in the input, then "0" is set as the input "`id`" field (for example, "`0,a,b,c`").
`probabilityThreshold`	Specify a decimal value for the probability to be used as the threshold for classification. This option cannot be used with `classificationObjective` or `testingObjective`. Similarly, because `probabilityThreshold` cannot be used during tuning, this option cannot be specified with `smartSearch`.
`smartSearchSeed`	When `smartSearch` is enabled, optionally specified a seed value for smart search. If omitted, a time-based seed will be used by default.
`tableSamplingSeed`	Specify a long value for the data sampling random number seed. If omitted, a time-based seed will be used.
`testingObjective`	Specify one of the following objectives for selecting the test model from a set of tuned models. ▪ `GINI` (default) ▪ `FSCORE` ▪ `ACCURACY` ▪ `CAPTURE_DEV` ▪ `PRECISION_AT_K` ▪ `YIELD` This option cannot be used with `probabilityThreshold`.

Model Examples

This section includes usage examples for creating models using the available methods. Refer to the following sections for more information:

AutoModel Examples

This section includes the following API usage examples for creating AutoModel models.

Classification with AutoModel

The following example shows how to build a simple model. The AutoModel module automatically recognizes this as a classification problem because the `objectiveColumn` is categorical rather than continuous.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"automodel_classification",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[100],
      "objectiveColumn": "yearly-income",
      "holdOutRatio":0.3
     }
    }' 'http://api.skytree.net:8080/v1/automodel/train'
```

Regression with AutoModel

The following example shows how to build a simple model. The AutoModel module automatically recognizes this as a regression problem because the `objectiveColumn` is continuous rather than categorical.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
```

```
-d '{
     "projectId":"24389108236",
     "name":"automodel_regression",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[[100]],
      "numberOfFolds":10,
      "objectiveColumn":"capital-gain"
     }
    }' 'http://api.skytree.net:8080/v1/automodel/train'
```

GBT Model Examples

This section includes the following advanced API usage examples for creating GBT models.

- Tuning the Number of Trees with a Holdout Set (page 114)
- Tuning the Tree Depth and Learning Rate (page 114)

Tuning the Number of Trees with a Holdout Set

This example specifies to hold out 20% of the training data for use in evaluating the model. The default behavior is to score the model using the tuning set after every 10% of iterations (10 in this case). Thus, the model will build 10 trees and score on the holdout data. Then it will go on to build 20 trees and score on the holdout data and so on, reporting accuracy scores each step of the way.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"gbt_numberoftrees_holdout",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numTrees":[100],
      "objectiveColumn":"yearly-income",
      "holdoutRatio":0.2
     }
    }' 'http://api.skytree.net:8080/v1/gbt/train'
```

Tuning the Tree Depth and Learning Rate

Train the Model

The example below builds a GBT model by setting the tree depth and the learning rate. In addition, the model is specified to be available in PMML format.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
```

```
    "projectId":"24389108236",
    "name":"gbt_treedepth_learnrate",
    "dataSetId":"1483908776",
    "configuration":
    {
     "numTrees":[[5],[5],[25]],
     "objectiveColumn":"yearly-income",
     "holdoutRatio":0.2,
     "treeDepth":[[2],[3]],
     "learningRate":[[0.05],[0.05],[0.2]],
     "enablePmmlOut":TRUE
    }
   }' 'http://api.skytree.net:8080/v1/gbt/train'
```

Retrieve PMML Output

Upon completion, the PMML output of the model can be retrieved using the following request:

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:text/xml" \
'http://api.skytree.net:8080/v1/models/2794736388168281/pmml'
```

Note: You can also retrieve PMML output in `gzip` format. in this case, specify "`Content-type:application/gzip`" and add `/gzip` to the end of the URL.

```
 -H "Content-type:application/gzip" \
 'http://api.skytree.net:8080/v1/models/2794736388168281/pmml/gzip'
```

GBTR Model Examples

This section includes the following advanced API usage examples for creating GBTR models.

- Tuning with a Loss Function (page 115)
- Tuning the Number of Trees using K-Fold Cross Validation (page 116)

Tuning with a Loss Function

In GBTR, the default loss function method performs least absolute deviation (LAD). Certain cases can exist, however, in which this loss function can lead to poor results. The `lossFuction` option allows you to specify a different method to use when performing regression. The example below specifies to use the least squares (LS) loss function.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"gbtr_ls_lossfunction",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[[5],[15],[25]],
      "objectiveColumn":"age",
```

```
     "holdOutRatio":0.2,
     "lossFunction":LS
    }
   }' 'http://api.skytree.net:8080/v1/gbtr/train'
```

Tuning the Number of Trees using K-Fold Cross Validation

Tuning can also be done using K-fold cross validation. This is better than using a single hold out set which is randomly generated. The downside is that it can be K times slower. This example uses the `numberOfFolds` option to return the best average results over all folds. This method can be used to fine tune parameters after honing in on approximate parameters using a holdout set.

Note: If both `numberOfFolds` and `holdOutRatio` are provided in a single command, then the algorithm will not do K-fold cross validation but instead repeat `numberOfFolds` times with a new holdout tuning set and return the parameters with the best average results over these `numberOfFolds` runs.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"gbtr_kfold",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[[5],[15],[25]],
      "objectiveColumn":"age",
      "numberOfFolds":5
     }
    }' 'http://api.skytree.net:8080/v1/gbtr/train'
```

GLM Model Examples

This section includes the following advanced API usage examples for creating GLM models.

- Training a Gaussian-Identity Model (Regression)
- Excluding the Bias Term (Classification)
- Tuning an Unbiased Poisson-Log Model (Regression)

Training a Gaussian-Identity Model

The following example trains a model using the guassian-identity Family/Link Function pair.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"glmr_gaussian_identity",
     "dataSetId":"1483908776",
```

```
    "configuration":
    {
     "objectiveColumn":"age",
     "maxIterations":"1000",
     "l1Penalty":[0.01],
     "l2Penalty":[0],
     "epsilon":"0.001",
     "family":"GAUSSIAN",
     "link":"IDENTITY"
    }
   }' 'http://api.skytree.net:8080/v1/glmr/train'
```

Excluding the Bias Term

By default, a bias term is included in the computed model. The bias term adjusts all predictions up or down by a constant amount, i.e. it is the predicted value when all inputs are exactly 0. It can be excluded (forced equal 0) with `"excludeBiasTerm": true`.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
    "projectId":"24389108236",
    "name":"glmc_exclude_bias",
    "dataSetId":"1483908776",
    "configuration":
    {
     "objectiveColumn":"yearly-income",
     "excludeBiasTerm": true,
     "l1Penalty":0.1,
     "l2Penalty":0,
     "epsilon":"0.001",
     "maxIterations":"1000",
     "link":"CLOGLOG"
    }
   }' 'http://api.skytree.net:8080/v1/glmc/train'
```

In GLM, the inverse of the link function is applied to obtain the final predictions (targets in GLMR and probabilities in GLMC). Excluding the bias term can negatively impact measured model quality, but it is appropriate when a given linear model should definitely predict 0 when all inputs are 0 (before the application of the inverse of the link function to obtain the final predictions). You may also want to exclude the bias/intercept term to choose a simpler model when appropriate and move away from overfitting.

> **Note:** This option cannot be used with the following family/link combinations: poisson/identity, poisson/sqrt, and gamma/inverse.

Tuning an Unbiased Poisson-Log Model

The following example tunes an unbiased GLMR model using the poisson-log Family/Link Function pair.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
```

```
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"glmr_tune_unbiased",
     "dataSetId":"1483908776",
     "configuration":
     {
      "objectiveColumn":"age",
      "excludeBiasTerm": true,
      "l1Penalty":[0,0.001,0.01,0.1],
      "l2Penalty":[0,0.001,0.01,0.1],
      "epsilon":"0.001",
      "family":"POISSON",
      "link":"LOG"
     }
    }' 'http://api.skytree.net:8080/v1/glmr/train'
```

RDF Model Examples

This section includes the following advanced API usage examples for creating RDF models.

- Tuning the Number of Dimensions (page 118)
- Tuning for Skewed Data Automatically (page 118)

Tuning the Number of Dimensions

This example shows how to tune the number of randomly selected dimensions considered in splitting each tree node. By default, RDF uses $\log_2(D + 1)$ dimensions, where D is the total number of attributes in the dataset.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"rdf_numdimensions",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[100],
      "objectiveColumn":"yearly-income",
      "holdOutRatio":0.2,
      "numberOfDimensions":[[1],[2],[4]]
     }
    }' 'http://api.skytree.net:8080/v1/rdf/train'
```

Tuning for Skewed Data Automatically

Often, the class distribution in the data is skewed, with far fewer points in one of the classes. Skytree provides a `samplingRatio` option for up-sampling or down-sampling each class separately, in order to build a model that better accounts for the smaller class's distribution. Alternatively, the `imbalance` option can be used to automatically handle skewed data.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"rdf_skewed",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[100],
      "objectiveColumn":"yearly-income",
      "imbalance":on
     }
    }' 'http://api.skytree.net:8080/v1/rdf/train'
```

RDFR Model Examples

This section includes the following advanced API usage examples for creating RDFR models.

- Tuning for the Sampling Ratio (page 119)
- Tuning the Minimum Node Size (page 119)

Tuning for the Sampling Ratio

The `samplingRatio` option can be used to change the fraction of the data that is used in each tree of a random forest.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"rdfr_sampling",
     "dataSetId":"1483908776",
     "configuration":
     {
      "numberOfTrees":[100],
      "objectiveColumn":"capital-gain",
      "samplingRatio":[[1.0],[2.3]]
     }
    }' 'http://api.skytree.net:8080/v1/rdfr/train'
```

Tuning the Minimum Node Size

Use the `minNodeWeight` option to specify the minimum bound on the total size in each leaf during tree building.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"rdf_minnodesize",
     "dataSetId":"1483908776",
```

```
    "configuration":
    {
     "numberOfTrees":[[5],[5],[25]],
     "objectiveColumn":"capital-gain",
     "minNodeWeight":[10,100]
    }
   }' 'http://api.skytree.net:8080/v1/rdfr/train'
```

SVM Model Examples

This section includes the following advanced API usage examples for creating SVM models.

- Tuning a Non-Linear SVM Model (page 120)
- Tuning for Best Accuracy (page 121)

Tuning a Non-Linear SVM Model

Use POST to train a dataset with a given configuration to produce an SVM model. The following example tunes a non-linear SVM model based on dataset ID 3355818668. This example tunes over regularization values 0.01,0.1 and RBF (radial-basis function) kernels with bandwidths 0.1, 1.0, and 10.0 (for a total of 6 parameters settings).

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"svm_train_model",
     "dataSetId":"3355818668",
     "configuration":
     {
      "numberOfFolds":10,
      "objectiveColumn":"capital-gain",
      "kernel":rbf,
      "lambdaValue": [0.01],[0.1],
      "rbfBandwidth":[[0.1],[1],[10]]
     }
    }' 'http://api.skytree.net:8080/v1/svm/train'
```

Response

After a model is trained, Skytree stores the model using a unique identifier, which is a string of a maximum of 200 alpha-numeric characters.

```
200 (OK)
"Content-type:application/json"
{
 "id":"324819856062",
 "name":"svm_train_model",
 "dataSetId":"3355818668",
 "status":
 {
  "code":"INPROGRESS",
  "message":"Model is currently being trained"
 },
```

```
 "createdAt":1401203084048,
 "updatedAt":1401203084048,
 "configuration":
 {
   "holdOutRatio":null,
   "numberOfFolds":10,
   "objectiveColumn":"capital-gain",
   "holdOutSeed":null,
   "excludeBiasTerm":null,
   "kernel":rbf,
   "epsilon":null,
   "maximumIterations":null,
   "maximumCacheMemory":null,
   "tableSamplingSeed":null,
   "classificationObjective":null,
   "kForPrecision":null,
   "testingObjective":null,
   "probabilityThreshold":null,
   "lambdaValue": [0.01],[0.1]
   "rbfBandwidth":[0.1],[1],[10],
   "polynomialDegree":null,
   "polynomialScale":null,
   "polynomialOffset":null
 },
 "projectId":"24389108236",
 "method":"SVM"
}
```

Tuning for Best Accuracy

During tuning, Skytree tunes for the best parameters and uses the model giving the best Gini by default. If you want to pick the model with the best accuracy rather than the best Gini, you can specify this with the `testingObjective` option.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"svm_train_model",
     "dataSetId":"3355818668",
     "configuration":
     {
      "numberOfFolds":10,
      "objectiveColumn":"capital-gain",
      "lambdaValue": [[0.01],[0.1],[0.5]],
      "testingObjective":ACCURACY
     }
    }' 'http://api.skytree.net:8080/v1/svm/train'
```

Chapter 9 Test Results

In the training stage of machine learning, labeled data is used to generate a model. The testing (or evaluation) stage of machine learning evaluates a model against data that was not presented during the training stage. If the new data is labeled, then Skytree can generate a score that measures the generalization power of the model.

URLs

Use the following base URLs to use for Test Results:

```
'http://api.skytree.net:8080/v1/<method>/test'
'http://api.skytree.net:8080/v1/results'
'http://api.skytree.net:8080/v1/results/<id>'
'http://api.skytree.net:8080/v1/results/<id>/dependents'
'http://api.skytree.net:8080/v1/results/<id>/interrupt'
'http://api.skytree.net:8080/v1/results/<id>/labels'
'http://api.skytree.net:8080/v1/results/<id>/probabilities'
'http://api.skytree.net:8080/v1/results/<id>/targets'
'http://api.skytree.net:8080/v1/results?projectId=<project_id>'
'http://api.skytree.net:8080/v1/results/<id>/binaryClassificationCurve'
```

Test a Trained \Model

For classification methods, predicted probabilities and labels can be compared against known results. For regression methods, the targets can be compared against known results.

> **Note:** When using a model to predict labels for new data, any new levels of a categorical variable are treated as missing values by the machine learning model.

Request

Use `POST` to test a trained model against a dataset. Note that you must know the method used to generate the model and add this to the URL. In the example below, a model with ID 3116497913 and built with GBT is scored. In this test, the Gini index is computed based on the Lorenz curve.

> **Note:** The dataset ID and model ID specified when testing a trained model must reside in the specified project. You will get an error if you attempt to reference datasets and/or models that are in a different project.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
```

```
-H "Content-type:application/json" \
-d '{
     "projectId":"24389108236",
     "name":"classifcation_score",
     "modelId":3116497913",
     "dataSetId": 2964688485,
     "configuration":
     {
       "curve":"lorenz",
       "classWeights":[1]
    }' 'http://api.skytree.net:8080/v1/gbt/test'
```

Interrupt a Test Result

You may find it necessary to interrupt a test result that is currently INPROGRESS. Note that you cannot interrupt test results that have a status of READY.

Request

Use POST to interrupt the process running on a result. The result ID is required. In the example below, a result with an ID of 8534304197005081735 is interrupted.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http:/api.skytree.net:8080/v1/results/8534304197005081735/interrupt'
```

Retrieve a Test Result

Use GET along with a valid test ID to retrieve a test result.

Request

The example below is a request to retrieve a test result whose ID is 831718717.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results/831718717'
```

Response

If the specified ID exists, then the response returns information for the test results.

Retrieve All Test Results

Use GET without a test ID to retrieve all available test results.

Request

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results'
```

Response

The response returns information for all tests generated across all projects.

Retrieve All Test Results in a Project

Use GET with a project ID to retrieve all test results that were generated in a project.

Request

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results?projectId=24389108236'
```

Response

The response returns information for all tests generated in the project.

Retrieve Test Result Dependents

Use this function to get a list of all downstream (dependent) files related to a result. This includes all plots and datasets created from this result.

Request

Use GET along with a test result ID to retrieve all files dependent on the result. The request below shows how to retrieve a list of files related to a test result whose ID is 733579628360679490'').

```
curl -X GET -H "x-auth-token:dGVzdDFAdXNlci5jb206dGVzdDE=" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results/733579628360679490")/dependents'
```

Response

Upon successful completion, the response returns a list of files that are dependent on this result.

```
200 (OK)
"Content-type:application/json"

{
 "id":"733579628360679490",
 "name":"GBT Result",
 "type":"TESTRESULT",
 "is_deletable":True
},
 "id":"7832448653536604639",
 "name":"test_result_source_733579628360679490",
 "type"="SOURCE",
 "is_deletable":True
},
{
 "id":"2224693962685268478",
 "name":"test_result_dataset_733579628360679490",
 "type":"DATASET",
```

```
 "is_deletable":True
},
...
```

Retrieve a Labels File

When generating a labels file, the labels are always "-1" and "1" for a binary classification problem. If the original dataset has a more descriptive label, such as "<=50k" and ">50k" (as is the case with the `income.data` file), then Skytree creates a mapping from the original labels to the integers. It does this via lexicographically ordering the label column. The first label gets mapped to "-1" and the second label gets mapped to "1" in the case of binary classification.

Request

Use GET with a test ID followed by `/labels` to retrieve the outputted file that includes the stored computed labels generated during a test run. Note that this API is only supported with classification methods. The example below requests the outputted labels for test ID 438626851.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results/438626851/labels'
```

Response

The response is a raw stream of the labels file.

```
1,-1
2,-1
3,-1
4,-1
5,-1
6,1
7,-1
8,1
9,1
```

Retrieve a Probabilities File

Use GET with a test ID followed by `/probabilities` to retrieve the outputted file that includes the stored computed probabilities generated during a test run. Note that this API is only supported with classification methods.

Request

The example below requests the outputted probabilities for result ID 438626851.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results/438626851/probabilities'
```

Response

The response is a raw stream of the probabilities file.

```
1,0.0000000000000000e+00
2,0.0000000000000000e+00
3,0.0000000000000000e+00
4,0.0000000000000000e+00
5,2.5000000000000000e-01
6,6.6666666666666663e-01
7,0.0000000000000000e+00
8,7.8861003861003864e-01
...
```

Retrieve a Targets File

Use GET with a test ID followed by `/targets` to retrieve the outputted file that includes the stored computed targets generated during a test run. Note that this API is only supported with regression methods.

Request

The example below requests the outputted targets for test ID 1076342703.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results/1076342703/targets'
```

Response

The response is a raw stream of the targets file.

```
1,-1
2,-1
3,-1
4,-1
5,-1
6,1
7,-1
8,1
9,1
```

Retrieve a ROC Curve

Use GET with a test ID followed by `/binaryClassificationCurve` to retrieve the test results' ROC curve points. The output includes the following information:

- Precision
- Recall
- Probability threshold
- True positives
- False positives
- True negatives
- False negatives
- Percentile

Note: This API is only supported with classification methods.

Request

The example below requests the outputted ROC for test ID 438626851.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results/438626851/binaryClassificationCurve'
```

Response

The response is a stream of the curve in JSON format.

```
[{"precision":1.0,"recall":0.0,"probabilityThreshold":1.0,"tp":0.0,"fp":0.0,"tn":47
664.0,"fn":32014.0,
"percentile":0.0},
{"precision":1.0,"recall":0.02489535828075217,"probabilityThreshold":0.997341275819
1728,"tp":797.0,
"fp":0.0,"tn":47664.0,"fn":31217.0,"percentile":98.99972388865183},
{"precision":1.0,"recall":0.04979071656150434,"probabilityThreshold":0.996845093353
2923,"tp":1594.0,
"fp":0.0,"tn":47664.0,"fn":30420.0,"percentile":97.99944777730364},
{"precision":1.0,"recall":0.07468607484225652,"probabilityThreshold":0.996014600615
7508,"tp":2391.0,
"fp":0.0,"tn":47664.0,"fn":29623.0,"percentile":96.99917166595547},
{"precision":1.0,"recall":0.09958143312300868,"probabilityThreshold":0.995118573435
4514,"tp":3188.0,
"fp":0.0,"tn":47664.0,"fn":28826.0,"percentile":95.9988955546073},
{"precision":1.0,"recall":0.124445555069657,"probabilityThreshold":0.99421719791182
62,"tp":3984.0,
"fp":0.0,"tn":47664.0,"fn":28030.0,"percentile":94.99987449484173},
{"precision":0.999790838736666,"recall":0.1493096770163054,"probabilityThreshold":0
.9931741258981935,
"tp":4780.0,"fp":1.0,"tn":47663.0,"fn":27234.0,"percentile":93.99959838349356},
...
```

Delete a Test Result

Use `DELETE` to delete a test result based on its ID.

Request

The example below is a request to delete a result with an ID of 831718717.

```
curl -X DELETE -H "x-auth-token:dXNlcjFAccGFzc3dvcmQy" \
-H "Content-type:application/json" \
 'http://api.skytree.net/v1/results/831718717'
```

Response

The API returns 204 `No Content` upon successful completion.

Chapter 10 Plots

Users can create a file to store the partial dependency variables of the relevant features and/or feature pairs in a JSON format. These pairs can then be used to generate partial dependence plots (PDPs). When used with models, these plots show the dependence between a feature or set of features marginalized over all other features. When used with results, the plots show either the relation between predictors and actual outcomes (regression) or the capture deviation information (classification).

URLs

Use the following base URLs to use for retrieving partial dependency variables and for generating plots:

```
'http://api.skytree.net:8080/v1/plots'
'http://api.skytree.net:8080/v1/plots/variableName/<model_id>/model'
'http://api.skytree.net:8080/v1/plots/<plot_id>'
'http://api.skytree.net:8080/v1/plots/<id>/interrupt'
'http://api.skytree.net:8080/v1/plots?projectId=<id>'
'http://api.skytree.net:8080/v1/plottypes/<id>'
'http://api.skytree.net:8080/v1/plottypes?viz=DATASET'
'http://api.skytree.net:8080/v1/plottypes?viz=MODEL'
'http://api.skytree.net:8080/v1/plottypes?viz=TESTRESULT'
```

Retrieve Partial Dependence Variables from a Model

Use GET along with a valid model ID to retrieve partial dependence variables from a model.

Request

The example below is a request for the partial dependence variables from a model whose ID is 1398605668.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/plots/variableName/1398605668/model'
```

Response

If the specified model ID exists, then the JSON response will include all variables as well as any variable pairs:

```
{
 "variableNames":
 [
  [
   "capital-gain"
```

```
  ],
  [
   "education-num"
  ],
  [
   "capital-loss"
  ],
  [
   "age"
  ],
  [
   "hours-per-week"
  ],
  [
   "relationship"
  ],
  [
   "occupation"
  ],
  [
   "marital-status"
  ],
  [
   "education"
  ],
  [
   "workclass"
  ],
  [
   "capital-gain",
   "education-num"
  ],
  [
   "capital-gain",
   "capital-loss"
  ],
  [
   "capital-gain",
   "age"
  ],
  [
   "capital-gain",
   "hours-per-week"
  ],
  [
   "capital-gain",
   "relationship"
  ],
 ...
]
}
```

Retrieve Plot Types for an ID

Plot values contain information about the arguments used to compute summary data for plots. Use `GET` along with a valid resource ID to retrieve all plot values for a plot type based on its ID.

Note: The Resource IDs can be retrieved using `http://api.skytree.net:8080/v1/plottypes?viz=<type>`. Refer to *Retrieve Plot Types for a Data Type* (page 131) for more information.

Request

The example below is a request to retrieve the plot values for a specific result. The resource ID is 5094586945860902, which indicates a regression result (Prediction vs. Actual).

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/plottypes/5094586945860902'
```

Response

If the specified ID exists, then the JSON response will include all plot values for the specified test result.

```
{
  "id": "5094586945860902",
  "name": "Prediction Vs Actual (Regression)",
  "type": "TESTRESULT",
  "argsSpec": "{
    "columns": ["Objective Column", "Predicted Targets"],
    "summarizer": [
      {
        "name": "returnSize",
        "kind": "fixed",
        "type" : "integer",
        "default": 10000
      }
    ],
  "plot": []
}",
  "script": "...",
  "deleted": false,
  "description": "Regression Prediction vs. Actual",
  "summarizerRegistry":
  {
    "id": "7843578453492987",
    "name": "Regression_Sampling_Summarizer",
    "script": "..."
  }
}
```

Retrieve Plot Types for a Data Type

Plot values contain information about the arguments used to compute summary data for plots. Use `GET` along with a valid type endpoint to retrieve all plot values for that data type. Accepted endpoints include:

- DATASET
- MODEL
- TESTRESULT

Request

The example below is a request to retrieve plot values for all test results.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/plottypes?viz=TESTRESULT'
```

Response

The JSON response is an array of plot arguments. These arguments are then included when generating a plot for a test result. In addition, the "id" value is used when retrieving plot arguments based on the resource ID.

```
[
  {
    "id": "1084506945850907",
    "name": "Capture Deviation Plot (Classification)",
    "type": "TESTRESULT",
    "argsSpec": "{
    "columns": ["Objective Column", "Predicted Probabilities", "Predicted Labels",
"Predicted Categories"],
    "summarizer": [
    {
      "name": "numBuckets",
      "kind": "optional",
      "type" : "integer",
      "default": 10
    }
    ],
    "plot": []
  }",
  "script": null,
  "deleted": false,
  "description": "Capture Deviation Plot for Classification",
  "summarizerRegistry": null
  },
  {
    "id": "5094586945860902",
    "name": "Prediction Vs Actual (Regression)",
    "type": "TESTRESULT",
    "argsSpec": "{
    "columns": ["Objective Column", "Predicted Targets"],
    "summarizer": [
    {
      "name": "returnSize",
      "kind": "fixed",
      "type" : "integer",
      "default": 10000
    }
    ],
    "plot": []
  }",
  "script": null,
  "deleted": false,
  "description": "Regression Prediction vs. Actual",
  "summarizerRegistry": null
  }
]
```

Create a Plot from a Model

Use POST to create a plot that shows the dependence between a feature or set of features marginalized over all other features.

> **Note:** The Plot Arguments to include are obtained using the 'http://api.skytree.net:8080/v1/plottypes?viz=MODEL' request.

Request

The example below is a request to create a plot from a model focusing on the capital-gain feature.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d {
    "inputId":"5935053016899787485",
    "inputType":"MODEL",
    "projectId":"2970189089867527669",
    "plotRegistryId":"4543534123453455",
    "columnName":"capital-gain"
  } \
'http://api.skytree.net:8080/v1/plot'
```

Upon successful completion, the response includes the generated Plot ID.

Create a Plot from a Test Result

Use POST to create a plot that shows the relation between predictors and outcomes (regression plot) or to visualize the capture deviation (classification plot).

> **Note:** The Plot Arguments to include are obtained using the 'http://api.skytree.net:8080/v1/plottypes?viz=TESTRESULT' request. Refer to *Retrieve Plot Types for a Data Type* (page 131) for more information.

Request

The example below is a request to generate a plot for a classification test result.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
-d {
    "inputId":"5935053016899787485",
    "inputType":"TESTRESULT",
    "projectId":"2970189089867527669",
    "plotRegistryId":"1084506945850907",
    "plotArgs":"{ \"columns\":[\"Objective Column\",
    \"Predicted Probabilities\",\"Predicted Labels\",
    \"Predicted Categories\"],\"summarizer\":[
    {\"name\":\"numBuckets\",\"kind\":\"optional\",
    \"type\":\"integer\",\"default\":10,\"value\":10}
    ],\"plot\":[]}",
```

```
    "plotName":"Capture Deviation Plot"
  } \
'http://api.skytree.net:8080/v1/plot'
```

Upon successful completion, the response includes the generated Plot ID.

Interrupt a Plot

You may find it necessary to interrupt a plot that is currently INPROGRESS. Note that you cannot interrupt plots that have a status of READY.

Request

Use POST to interrupt the process running on a plot. The plot ID is required. In the example below, a plot with an ID of 1355840547833663337 is interrupted.

```
curl -X POST -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http:/api.skytree.net:8080/v1/plots/1355840547833663337/interrupt'
```

Retrieve a Plot

Use GET to retrieve a single plot based on its Plot ID.

Request

The example below is a request to retrieve a plot whose ID is 4271020660339219591.

```
curl -H "x-auth-token:dXNlcjFAt5dHJZS5uZXQ6cGFz" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/plots/4271020660339219591'
```

Response

If the specified ID exists, then the JSON response will include information for the specified plot, including the type.

```
{
  "id": "4271020660339219591",
  "name": "Capture Deviation Plot",
  "plotRegistryId": "1084506945850907",
  "args": "{ "columns": ["Objective Column", "Predicted Probabilities", "Predicted
Labels",
"Predicted Categories"], "summarizer": [ { "name": "numBuckets", "kind": "fixed",
"type" : "integer", "default": 10 } ], "plot": [] }",
  "inputId": "2018345810970498281",
  "inputType": "TESTRESULT",
  "createdAt": 1450454920287,
  "modifiedAt": 1450454920287,
  "deleted": false,
  "status":
  {
    "code": "READY",
    "message": "Plot is Ready."
  }
```

```
}
```

Retrieve All Plots in a Project

Use GET along with a project ID to retrieve all available plots in a project.

Request

The following example shows a request to retrieve all plots in a project whose ID is 7842277202471690784. Note that GET is the default behavior and, thus, is not required.

```
curl -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/plots?projectId=7842277202471690784'
```

Response

The response returns an array of all plots in the specified project.

Delete a Plot

Use DELETE to delete a plot based on its ID.

Request

The example below is a request to delete a plot with an ID of 8434238370819421548.

```
curl -X DELETE -H "x-auth-token:dXNlcjFAccGFzc3dvcmQy" \
-H "Content-type:application/json" \
 'http://api.skytree.net/v1/plots/8434238370819421548'
```

Response

The API returns 204 No Content upon successful completion.

Chapter 11 Log Files

The REST API provides a series of URLs that allow you to view logs generated during machine learning.

URLs

Use the following base URLs to manage ensemble modeling resources:

```
'http://api.skytree.net:8080/v1/models/<model_id>/logs'
'http://api.skytree.net:8080/v1/results/<results_id>/logs'
```

Retrieve Log File for Models

Use GET with a model ID to retrieve the log file generated during model training.

Request

In the request below, a log file for model 2119345000 is retrieved.

```
curl -H GET -H "x-auth-token:dXNlcjFAc6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/models/2119345000/logs'
```

Response

The response returns a raw stream of the log generated during training of the model.

Retrieve Log File for Test Results

Use GET with a results ID to retrieve the log file generated during a test request.

Request

In the request below, a log file for test results 1076342703 is retrieved.

```
curl -H GET -H "x-auth-token:dXNlcjFAc6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/results/1076342703/logs'
```

Response

The response returns a raw stream of the log generated during the test.

Chapter 12 Troubleshooting

This section provides tips for some common scenarios that you might encounter while using Skytree.

What version of Skytree am I running?

The REST API includes a URL for retrieving the current version. This information can be helpful, for example, when communicating with customer support.

Request

Use GET to retrieve the version information.

```
curl -H GET -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/version'
```

Response

The response returns the version information from the current manifest. This includes the implementation version as well as the build number.

Restarting the API

Restarting the API can be done manually or by setting up a cron job.

```
# sudo service skytree-platform restart
```

My jobs are stuck "INPROGRESS" state

1. Review the list of applications currently running on YARN.

```
# $HADOOP_HOME/bin/yarn application -list
```

2. This will list all applications running on YARN. Verify that you can see the following application:

```
  "Skytree_Yarn_Application_*" are being run by skytree platform.
```

3. This application can be killed by using command:

```
# $HADOOP_HOME/bin/yarn application -kill $applicationId
```

4. If jobs still don't complete, we recommend that you restart the Skytree platform.

My job failed

The most common cause of a job failure is do to lack of resources/memory. In this case, you might see an error such as the following:

```
ValueError: Resource 3750527812 has Error:
Process failed with error: Job has failed.
Please check logs at
/user/skytree/skytree/data_json/logs/-5301590031728_3750527812//
skytree-server.log for more details.
```

The steps for diagnosing this can vary based depending on whether you know the size of your data.

Verify environment settings

If you know the size of your data, then you can use the following API to determine if your environment is set up with enough memory to support your data.

```
curl -X GET -H "x-auth-token:dXNlcjFAc2t5dHJlZS5uZXQ6cGFzc3dvcmQy" \
-H "Content-type:application/json" \
'http://api.skytree.net:8080/v1/environments/default'
```

The response returns information about the default environment, including the number of containers in the environment along with the number of virtual cores and the amount of memory allocated to each container.

> **Note:** If you used an environment other than the default, replace the `default` keyword in the URL with the appropriate environment ID. For example:
> `'http://api.skytree.net:8080/v1/environments/1314815213'`

Admin assistance

A job that is running in a cluster might fail on a single node or on multiple nodes. In either case, the Admin (or a user who has READ access to each node) can diagnose the failure.

1. Provide the job ID to the Admin. In the above example error message, the job (Resource) ID is 3750527812.

2. Provide the environment that you used to run the job.
3. The Admin can review the YARN logs in each node a warning that is similar to the following:

```
2014-12-22 16:31:04,160 WARN
org.apache.hadoop.yarn.server.nodemanager.containermanager.monitor.ContainersMonitorImpl:
Container [pid=25605,containerID=container_1418692474261_0035_01_000003] is running beyond
physical memory limits.
Current usage: 1.4 GB of 1 GB physical memory used; 6.9 GB of 2.1 GB virtual memory used. Killing
container.
```

4. Once this warning is discovered, the Admin can reconfigure the environment to allow for more space.

I lost my password

Lost passwords cannot be retrieved. If you lose your password, contact your Admin to reset your password.

SKYTREE®

Index

N

P

R

S

T

U

V