# 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.

# 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*.

## 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
- `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*: 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.

**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.

**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*: 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*: 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.

**Description:** Analyze specific feature importances for a particular sample or samples given the model name and sample data.

**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.

**Parameter Descriptions**:

- *dataset_name*: 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. An email will be sent to the user's email address with a temporary password and instructions for changing it.

**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

**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. **Note:** */analyze/data* must be run first before *clean_data()* can be run. */clean/dataset* is only used for visualizing what Darwin would do or for when you want to use the cleaned data outside of Darwin. Do not clean data and then train on the cleaned data with Darwin. Invoking */train/model* has its own cleaning function as part of the model creation process.

**Form Data:**

- *dataset_name*: Name of dataset to clean
- *job_name*: Name of job
- *artifact_name*: Name given to the cleaned dataset

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

**Successful Response:**

```
{
    "job_name": "string",
    "artifact_name": "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 an 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'*.

**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.

---

## 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_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",
    }
]
```

---

**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",
    "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": [
        "language_data"
    ],
    "artifact_names": null,
    "model_name": None
}
```

**Request Type:** PATCH

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

**Headers:**

- Authorization: Bearer token

**Description:** Stop a running job.

**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": {},
    "description": {"best_genome": "RandomForestClassifier", "recurrent": false}
}
```

---

**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,
        "updated_at": "20170924T000000",
        "categorical": False,
        "sequential": True,
        "imbalanced": True,
    },
    {
        "name": "dataset2_name",
        "mbytes": 3.5,
        "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,
    "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",
  "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",
  "parent_id": "string"
}
```

---

### run

**Request Type:** POST

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

**Headers:**

- Authorization: Bearer token

**Form Data:**

- *job_name*: The name of the job.
- *artifact_name*: The name of the artifact.
- *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*.

**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:**

- *target*: (Mandatory) String denoting target prediction column in input data.
- *dataset_names*: A list of dataset names to use for training.
  **Note:** Using only 1 dataset is currently supported.
- *job_name*: The job name.
- *model_name*: The string identifier of the model to be trained.
- *loss_func*: Specify the loss function. Possible values include: *"CrossEntropy"*, *"MSE"*, *"BCE"*, *"L1"*, *"NLL"*, *"BCEWithLogits"*, *"SmoothL1"*. *"CrossEntropy"* 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.

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

- *max_epochs* (unsupervised only): 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.
  **Note**: This option is automatically enabled if a *datetime* column is detected in the input data. This may result in slower model evolution.
- *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 |
| 'mode' | Mode Fill: Uses the most common value on a column-by-column basis to fill in missing examples. The result is interpretable for both numerical and categorical variables. | Linear Fast |
| 'spline' | Spline Fill: Interpolation using a spline (piecewise function). Might be useful for timeseries or sequential data. | Linear Fast |
| 'linear' | Linear Interpolation Fill: Interpolation using a Linear function. Might be useful for timeseries or sequential data. | Linear Fast |

- *drop*: Enables automatic pruning of input columns based on different criteria such as amount of missing data, number of unique values, and standard deviation. Possible values are: *'hard'*, *'soft'*, or *'no'*.
  **Note**:This automatically drops identifier columns (unique value for each sample) and columns that do not contain sufficient data to aid prediction.
- *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.
- *feature_eng*: Enables automatic feature generation. Identifies an appropriate time window and augments input with new features derived in the frequency and time domains.
  **Note**: Can only be applied to timeseries data. String aliases specify methods for window computation.

| ALIAS | DESCRIPTION |
|---|---|
| 'mi' | Uses mutual information to estimate the window length. |
| 'auc' | (**Default**) Uses autocorrelation to estimate the window length. |
| 'user' | User specified window length: see *window_len* below. |

- *window_len*: Expected input/type: *integer*. User specified window length for feature generation.
  **Note**: This parameter is used only in the case that *user* is provided for the *feature_eng* parameter.
- *feature_select*: A number in [0,1] specifying the percentage of numerical features to maintain based on their dependency to the target. Ranks all features using mutual information and drops (1 - feature_select)% of the lowest-ranking features. **Default is 1** (keep all features).
- *outlier*: A string alias that indicates the outlier detection to apply during preprocessing.
  **Note**: Outliers are removed and later filled using imputation.

| ALIAS | DESCRIPTION |
|---|---|
| None | (**Default**) No outlier detection applied. |
| 'mad' | Uses Median Absolute Deviation to detect outliers. |
| 'perc' | Uses Percentile-based outlier detection. |
| 'isol' | Uses an Isolation Forest to detect outliers. |

- *imbalance* (*supervised* only): Expected input/type: *true/false*. Enables automatic imbalance correction that selectively applies *random oversampling*, *random undersampling*, *synthetic minority oversampling* (SMOTE), or *adaptive synthetic sampling* (ADASYN) to the input data depending on problem characteristics.
- *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.
- *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.
- *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: `"07/01/2015"`
- *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: `"11/01/2015"`

**Payload:**

```
{
  "target": "string",
  "dataset_names": ["dataset_name1"],
  "job_name": "my_job",
  "model_name": "string",
  "loss_func": "CrossEntropy",
  "max_train_time": "00:01",
  "max_epochs": 0,
  "recurrent": true,
  "impute": "mean",
  "drop": "no",
  "max_int_uniques": 15,
  "max_unique_values": 50,
  "feature_eng": "mi",
  "feature_select": 1,
  "outlier": "mad",
  "imbalance": true,
  "n_clusters": 5,
  "anomaly_prior": 0.01,
  "lead_time_days": 60,
  "nbm_window_size": 256,
  "nbm": false,
  "return_risk": true,
  "failure_dates": ["string"],
  "recovery_dates": ["string"]
}
```

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

**Successful Response:**

```
{
    "job_name": "nameofjob",
    "model_name": "nameofmodel",
}
```

---

**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* (unsupervised only): 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": "nameofjob",
    "model_name": "nameofmodel",
}
```

---

**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)
- *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

---

## Examples

The following sections provide examples of how to use the Darwin API.

## Login

1. Login using the `/v1/auth` routes. It is possible to login as either a *service* or a *user*.

| Login | Request |
|---|---|
| Login as service: | **Request Type: POST**<br>**URI**: /v1/auth/login<br>**Form Data**:<br>*api_key*: The api key of the service<br>*pass1*: The service level password |
| Login as an end user: | **Request Type: POST**<br>**URI**: /v1/auth/login/user<br>**Form Data**:<br>*api_key*: The api key of the service<br>*username*: The end user name<br>*pass1*: The end user level password |

2. Receive token. If login is successful, a response arrives with an access token. This token is used in the *authorization header* for other requests:

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

*Note:* The token (in this example "some string") must be prepended with the string `Bearer`.
For example the token becomes:
`Bearer MyNewTokenString` - (that is: **Bearer(space)MyNewTokenString**).

## Train a model - supervised

1. Upload a dataset using the following:

   **Request Type**: POST

   **URI**: /v1/upload

   **Headers:**

   - Authorization: Bearer token

   **Form Data**:

   - *dataset*: a dataset file
   - *dataset_name*

   **Notes**:

- Assign a name to the dataset. If no name is assigned, a random string is assigned in its place. It is necessary to keep track of lookup current datasets with the `/v1/lookup/dataset` route.
- Ensure not to exceed upload limits. If the limits are exceeded, a *403 forbidden error* is generated. To fix the exceeded limit, delete a dataset that is older or no longer required.

2. Set target parameter.
   Use the uploaded dataset to train a model. Specify the dataset name in the URI of the train route. Note that training a supervised model requires the *target* parameter is set:

   **Request Type**: POST

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

   **Headers:**

   - Authorization: Bearer token

   **Payload**:

```
{
  "target": "string",
  "dataset_names": ["dataset_name1"],
  "job_name": "nameofjob",
  "artifact_name": "string",
  "model_name": "string",
  "max_train_time": "00:01",
  "max_epochs": 0,
  "recurrent": true,
  "impute": "mean",
  "drop": "no",
  "max_int_uniques": 15,
  "max_unique_values": 50,
  "feature_eng": "mi",
  "feature_select": 1,
  "outlier": "mad",
  "imbalance": true,
  "n_clusters": 5,
  "anomaly_prior": 0.01
}
```

   **Note**: Because many of the payload parameters are optional, depending on your use case, it is possible to use a simple payload, for example:

```
{
  "target": "string",
  "dataset_names": ["dataset_name1"]
}
```

3. Consult return.
   In response to the payload, a job name and model name are returned. Note that if the dataset was not named, a random string is returned as its name. For example:

```
{
    "job_name": "nameofjob",
    "model_name": "nameofmodel",
}
```

4. Check job status.
   When the job name is returned, use the `/v1/job/status` or `/v1/job/status/{job_name}` route together with the job name to check the job status. For example:

   **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*.

   The query return contains information about how far the job has progressed, as a percent_complete and status, the number of generations run (so far), and the model loss. For example:

```
{
        "status": "Requested",
        "percent_complete": 0,
        "loss": 0,
        "generations": 0,
        "dataset_names": [
            "phone_data"
        ]
        "model_name": "nameofmodel",
    }
```

   When the job completes, the percent_complete shows `100` and status is set to `Complete`. The generated model can be used for additional tasks, described below.

## Train a model - unsupervised

The process to train an unsupervised model follows the same procedure as the supervised model training procedure. The difference is the target parameter in the `/v1/train/model/{dataset_name}` route is left unspecified. Depending on the use case, it is possible to simplify the payload to a set of empty braces:

```
{}
```

## Analyze a dataset

1. If necessary, upload a dataset using the instructions and the `/v1/upload` route.

2. Analyze the dataset through the `/v1/analyze/data/{dataset_name}` route:

   **Request Type**: POST

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

   **Headers:**

   - Authorization: Bearer token

   Payload:

```
{
    "job_name": "string",
    "artifact_name": "string",
    "max_unique_values": 0
}
```

3. Consult the return.
   The post action returns job and artifact names. The job name enables monitoring the job status.
4. Download the analysis.
   When the job completes, download the data analysis with the `/v1/download/artifacts/{artifact_name}` route.

## Analyze a model

Analyze the trained model.

1. Use the the `/v1/analyze/model/{model_name}` route:

   **Request Type**: POST

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

   **Headers:**

   - Authorization: Bearer token

   **Form Data:**

   - *job_name*: The name of the job
   - *artifact_name*: The name of the artifact

2. Consult the return.
   The post action returns job and artifact names. The job name enables monitoring the job status.
3. Download the analysis.
   When the job completes, download the model analysis with the `/v1/download/artifacts/{artifact_name}` route.
   **Note**: The model analysis takes the form of a pandas Series.

### Model uses

A trained model and dataset can be used for prediction, classification, or for detecting data clusters and/or anomalies.
After a model is trained, additional datasets can be uploaded and the model applied against those additional datasets. To perform these tasks:

1. Use the `/v1/run/model/{model_name}/{dataset_name}` route:

   **Request Type**: POST

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

   **Headers:**

   - Authorization: Bearer token

   **Form Data:**

   - *model_name*: The name or identifier of the model.
   - *dataset_name*: The name or identifier of the dataset.

2. Consult the return.
   The post action returns job and artifact names. The job name enables monitoring the job status.

3. Download the results.
   When the job completes, download the results with the `/v1/download/artifacts/{artifact_name}` route.

---

## 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:<br>• /v1/job/status/*{job_name}*<br>• /v1/lookup/user<br>• /v1/lookup/username/*{username}*<br>• /v1/train/model<br>• /v1/run/model/{model_name}/*{dataset_name}*<br>Name change: /v1/lookup/client to /v1/lookup/limits |

| Version | Date | Notes |
|---|---|---|
| v 1.3 | 23-May-2018 | Added or changed:<br>• /v1/analyze/model/*{model_name}*<br>• /v1/analyze/model/predictions/*{model_name}*/*{dataset_name}*<br>• /v1/auth/email<br>• /v1/auth/password/reset<br>• /v1/auth/register<br>• /v1/train/model<br>• /v1/train/model/*{model_name}*<br>Name change: /v1/lookup/client to /v1/lookup/limits |
| v 1.3.1 | 14-Jun-2018 | Edits to:<br>/v1/job/status/<br>/v1/download/artifacts<br>Model uses example |
| v 1.4 | 31-Jul-2018 | Edits to:<br>/v1/analyze/model/*{model_name}*<br>/v1/analyze/data/*{dataset_name}*<br>/v1/lookup/model<br>/v1/lookup/model/*{model_name}*<br>/v1/train/model<br>/v1/train/model/*{model_name}* |