sparkcognition™

Darwin ™ API User Guide

A SparkCognition ™ Education Document

v. 1.36.0 - 12.2019

SparkCognition Darwin API User Guide

Contents

About this guide

This manual describes the Darwin™ API and its use in automated model building. It is intended for data scientists, software engineers, and analysts who want to use the Darwin API to interact with Darwin to create and train models, monitor jobs, and perform analysis.

The Darwin API has an independent version number to allow for release outside of the normal Darwin product release window. As of this printing, the Darwin API is at version 1.36.0.

Darwin overview

Darwin is a SparkCognition™ tool that automates model building processes to solve specific problems. This tool enhances data scientist potential because it automates various tasks that are often manually performed. These tasks include data cleaning, latent relationship extraction, and optimal model determination. Darwin promotes rapid and accurate feature generation through both automated windowing and risk generation. Darwin quickly creates highly-accurate, dynamic models using both supervised and unsupervised learning methods.
For additional information on Darwin, contact your local SparkCognition partner for access to the white paper titled: *Darwin - A Neurogenesis Platform*.

The documentation for this version of Darwin includes:

- The *Darwin Release Notes*, version 2.0.4
- The *Darwin User Interface Guide*, version 2.0.4
- The *Darwin API User Guide*, version 1.36.0
- The *Darwin Python SDK User Guide*, version 1.45.0
- The *Darwin RTE User Guide*, version 2.0.4

All of these documents are available for download from the Darwin support portal.

Accessing the API

The Darwin API can normally be accessed through one of three methods:

- the Darwin Python SDK (preferred, recommended)
- the `https://darwin-api.sparkcognition.com/v1` end point
- optionally, through user-created `curl` commands

For additional information on the Darwin SDK, see the *SparkCognition Darwin Python SDK Guide*.

Expectation

This document assumes the experience of a data scientist or software engineer that is knowledgeable of data science techniques and associated programming tasks.

Technical routes

The Darwin API includes the following api operations:

- `analyze` - analyze a model or dataset
- `auth` - register and authenticate
- `clean` - preprocess a dataset
- `download` - download or delete a generated artifact
- `info` - get API system information including available routes and version number
- `job` - return status on jobs

- `lookup` - get model or dataset metadata
- `run` - run a model on a dataset
- `train` - train a model
- `upload` - upload or delete a dataset

analyze

Request Type: POST

URI: /v1/analyze/model/*{model_name}*

Headers:

- Authorization: Bearer token

Form Data:

- *model_name*: (required) The name of the model to be analyzed
- *job_name*: (optional) If not specified, a uuid is created as the *job_name*.
- *artifact_name*: (optional) If not specified, a uuid is created as the *artifact_name*.
- *category_name*: (optional) The name of the class for supervised or cluster for unsupervised to get feature importances for. If this is not specified, the feature importances will be over all classes/clusters.
- *model_type*: (optional) Model type from the population. Possible values include: *DeepNeuralNetwork*, *RandomForest*, *GradientBoosted*.

Description: Analyze the universal feature importances for a particular model given the model name. **Note**: This API is capable of returning the structure of the model in the form of a pandas Series.

Note: This method is supported for clustering and NBM models. It does not support forecasting or unsupervised anomaly detection.

Response Codes: 201, 400, 401, 403, 422

Successful Response:

```
{
  "job_name": "string",
  "artifact_name": "string"
}
```

Request Type: POST

URI: /v1/analyze/model/predictions/*{model_name}*/*{dataset_name}*

Headers:

- Authorization: Bearer token

Form Data:

- *dataset_name*: (required) The name of the dataset containing the data to analyze predictions for. This is a new dataset that was not used during training for which you want feature importance scores for each row of this dataset. This dataset has a limit of 500 rows. There is no limit for columns.
- *model_name*: (required) The name of the model to be analyzed
- *job_name*: (optional) If not specified, a uuid is created as the *job_name*.
- *artifact_name*: (optional) If not specified, a uuid is created as the *artifact_name*.
- *start_index*: (optional) Index to start at in the dataset when analyzing model predictions. All numeric and datetime data types can be indexes. When specifying an index as a datetime, the preferred timestamp format is `2019-02-15 19:46:48`.
- *end_index*: (optional) Index to stop at in the dataset when analyzing model predictions. All numeric and datetime data types can be indexes. When specifying an index as a datetime, the preferred timestamp format is `2019-02-15 19:46:48`.
- *model_type*: (optional) Model type from the population. Possible values include: *DeepNeuralNetwork, RandomForest, GradientBoosted*.

Description: Analyze specific feature importances for a particular sample or samples given the model name and sample data. Analyze predictions cannot be used if you trained your model with a dataset that is larger than 500 MB.

Note: This route is not supported in forecasting models or for clustering/anomaly detection, however it does support NBM modeling.

Response Codes: 201, 400, 401, 403, 422

Successful Response:

```
{
  "job_name": "string",
  "artifact_name": "string"
}
```

Request Type: POST

URI: /v1/analyze/data/*{dataset_name}*

Headers:

- Authorization: Bearer token

Description: Analyze a dataset and return statistics/metadata concerning designated data.

Note: You can only analyze a dataset once. If you try to analyze the dataset a second time, you will get a `400: BAD REQUEST` error.

Parameter Descriptions:

- *dataset_name*: (required) The name of the dataset to analyze and return statistics/metadata for
- *job_name*: The job name

- *artifact_name*: The artifact name
- *max_unique_values*: Threshold for automatic pruning of categorical columns prior to one hot encoding based on the number of unique values
  Note: If a categorical column contains at least *max_unique_values*, it is dropped during preprocessing prior to one hot encoding.

Payload:

```
{
  "job_name": "string",
  "artifact_name": "string",
  "max_unique_values": 30
}
```

Response Codes: 201, 400, 401, 403, 408, 422

Successful Response:

```
{
  "job_name": "string",
  "artifact_name": "string"
}
```

---

auth

Request Type: PATCH

URI: /v1/auth/email

Headers:

- Authorization: Bearer token

Description: Add or change an email address.

Form Data:

- *email*: Email address

Response Codes: 204, 400, 401, 422

Successful Response:

```
{
    'access_token': 'token_string'
}
```

Request Type: POST

URI: /v1/auth/login

Headers:

- Authorization: Bearer token

Description: Login as a service.

Form Data:

- *api_key*: The api key of the service
- *pass1*: The service level password

Response Codes: 201, 400, 401

Successful Response:

```
{
    'access_token': 'token_string'
}
```

Request Type: POST

URI: /v1/auth/login/user

Description: Login as a user.

Form Data:

- *username*: The end user's name
- *pass1*: The end user's password

Response Codes: 201, 400, 401, 422

Successful Response:

```
{
    'access_token': 'token_string'
}
```

Request Type: PATCH

URI: /v1/auth/password

Headers:

- Authorization: Bearer token

Description: Change the password.

Form Data:

- *curpass*: Current password
- *newpass1*: New password
- *newpass2*: Confirmation of new password

Response Codes: 204, 400, 401, 422

Successful Response:

```
{
    'access_token': 'token_string'
}
```

Request Type: PATCH

URI: /v1/auth/password/reset

Headers:

Description: Reset a user's password. Any user can reset another user's password. You do not have to be an admin to execute this function. For cloud installation, a temporary password will be sent to the user's email address. For on-prem installations, password resets are executed using a new default password, there is no email sent. If you do not know the default password, contact Darwin support.

Form Data:

- *username*: The username of the user whose password needs resetting

Response Codes: 201, 400, 401, 422

Successful Response:

```
{
    'access_token': 'token_string'
}
```

Request Type: POST

URI: /v1/auth/register

Headers:

Description: Register as a service.

Form Data:

- *api_key*: The api key of the service
- *pass1*: The service level password
- *pass2*: The service level password confirmation
- *email*: Email address

Response Codes: 201, 400, 401, 403

Successful Response:

```
{
    'access_token': 'token_string'
}
```

Request Type: POST

URI: /v1/auth/register/user

Headers:

- Authorization: Bearer token

Description: Register a user for your service.

Form Data:

- *username*: The end user's name
- *pass1*: The end user's password
- *pass2*: The end user's password confirmation
- *email*: The end user's email address

Response Codes: 201, 400, 401, 422

Successful Response:

```
{
    'access_token': 'token_string'
}
```

Request Type: DELETE

URI: /v1/auth/register/user/{username}

Headers:

- Authorization: Bearer token

Description: Remove/Unregister a user.

Form Data:

- *username*: The username of the user to remove

Response Codes: 201, 401, 403, 422

Successful Response: None

---

clean

Request Type: POST

URI: /v1/clean/dataset/*{dataset_name}*

Headers:

- Authorization: Bearer token

Description: Clean a named dataset. The output is the cleaned dataset which is scaled and one-hot-encoded based on parameters in */analyze/data*. Use */download/dataset* to retrieve the cleaned dataset. *clean_data()* needs to be performed prior to creating a model and again before running a model. When you run clean_data() before creating a model, you must specify a dataset_name and a target. When you run clean_data() before running a model, you must specify a dataset_name and a model_name. clean_data() can also be used for visualizing what Darwin would do with the dataset or for when you want to use the cleaned data outside of Darwin.

Form Data:

- *dataset_name*: Name of dataset to clean
- *job_name*: Name of job
- *artifact_name*: Name given to the cleaned dataset
- *model_name*: (Mandatory for running a model) Specify the model name when you clean data before running a model.
- *target*: (Mandatory for Supervised Model Building) String denoting target prediction column in input data.
- *index*: String denoting the date/time column name to use as an index.
- *impute*: String alias that indicates how to fill in missing values in input data.

ALIAS	DESCRIPTION	COMPLEXITY
`ffill`	**(Default)** Forward Fill: Propagate values forward from one example into the missing cell of the next example. Might be useful for timeseries data, but also applicable for both numerical and categorical data.	Linear Fast
`bfill`	Backward Fill: Propagate values backward from one example into the missing cell of the previous example. Might be useful for timeseries data, but also applicable for both numerical and categorical data.	Linear Fast
`mean`	Mean Fill: Computes the mean value of all non-missing examples in a column to fill in missing examples. The result may or might not be interpretable in terms of the input space for categorical variables.	Linear Fast
`median`	Median Fill: Computes the median value of all non-missing examples in a column to fill in missing examples. While the result is interpretable in terms of the input space for categorical variables, the approach might not be appropriate for non-ordinal data.	Linear Fast
`linear`	Linear Interpolation Fill: Interpolation using a Linear function. Might be useful for timeseries or sequential data.	Linear Fast

- *max_int_uniques*: Expected input/type: *integer*. Threshold for automatic encoding of categorical variables. If a column contains less than *max_int_uniques* unique values, it is treated as categorical and one hot encoded during preprocessing. **Note:** If the target has more numeric values than the *max_int_uniques* set point, the problem is treated as a regression and will use MSE.

- *max_unique_values*: Expected input/type: *integer*. Threshold for automatic pruning of categorical columns prior to one hot encoding based on the number of unique values.
  Note: If a categorical column contains at least *max_unique_values*, it is dropped during preprocessing prior to one hot encoding.

Response Codes: 400, 401, 403, 422

Successful Response:

```
{
    "job_name": "string",
    "job_id": "string",
    "profile_name": "string",
    "profile_id": "string"
}
```

---

download

Request Type: GET

URI: /v1/download/artifacts/*{artifact_name}*

Headers:

- Authorization: Bearer token

Description: Download an artifact by name.

Form Data:

- *artifact_name*: Name of the artifact to download

Response Codes: 201, 401, 404, 408, 422

Successful Response:

```
{
    'artifact': 'artifact_name'
}
```

Request Type: DELETE

URI: /v1/download/artifacts/*{artifact_name}*

Headers:

- Authorization: Bearer token

Description: Delete an artifact.

Form Data:

- *artifact_name*: Name of the artifact to download

Response Codes: 204, 401, 404, 408, 422

Successful Response: None

Request Type: GET

URI: /v1/download/dataset/*{dataset_name}*

Headers:

- Authorization: Bearer token

Description: Download a dataset by name. It can be an original or cleaned dataset.

Form Data:

- *dataset_name*: Name of the dataset to download. In the case of downloading a cleaned dataset, this would be the name returned by */clean/dataset/{dataset_name}*.
- *file_part*: Part number of a multi-part dataset, expressed as an integer.

Response Codes: 401, 404, 408, 422

Successful Response:

```
{
    "dataset": "string",
    "part": 1,
    "note": "string"
}
```

Request Type: GET

URI: /v1/download/model/*{model_name}*

Headers:

- Authorization: Bearer token

Description: Download a supervised model by name.

Form Data:

- *model_name*: Name of the model to download
- *path*: (optional) Relative or absolute path of the directory to download the model to. This directory must already exist prior to model download. If no path is specified, the current directory is used. There are two files associated with a model: *'model'* and *'data_profiler'*.
- *model_type*: (optional) Model type of the model to be downloaded. Possible values include the following: *DeepNeuralNetwork, RandomForest, GradientBoosted.*
- *model_format*: (optional) Format in which the model is to be downloaded. Possible values include: *json, onnx.* Note: The ONNX format is only available for neural network models.

Response Codes: 401, 404, 408, 422

Successful Response:

A successful response returns a .zip file, which contains two files: the supervised model itself and the data profiler. Downloading unsupervised models is not supported.

---

info

Request Type: GET

URI: /v1/info

Query Parameters: None

Description: Get info on the routes available and the API version. The `local` flag will return `True` for an on-prem installation.

Response Codes: 200

Successful Response:

```
{
  'available_routes': {
    'Info': true,
    'Auth': true,
    'Job': true,
    'Metadata': true,
    'Train': true,
    'Risk': true,
    'Upload': true,
    'Download': true,
    'Analyze': true,
    'Run': true,
    'Admin': true,
    'Clean': true,
    'Model': true
},
    'local': false,
    'api_version': '1.34.0'
}
```

---

job

Request Type: GET

URI: /v1/job/status

Headers:

- Authorization: Bearer token

Query Parameters:

- *age*: List jobs that are less than X units old (for example, 3 weeks, 2 days)
- *status*: List job of a particular status, for example *Running*

Description: Get the status for all jobs. Note that only 2 jobs can be running concurrently.

Response Codes: 200, 400, 401, 422

Successful Response:

```
[
    {
        "job_name": "job1_name",
        "status": "Requested",
            "starttime": "2018-01-30T13:27:46.449865",
       "endtime": "2018-01-30T13:28:46.449865",
        "percent_complete": 0,
       "job_type": "TrainModel",
        "loss": 0,
        "generations": 0,
        "dataset_names": [
            "phone_data"
        ],
        "artifact_names": [
            "art1"
        ]
        "model_name": null,
        "job_error": "string"
    },
    {
        "job_name": "job2_name",
        "status": "Running",
        "starttime": "2018-01-30T13:27:46.449865",
        "endtime": "2018-01-30T13:28:46.449865",
        "percent_complete": 23,
        "job_type": "UpdateModel",
        "loss": 0.92,
        "generations": 50,
        "dataset_names": [
            "language_data"
        ],
        "artifact_names": null,
        "model_name": "test_model",
        "job_error": "string"
    }
]
```

Request Type: GET

URI: /v1/job/status/*{job_name}*

Headers:

- Authorization: Bearer token

Description: Get the status for a particular job.

Form Data:

- *job_name*: The job name you want status on.

Response Codes: 200, 400, 401, 403, 404, 422

Successful Response:

```
{
    "status": "Requested, Running, Completed",
    "starttime": "string",
    "endtime": "string",
    "percent_complete": 30,
    "job_type": "string",
    "loss": 0,
    "generations": 0,
    "dataset_names": [
        "string"
    ],
    "artifact_names": [
        "string"
    ],
    "model_name": "string",
    "job_error": "string"
}
```

Request Type: PATCH

URI: /v1/job/status/*{job_name}*

Headers:

- Authorization: Bearer token

Description: Stop a running job. The job will not stop right away, but it will stop when the current generation is complete.

Form Data:

- *job_name*: The job name you want to stop.

Response Codes: 200, 400, 401, 403, 404, 422

Successful Response:

```
"Job is scheduled to stop"
```

Request Type: DELETE

URI: /v1/job/status/*{job_name}*

Headers:

- Authorization: Bearer token

Description: Soft delete a running job.

Form Data:

- *job_name*: The job name you want to delete.

Response Codes: 200, 400, 401, 403, 404, 422

Successful Response:

None

---

lookup

Request Type: GET

URI: /v1/lookup/limits

Headers:

- Authorization: Bearer token

Description: Get a client's usage limit metadata.

Response Codes: 200, 401, 422

Successful Response:

```
{
        "username": "string",
        "tier": 0,
        "model_limit": 0,
        "job_limit": 0,
        "upload_limit": 0,
        "user_limit": 0
}
```

Request Type: GET

URI: /v1/lookup/artifact

Headers:

- Authorization: Bearer token

Query Parameters:

- *type*: filter on the type of artifact (for example, Model, Dataset, Test, or Run)

Description: Get artifact metadata

Response Codes: 200, 401, 422

Successful Response:

```
[
  {
    "id": "string",
    "name": "string",
    "type": "string",
    "created_at": "2018-01-22T19:00:39.863Z",
    "mbytes": 0
  }
]
```

Request Type: GET

URI: /v1/lookup/artifact/*{artifact_name}*

Headers:

- Authorization: Bearer token

Description: Get artifact metadata for a single artifact

Form Data:

- *artifact_name*: The artifact name you want to look up.

Response Codes: 200, 401, 404, 422

Successful Response:

```
{
  "name": "string",
  "type": "string",
  "created_at": "2018-01-22T19:00:39.869Z",
  "mbytes": 0
}
```

Request Type: GET

URI: /v1/lookup/model

Headers:

- Authorization: Bearer token

Description: Get the model metadata for a user. This is useful if a user has forgotten certain model names.

Response Codes: 200, 401, 422

Successful Response:

```
[
    {
        "id": {},
        "name": "model1_name",
        "type": "Supervised",
        "updated_at": "2017-02-03T073000",
        "problem_type": "string"
        "trained_on": ["dataset1_id", "dataset2_id"],
        "generations": 100,
        "loss": 0.8,
        "complete": {},
        "parameters": {},
        "train_time_seconds": 240,
        "algorithm": "string",
        "running_job_id": "string",
        "description": {"best_genome": "RandomForestClassifier", "recurrent": False}
    },
    {
        "id": {},
        "name": "model2_name",
        "type": "Ensembled",
        "updated_at": "2017-08-22T175022",
        "trained_on": ["dataset3_id"],
        "loss": 0.82,
        "complete": {},
        "generations": 80,
        "parameters": {
            "target": "target1"
        },
        "train_time_seconds": 180,
        "algorithm": "string",
        "running_job_id": "string",
        "description": {"best_genome": "DeepNet(\n (l0): LSTM(20, 18, num_layers=2)\n
         (l1): Linear(in_features=18, out_features=1, bias=True)\n)",
         "recurrent": True}
    }
]
```

Note: *running_job_id* is only returned when *complete* is False.

Request Type: GET

URI: /v1/lookup/model/*{model_name}*

Headers:

- Authorization: Bearer token

Description: Get all of the model metadata for a particular model.

Form Data:

- *model_name*: The model name you want to look up.

Response Codes: 200, 401, 404, 422

Successful Response:

```
{
    "type": "Unsupervised",
    "updated_at": "2017-02-03T073000",
    "trained_on": ["dataset1_id", "dataset2_id"],
    "generations": 100,
    "loss": 0.8,
    "parameters": {},
    "train_time_seconds": 180,
    "algorithm": "string",
    "running_job_id": "string",
    "description": {"best_genome": "RandomForestClassifier", "recurrent": False}
}
```

Note: *running_job_id* is only returned when *complete* is False.

Request Type: GET

URI: /v1/lookup/model/*{model_name}*/population

Headers:

- Authorization: Bearer token

Description: Get model descriptions of the best genomes for all model types that were trained. The population is displayed for unsupervised models only.

Form Data:

- *model_name*: The model name or identifier.

Response Codes: 201, 401, 404, 422

Successful Response:

```
{
    "population": {
        "model_types": {
            "DeepNeuralNetwork": {
                "model_description": "string",
                "loss_function": "string",
                "fitness": Double
            },
            "RandomForest": {
                "model_description": "string",
                "loss_function": "string",
                "fitness": Double
            },
            "GradientBoosted": {
                "model_description": "string",
                "loss_function": "string",
                "fitness": Double
            }
        }
    }
}
```

Request Type: GET

URI: /v1/lookup/dataset

Headers:

- Authorization: Bearer token

Description: Get the dataset metadata for a user. This is useful if a user has forgotten certain dataset names.

Response Codes: 200, 401, 422

Successful Response:

```
[
    {
        "name": "dataset1_name",
        "mbytes": 0.2,
        "minimum_recommended_train_time": "string",
        "updated_at": "20170924T000000",
        "categorical": False,
        "sequential": True,
        "imbalanced": True,
    },
    {
        "name": "dataset2_name",
        "mbytes": 3.5,
        "minimum_recommended_train_time": "string",
        "updated_at": "20170902T010101",
        "categorical": True,
        "sequential": False,
        "imbalanced": False,
    }
]
```

Request Type: GET

URI: /v1/lookup/dataset/*{dataset_name}*

Headers:

- Authorization: Bearer token

Description: Get all of the metadata for a particular dataset.

Form Data:

- *dataset_name*: The dataset name for which you want the metadata.

Response Codes: 200, 401, 404, 422

Successful Response:

```
{
    "mbytes": 0.2,
    "minimum_recommended_train_time": "string",
    "updated_at": "20170924T000000",
    "categorical": False,
    "sequential": True,
    "imbalanced": True,
}
```

Request Type: GET

URI: /v1/lookup/tier

Headers:

- Authorization: Bearer token

Description: Get all of the tier metadata.

Response Codes: 200, 401, 422

Successful Response:

```
[
  {
    "tier": 0,
    "model_limit": 0,
    "job_limit": 0,
    "upload_limit": 0,
    "user_limit": 0
  }
]
```

Request Type: GET

URI: /v1/lookup/tier/*{tier_num}*

Headers:

- Authorization: Bearer token

Description: Get the metadata for a particular tier.

Form Data:

- *tier_num*: Tier for which you want metadata.

Response Codes: 200, 401, 404, 422

Successful Response:

```
{
  "tier": 0,
  "model_limit": 0,
  "job_limit": 0,
  "upload_limit": 0,
  "user_limit": 0
}
```

Request Type: GET

URI: /v1/lookup/user

Headers:

- Authorization: Bearer token

Description: Get user metadata for all users.

Response Codes: 200, 401, 422

Successful Response:

```
[
 {
  "user_id": "string",
  "internal_name": "string",
  "username": "string",
  "tier": 0,
  "created_at": "string",
  "client_api_key": "string",
  "expires_on": "string",
  "parent_id": "string"
 }
]
```

Request Type: GET

URI: /v1/lookup/user/*{username}*

Headers:

- Authorization: Bearer token

Description: Get user metadata for a particular user.

Form Data:

- *username*: Username for which you want user metadata.

Response Codes: 200, 401, 404, 422

Successful Response:

```
{
  "user_id": "string",
  "internal_name": "string",
  "username": "string",
  "tier": 0,
  "created_at": "string",
  "client_api_key": "string",
  "expires_on": "string",
  "parent_id": "string"
}
```

---

run

Request Type: POST

URI: /v1/run/model/*{model_name}*/*{dataset_name}*

Headers:

- Authorization: Bearer token

Form Data:

- *model_name*: The name of the model.
- *artifact_name*: The name of the artifact.
- *dataset_name*: The name of the dataset.

- *anomaly*: Setting this parameter to **True** indicates that an isolation forest should be built for anomaly detection. If set to **True**, clustering will automatically be interpreted as **False**.
- *supervised*: (**Deprecated**. This argument exists only for backward compatibility.) A boolean (True/False) indicating whether the model is supervised or not, for example, set this to *False* for *unsupervised*.
- *model_type* - (optional) User can specify a model type to use for their prediction. If nothing is defined, the SDK will use the best model type. Possible values include:
  - `DeepNeuralNetwork`: The run_model command will pick the best performing neural network to use when running the prediction.
  - `RandomForest`: The run_model command will pick the best performing sklearn random forest to use when running the prediction.
  - `GradientBoosted`: The run_model command will pick the best performing sklearn gradient boosted model to use when running the prediction.

Description: Run a model on a dataset and return the predictions/classifications/clusters found by the model.

Response Codes: 201, 400, 401, 403, 404, 408, 422

Successful Response:

```
{
    "job_name": "name_of_job",
    "artifact_name": "name_of_artifact"
}
```

---

train

Request Type: POST

URI: /v1/train/model

Headers:

- Authorization: Bearer token

Description: Create a model trained on the dataset identified by `dataset_names`.

Parameter descriptions:

- *dataset_names*: (required) A list of dataset names to use for training. The maximum file size is 500 MB for unsupervised and NBM and 10 GB for supervised.
  Note: Using only 1 dataset is currently supported.
- *fit_profile_name*: (required) This is the *profile_name* that is generated from the */clean/dataset/{dataset_name}* route.
- *val_size*: Portion of the dataset to be used as a validation set during training, expressed as a decimal that is greater than 0 and less than 1. Default value is 0.2 (i.e., 20%).

- *cv_kfold*: k-fold cross-validation, where k is the number of groups that a given data sample is to be split into for training/validation. Default is 1 for non-timeseries data or 3 for timeseries data. Maximum value allowed is 10.
- *job_name*: The job name.
- *model_name*: The string identifier of the model to be trained.
- *loss_fn_name*: Specify the loss function. Possible values include: *"CrossEntropy"*, *"MSE"*, *"BCE"*, *"L1"*, *"NLL"*, *"BCEWithLogits"*, *"SmoothL1"*. *"CrossEntropy"*, *"BCE"*, and *"BCEWithLogits"* can be used for classification data, while all others can be used for regression data. The default value is *"CrossEntropy"* if this field is left empty.
- *fitness_fn_name*: Specify the fitness function. This represents the name of the fitness function used for evolution of the model population during training.

  For classification problems, possible values include:

  - `average_precision` - (Average Precision) Measures the average precision across the spectrum of all recall values from 0 to 1. Average precision is a good metric to use for imbalanced problems, and only works on binary target columns, that is, there are two class labels being predicted.
  - `roc_auc` - (ROC Area Under Curve) Measures the area under the Receiver Operating Characteristics curve, which plots the relationship between precision and recall for a model. ROC area under curve only works on binary target columns, that is, there are two class labels being predicted.
  - `accuracy` - (Accuracy) Measures the total number of correct predictions divided by the total number of predictions made.
  - `f1_weighted` - (F1 Weighted) (default) Measures the F1 score for each label and finds their average, which is weighted by the number of true instances for each label. This alters 'macro' to account for label imbalance.
  - `f1_macro` - (F1 Macro) Measures the F1 score, but calculates metrics for each label, and finds their unweighted mean. This is recommended for imbalanced problems.
  - `f1_micro` - (F1 Micro) Measures the F1 metrics globally by counting the total true positives, false negatives, and false positives.
  - `balanced_accuracy` - (Balanced Accuracy) Measures the proportion correct of each class individually and then averages those values. This is a good metric to use for imbalanced problems.
  - `neg_log_loss` - (Log Loss) Measures the prediction probability of each output and how closely that maps to the actual label. In binary classification, if the actual label was 0 and the prediction probability was 0.01, the prediction would be 0.49 better than a prediction probility of 0.5. This is a very harsh penalty mechanism and will result in a model that tries to find a very defined boundary between classes.
  - `precision_macro` - (Precision Macro) Measures precision for each label and finds their unweighted mean. This is recommended for imbalanced problems.
  - `precision_micro` - (Precision Micro) Measures the precision metrics globally by counting the total true positives predicted.
  - `precision_weighted` - (Precision Weighted) Measures the precision score for each label, and then finds their average weighted by the number of true instances for each label. This alters

'macro' to account for label imbalance.
  - `recall_macro` - (Recall Macro) Measures recall for each label and finds their unweighted mean. This is recommended for imbalanced problems.
  - `recall_micro` - (Recall Micro) Measures the recall metrics globally by counting the total true positives predicted.
  - `recall_weighted` - (Recall Weighted) Measures the recall score for each label and finds their average weighted by the number of true instances for each label. This alters 'macro' to account for label imbalance.

  For regression problems, possible values include:

  - `r2` - (R^2) (default) Measures how closely the data maps to the fitted regression line. It is also known as the coefficient of determination and is useful for mapping the relationships that exist in data.
  - `neg_mean_absolute_error` - (Mean Absolute Error) Measures the average error for each predicted data point versus the expected value. This is useful as a good baseline metric or for capturing general trends.
  - `neg_mse` - (Mean Squared Error) Measures the square of the average error for each predicted data point versus the expected value. This is useful if you want to penalize large errors more harshly.
  - `neg_median_absolute_error` - (Median Absolute Error) Measures the median error for the predicted data point versus the expected value. This is useful if your dataset has biases toward certain values.
  - `neg_rmse` - (Root Mean Squared Error) Measures the square root Mean Squared Error values. This is useful if there are not a lot of outliers in your data.
  - `neg_rmsle` - (Root Mean Squared Logarithmic Error) Measures the ratio between the actual and predicted values by calculating the square root of the Mean Squared Error values in which a logarithmic transform is performed on predicted and actual values. This is useful for targets with very large numbers or that contain outliers. An error will be generated if a negative target value is encountered. This fitness function should only be used for positive datasets.

- *max_train_time* (supervised only): Sets the training time for the model in 'HH:MM' format. Default value is `00:01`.

- *max_epochs*: Expected input/type: *numeric*. Sets the training time for the model in epochs. Default value is `10`.

- *recurrent*: Expected input/type: *True/False*. Enables recurrent connections to be evolved in the model. This option can be useful for timeseries or sequential data, but may result in slower model evolution. If you want to see the LSTM and TCN models used during training, you must set `recurrent = True`.

- *anomaly*: Setting this parameter to **True** indicates that an isolation forest should be built for anomaly detection. If set to **True**, clustering will automatically be interpreted as **False**.

- *n_clusters* (*unsupervised* only): Specifies the number of clusters to be used.
  Note: If this value is not provided, the number of clusters will be heuristically determined.

- *forecast_horizon*: Integer indicating how long in the future you want to forecast predictions. For example, if you have 6 months of time-series data and each row represents a 1 day interval and you want to predict the next week of data, you should set `forecast_horizon=7`. If each row is a 1 hour interval, then you should set `forecast_horizon=168`. (168 = 7*24)

  Note: For best results, be sure that the amount of data to train on is at least 4 times the amount of the *forecast_horizon*.

- *anomaly_prior* (*unsupervised* only): Expected input/type: *between [0,1]*. Significance level at which a point is defined as anomalous. This is only used for unsupervised problems if *clustering* is disabled.
- *class_weights*: A string to indicate how relatively important each class is for predictive correctness. This is done by providing a numeric value to each class. Note that the class name is case-sensitive. The following is an example *class_weights* setting:

  `class_weights = "{'BENIGN': 4, 'MALIGNANT': 6}"`

- *lead_time_days* (*nbm* only): Expected input/type: *integer*. Default value is `60`. The number of days prior to failure when the behavior starts trending toward either abnormal behavior or failure.
- *nbm_window_size* (*nbm* only): Expected input/type: *integer*. Default value is `256`. The number of sample points to consider for each failure detection.
- *nbm* (*nbm* only): Expected input/type: *True/False*. Default value is `False`. Set value to `True` for a normal behavioral model (NBM).
- *failure_dates* (*nbm* only): Expected input/type: *string*. List of failure dates to use for the calculation. Currently, only a list of one date can be used in the query. Example date format: `"07/15/2015"`
- *recovery_dates* (*nbm* only): Expected input/type: *string*. List of recovery dates to use for the calculation. Currently, only a list of one date can be used in the query. Example date format: `"11/15/2015"`

Payload:

```
{
  "dataset_names": ["dataset_name1"],
  "val_size": 0.2,
  "cv_kfold": 0,
  "job_name": "string",
  "model_name": "string",
  "loss_fn_name": "CrossEntropy",
  "fitness_fn_name": "Accuracy",
  "max_train_time": "00:01",
  "max_epochs": 0,
  "recurrent": True,
  "clustering": True,
  "anomaly": False,
  "n_clusters": 5,
  "forecast_horizon": 0,
  "anomaly_prior": 0.01,
  "class_weights": "string",
  "lead_time_days": 60,
  "nbm_window_size": 256,
  "nbm": False,
  "failure_dates": ["string"],
  "recovery_dates": ["string"],
  "fit_profile_name": "string"
}
```

Response Codes: 201, 400, 401, 403, 404, 408, 422

Successful Response:

```
{
    "job_name": "string",
    "job_id": "string",
    "model_name": "string"
}
```

Request Type: PATCH

URI: /v1/train/model/*{model_name}*

Headers:

- Authorization: Bearer token

Description: Resume training for a model on the dataset identified by *dataset_names*.

Parameter Descriptions:

- *dataset_names*: A list of dataset names to use for training.
  Note: Using only 1 dataset is currently supported.
- *job_name*: The job name
- *max_train_time* (supervised only): Sets the training time for the model in 'HH:MM' format. Default value is `00:01`.

- *max_epochs*: Sets the training time for the model in epochs. Default value is `10`.

Payload:

```
{
  "dataset_names": ["dataset_name1"],
  "job_name": "my_job",
  "max_train_time": "00:01",
  "max_epochs": 0
}
```

Response Codes: 201, 401, 403, 404, 408, 422

Successful Response:

```
{
    "job_name": "string",
    "job_id": "string",
    "model_name": "string"
}
```

Request Type: DELETE

URI: /v1/train/model/*{model_name}*

Headers:

- Authorization: Bearer token

Description: Delete a model.

Form Data:

- *model_name*: - Name of the model to delete.

Response Codes: 204, 400, 401, 403, 404, 408, 422

Successful Response: None

---

upload

Request Type: POST

URI: /v1/upload

Headers:

- Authorization: Bearer token

Description: Upload a dataset.

Form Data:

- *dataset*: a dataset file in a supported format (.csv, .h5)

  Note: For .csv files, ensure they are encoded to one of the following standards:

  - utf-8
  - us-ascii
  - iso-8859-1
  - iso-8859-2
  - ascii

- *dataset_name*: the name for the uploaded dataset
  Note: If not set, a guid will be provided

Response Codes: 201, 400, 401, 403, 408, 413, 422

Successful Response:

```
{
    "dataset_name": "name_of_dataset"
}
```

Request Type: DELETE

URI: /v1/upload/*{dataset_name}*

Headers:

- Authorization: Bearer token

Description: Delete a dataset.

Form Data:

- *dataset_name*: Name or identifier of dataset to delete.

Response Codes: 204, 401, 403, 404, 422

Successful Response: None

Contact Support

The following methods enable you to research issues, create a support ticket, or contact SparkCognition:

- Use the Darwin support portal - Read Frequently Asked Questions (FAQ), download documentation, or log your issue.
- **Email Support** - Send email to darwin_support@sparkcognition.com.
- **Phone Support** - The SparkCognition support line is +1-512-400-2001.

Revision Table

Version	Date	Notes
v 1.0	02-Feb-2018	First Release
v 1.1	15-Feb-2018	added types: supervised and ensembled
v 1.2(pre)	16-Mar-2018	added Status: Type= PATCH
v 1.2	27-Mar-2018	Added or changed: • /v1/job/status/*{job_name}* • /v1/lookup/user • /v1/lookup/username/*{username}* • /v1/train/model • /v1/run/model/{model_name}/*{dataset_name}* Name change: /v1/lookup/client to /v1/lookup/limits

Version	Date	Notes
v 1.3	23-May-2018	Added or changed: • /v1/analyze/model/*{model_name}* • /v1/analyze/model/predictions/*{model_name}*/*{dataset_name}* • /v1/auth/email • /v1/auth/password/reset • /v1/auth/register • /v1/train/model • /v1/train/model/*{model_name}* Name change: /v1/lookup/client to /v1/lookup/limits
v 1.3.1	14-Jun-2018	Edits to: • /v1/job/status/ • /v1/download/artifacts • Model uses example
v 1.4	31-Jul-2018	Edits to: • /v1/analyze/model/*{model_name}* • /v1/analyze/data/*{dataset_name}* • /v1/lookup/model • /v1/lookup/model/*{model_name}* • /v1/train/model • /v1/train/model/*{model_name}*
v 1.5	15-Oct-2018	Added: • /v1/clean/dataset/*{dataset_name}* • /v1/download/dataset/*{dataset_name}* • /v1/download/model/*{model_name}* Edits to: • /v1/analyze/data/*{dataset_name}* • /v1/lookup/model • /v1/train/model • /v1/download/artifacts/*{artifact_name}*
v 1.6	16-Jan-2019	Added: • /v1/lookup/model/*{model_name}*/population Edits to: • /v1/analyze/model/predictions/*{model_name}*/*{dataset_name}* • /v1/analyze/model/*{model_name}* • /v1/clean/dataset/*{dataset_name}* • /v1/download/model/*{model_name}* • /v1/train/model • /v1/run/model/*{model_name}*/*{dataset_name}*
v 1.6.1	06-Feb-2019	Fixed issues only. See Release Notes. Added on-prem installation notes.

Version	Date	Notes
v 1.6.2	22-Mar-2019	Fixed issues only. See Release Notes.
v 1.33.0	16-May-2019	Major change to version number to facilitate independent releases of the API Edits to: • /v1/train/model • /v1/info • /v1/analyze/data/*{dataset_name}*
v 1.34.0	22-Jul-2019	Edits to: • /v1/train/model: Added forecast_horizon, class_weights, cv_kfold, fit_profile_name
v 1.34.1	04-Sep-2019	Edits to: • /v1/download/model to facilitate the RTE
v 1.36.0	18-Dec-2019	Edits to: Added model_type parameter to run_model() Added impute parameter to clean_data() Added fitness_fn_name to create_model() • /v1/run/model to add model_type parameter • /v1/clean/dataset to add impute parameter • /v1/train/model to add finess_fn_name