SparkCognition Darwin Python SDK Guide

Contents

About this guide

This guide describes using the Darwin™ SDK to access and use the Darwin API 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, test the generated models, monitor jobs and perform analysis. The SDK also provides some convenience functions. Note that throughout this document, long key and token values are truncated, indicated by ellipses (**. . .**).

Expectation

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

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.
The general workflow for simple modeling includes:

- Upload training data
- Create model
- Upload test data
- Test the model
- Download result artifact

Note: Darwin expects all uploaded ingestion files to be in a *rectangular* format. This means a flat file with features that span columns and data samples that span rows. Plan your data file so it fits this expectation to help prevent errors.

See the SDK example for a modeling example.

For additional information on Darwin, contact your local SparkCognition partner for access to the white paper titled: *Darwin - A Neurogenesis Platform*.

Accessing the API

This document describes the python SDK and explains how to access the Darwin API and its functionality. Additional methods to access the Darwin API include:

- through the `https://darwin-api.sparkcognition.com/v1` end point
- optionally, through user created `curl` commands

For additional information on the Darwin API, contact your local SparkCognition partner for access to see the *SparkCognition Darwin API User Guide*.

Notes:

- An *API key* is necessary to use the Darwin SDK.
  Contact SparkCognition or your IT manager for an appropriate key.
- All methods return a 2-tuple, for example:

```
(True,  <context-dependent-return-object> )
(False,  <some-helpful-message> )
```

Darwin SDK interface

Setup Darwin SDK

Perform the following to download and setup the Darwin SDK:

1. Install Python 3.5 or greater. Alternatively, install *Miniconda*, from https://conda.io/miniconda.html.
2. Create a directory to receive the git repository clone.
3. Change (*cd*) into the new directory.
4. Clone the *darwin-sdk* repository:

   ```
   git clone https://github.com/sparkcognition/darwin-sdk
   ```

5. Change into the new root directory of the *darwin-sdk* cloned darwin-sdk project:

   ```
   cd <NewCloneRootDirectory>
   ```

   Note: By default this is the *master* trunk.

6. Ensure code is from master trunk:

   ```
   git pull
   ```

7. Setup the SDK:

   ```
   python setup.py install
   ```

   The SDK defaults to using the production URL: https://darwin-api.sparkcognition.com/v1/

Connect to the Darwin interface

1. Obtain an api key.

   To use the Darwin SDK, an api key is required. A key can be obtained from SparkCognition support or your IT manager. An api_key is a long string, for example:

   ```
   'RsJ74ZS5AvwznbHh0AfVSgrchhS9KxACDy3jefaQnxb9f6QTSDBFmhnGa0cOIWtNAIFRAG9ToOTpi0mn'
   ```

2. Register the api key.

   The purpose of this method is to set a password and an email address for an api_key. Each api_key is synonymous with a service. This method must be invoked once for each api_key to establish a password for that key.

   Note: After successful registration, the service uses `auth_login()` to login as a service.

   Example

   ```
   >>> from amb_sdk.sdk import DarwinSdk
   >>> s = DarwinSdk()
   >>> s.auth_register('asdf', 'iRgwut4kGs0ymULiuKtMd0WFvBYLMWSj16q2uysQeteqH9ssc+\
   EETUvcysnPojRpfycLVHa2IlN1IlrfEk1YMA', 'admin@company.com')
   ```

```
(True,'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1MTU1MzM4NjEsImlh\
dCI6MTUxNTUzMDI2MS ... F56xZQiBT-89nrRz1nIXD5LfawHIj_MlUHQqM36vU')
```

3. Login as a *service* or *create a user* under the service and login as a user.
   The following explains how to log in as a *service*.

   Notes:

   - Although `Bearer <auth-token>`, returned by `auth_login()`, is used in subsequent calls to validate authenticity, it is not required for each method.
   - The SDK remembers the auth token for the DarwinSdk object. Although an auth token is currently valid for 1 hour, if the DarwinSdk object session life time exceeds 1 hour, the SDK will request another auth token until the session ends.

   Example

```
>>> s.auth_login('asdf', 'iRgwut4kGs0ymULiuKtMd0WFvBYLMWSj16q2uysQeteqH9ssc+EET\
UvcysnPojRpfycLVHa2IlN1IlrfEk1YMA')
(True,'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1MTU1MzQxNzIsImlh\
dCI6MTUxNTUzMD ... UQQfoXqYFKJSoRXXDNPE985-a08cE6_o')
```

   When login (as a service) successfully completes, the SDK can be used to create and model a workflow.

   Note, there are also `auth_register_user()` and `auth_login_user()` methods that allow you to create users and login as a specific user. You can choose to use the SDK as a service or create users underneath the service to partition datasets/models to be owned by specific users. It is more convenient to employ user accounts because the api_key is not necessary for logging in as a user.

   Example

```
>>> s.auth_register_user('atestuser', 'apassword', 'anemail')
(True,
 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiJkNjY0MmJjOC1iMmU5LTQxO\
DctODFlNS00YjI2MD ... 5zMp_1FfxU')

>>> s.auth_login_user('atestuser', 'apassword')
(True,
 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiI3NGYzYmUxZS0yOTlmLTRhN\
zMtODU5ZC01NGRmM2F ... u1zGCeCONA')
```

4. Verify the connection.

   The default url in the SDK is `https://darwin-api.sparkcognition.com/v1/`. Use `get_url()` and `set_url()` to verify connection to the right Darwin service.

Darwin SDK methods

URL Get/Set methods

DarwinSdk.get_url()

Get Darwin service url.

Parameters: None

Returns:

```
    (True, <url-string>)
```

Example

```
In [10]: s.get_url()

Out[10]: (True, 'https://darwin-api.sparkcognition.com/v1/')
```

DarwinSdk.set_url(*url, version=‘v1’*)

Set Darwin service url and version.

Parameters:

- **url** - URL to the Darwin service
- **version** - Set to ‘v1’

Returns:

```
   (True, <url>) or (False, 'invalid url')
```

Example

```
In [9]: s.set_url('https://darwin-api.sparkcognition.com/v1/')

Out[9]: (True, 'https://darwin-api.sparkcognition.com/v1/')
```

Authentication methods

DarwinSdk.auth_register(*password, api_key, email*)

Register as a service. The purpose of this method is to set a password for an api_key. Each api_key is synonymous with a service. This method is invoked only once for each api_key to establish a password for that key. After registration, the service can use *auth_login*() to login as a service.

Parameters:

- *password* - The service level password
- *api_key* - The api key for the service
- *email* - Email address

Returns:

```
(True, 'Bearer <auth-token>') or (False, <error-message>)
```

`Bearer <auth-token>` is used in subsequent calls to validate authenticity.
The SDK remembers the auth token for the DarwinSdk object.
Note: Although an auth token is currently valid for 1 hour, if the DarwinSdk object session life time exceeds 1 hour, the SDK will request another auth token until the session ends.

Example

```
In [4]: s.auth_register('asdf', 'RsJ74ZS5AvwznbHh0AfVSgrchhS9KxACDy\
3jefaQnxb9f6QTSDBFmhnGa0cOIWtNAIFRAG9ToOTpi0mnEo3zFA', 'email')
Out[4]:
(True,
 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiO...iSdU8xlF4yJk')
```

DarwinSdk.auth_login(*password, api_key*)

Login as a service.
Note: A service must have a password set using *auth_register()* to login successfully.

Parameters:

- *password* - The service level password
- *api_key* - The api key for the service

Returns:

```
(True, 'Bearer <auth-token>') or (False, <error-message>)
```

`Bearer <auth-token>` is used in subsequent calls to validate authenticity. The SDK remembers the auth token for the DarwinSdk object.
Note: Although an auth token is currently valid for 1 hour, if the DarwinSdk object session life time exceeds the 1 hour limit, the SDK will acquire another auth token until the session ends.

Example

```
In [5]: s.auth_login('asdf',
'iRgwut4kGs0ymULiuKtMd0WFvBYLMWSj16q2uysQeteqH9ssc+EETUvcysnPojRpfyc\
LVHa2IlN1IlrfEk1YMA')

Out[5]:

(True,
'Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1MTU1MzQxN....')
```

DarwinSdk.auth_register_user(username, *password*, *email*)

Register a user. This method registers a new user.
Note: You must be logged in as a service to create a user.

Parameters:

- *username* - The new end user's username
- *password* - The new end user's password
- *email* - The new end user's email address

Returns:

```
    (True, 'Bearer <auth-token>') or (False, <error-message>)
```

`Bearer <auth-token>` is used in subsequent calls to validate authenticity. The SDK remembers the auth token for the DarwinSdk object.
Note: Although an auth token is currently valid for 1 hour, if the DarwinSdk object session life time exceeds the 1 hour limit, the SDK will acquire another auth token until the session ends.

Example

```
In [8]: s.auth_register_user('user1', 'user1-password', 'user-email',
'iRgwut4kGs0ymULiuKtMd0WFvBYLMWSj16q2uysQeteqH9ssc+EETUvcysnPojRpfycLV\
Ha2IlN1IlrfEk1YMA')

Out[8]:

(True,

'Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1MTU1MzQyN....')
```

DarwinSdk.auth_set_email(*username*, *email*)

Add or change a user's email address.

Parameter:

- *username* - The end user's username
- *email* - The end user's email address

Returns:

```
(True, None) or (False, <error-message>)
```

User must be logged in to add or change an email address. This email address will be used for password resets and other notifications.

Example

```
In [9]: s.auth_set_email('user1', 'user1@company.com')

Out [9]: (True, None)
```

DarwinSdk.auth_login_user(*username, password*)

Login as a user.
Note: A user must have a username and password set using **auth_register_user**() to successfully login.

Parameters:

- *username* - The end user's username
- *password* - The end user's password

Returns:

```
(True, 'Bearer <auth-token>') or (False, <error-message>)
```

`Bearer <auth-token>` is used in subsequent calls to validate authenticity. The SDK remembers the auth token for the DarwinSdk object.
Note: Although an auth token is currently valid for 1 hour, if the DarwinSdk object session life time exceeds the 1 hour limit, the SDK will acquire another auth token until the session ends.

Example

```
In [9]: s.auth_login_user('user1', 'user1-password',
'iRgwut4kGs0ymULiuKtMd0WFvBYLMWSj16q2uysQeteqH9ssc+EETUvcysnPojRpfycLV\
Ha2IlN1IlrfEk1YMA')

Out[9]:

(True,

'Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1MTU1MzQzM....')
```

DarwinSdk.auth_change_password(*curpass, newpass*)

Change the current user's password.

Parameters::

- *curpass* - User's current password
- *newpass* - User's new password

Returns:

```
(True, None) or (False, <error-message>)
```

User must be logged in to change password. If the current password is forgotten, use the following **DarwinSdk.auth_reset_password (username)** method to reset it. An email will be generated with a temporary password.

Example

```
In [10]: s.auth_change_password('mypassword', 'newpassword')

Out[10]: (True, None)
```

DarwinSdk.auth_reset_password(*username*)

Reset a user's password. A temporary password will be sent to the user's email address.

Parameter:

- *username* - Username to reset password for.

Returns:

```
    (True, <deleted-user-id>) or (False, <error-message>)
```

Example

```
In [8]: s.auth_reset_password('username')

Out[8]: (True, None)
```

DarwinSdk.auth_delete_user(*username*)

Remove/Unregister a user.

Parameter:

- *username* - Username of the user to be deleted.

Returns:

```
    (True, <deleted-user-id>) or (False, <error-message>)
```

You must be logged in with your api_key and password to delete a user.

Example

```
In [8]: s.lookup_username('testuser2')
Out[8]:
(True,
 [{'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrch...Eo3zFA',
   'created_at': '2018-01-03T12:54:30.653478',
   'parent_id': '2516b462-df85-11e7-bdd1-e37424c63ea4',
   'tier': 0,
   'username': 'testuser2'}])

In [9]: s.auth_delete_user('testuser2')
Out[9]: (True, '87d721fc-f0b7-11e7-b58d-a3441423b160')

In [10]: s.auth_delete_user('testuser2')
Out[10]:
(False,
 '404: NOT FOUND - {\n    "message": "Username not found. \
 You have requested this URI [/v1/auth/register/user/testuser2] \
 but did you mean /v1/auth/register/user/ <username> or /v1/auth/register/user \
 or /v1/auth/register ?"\n}\n')
```

Job status methods

DarwinSdk.lookup_job_status(*age*=*None*, *status*=*None*)

Get status information for all jobs belonging to the current user or service.

Parameters:

- *age* - (optional) Filter jobs that are less than *X* units old, for example 3w, 2d, or 1h.
- Optional parameters:
  - *status* - If not specified, returns all jobs.
  - *running* (Note that only 2 jobs can be running concurrently.)
  - *requested*
  - *complete*
  - *failed*

Returns:

```
(True, <list-of-jobs>) or (False, <error-message>)
```

Example

```
In [6]:  s.lookup_job_status(status='Complete')
Out[6]:
(True,
 [{'artifact_names': None,
   'dataset_names': ['cancer-train'],
   'endtime': '2018-02-01T10:53:50.451598',
   'generations': 0,
   'job_name': 'eeef500d629e4a2185eb8af6e18a83b4',
   'job_type': 'TrainModel',
   'loss': 2.0,
   'model_name': 'cancer-model',
   'percent_complete': 100,
   'starttime': '2018-02-01T10:52:42.280929',
   'status': 'Complete'}])
```

DarwinSdk.lookup_job_status_name(*job_name*)

Get job status information for a job by its name.

Parameters:

- *job_name* - The name of the job you want status on

Returns:

```
(True, <job-info>) or (False, <error-message>)
```

Example

```
In [19]: s.lookup_job_status_name('eeef500d629e4a2185eb8af6e18a83b4')
Out[19]:
(True,
 {'artifact_names': None,
  'dataset_names': ['cancer-train'],
  'endtime': None,
  'generations': 0,
  'job_error': "MultipleDateColumns: multiple date columns \
    - ['Date' 'PeakMonth' 'PeakQuarter']",
  'job_type': 'TrainModel',
  'loss': None,
  'model_name': 'cancer-model',
  'percent_complete': 0,
  'starttime': '2018-02-01T10:52:42.280929',
  'status': 'Running'})

In [20]: s.lookup_job_status('Running')
```

DarwinSdk.delete_job(*job_name*)

Delete a job.

Parameter:

- *job_name* - The name of the job you want to delete

Returns:

`(True, None) or (False, <error-message>)`

Example

```
In [17]: s.lookup_job_status_name('7df54dfddfa046d581522f7540e3256c')
Out[17]:
(True,
 {'artifact_names': ['7a245119ca3b42efadc27006e75a225d'],
  'dataset_names': ['market-train'],
  'endtime': '2018-03-06T14:23:59.975793',
  'generations': None,
  'job_error': '',
  'job_type': 'AnalyzeData',
  'loss': None,
  'model_name': None,
  'percent_complete': 100,
  'starttime': '2018-03-06T14:23:57.18095',
  'status': 'Complete'})

In [18]: s.delete_job('7df54dfddfa046d581522f7540e3256c')
Out[18]: (True, None)

In [19]: s.lookup_job_status_name('7df54dfddfa046d581522f7540e3256c')
Out[19]: (False, '404: NOT FOUND - {\n    "message": "Job name not found"\n}\n')
```

DarwinSdk.stop_job(*job_name*)

Stop a job.

Parameter:

- *job_name* - The name of the job.

Returns:

`(True, 'Job is scheduled to stop') or (False, <error-message>)`

Example

```
In [21]: s.stop_job('34787793a48b42b48a319bbbf68f13ea')
Out[21]: (True, 'Job is scheduled to stop')
```

Lookup methods

DarwinSdk.lookup_artifact(*type=None*)

Get a list of artifacts belonging to the current user or service.

Parameter:

- *type* - (optional) specifies the type of artifact. Values can be ’Model’, ’Dataset’, ’Run’.

Returns:

`(True, <artifact-list>) or (False, <error-message>)`

Example:

```
In [30]: s.lookup_artifact('Run')
http://localhost:5000/v1/lookup/artifact
Out[30]:
(True,
 [{'created_at': '2018-02-01T11:09:55.731040',
   'id': 'b9a9205a-0772-11e8-a003-3b1c8766dad0',
   'mbytes': 0.0,
   'name': '8a63e21030d1483abb0f892963c1728f',
   'type': 'Run'},
  {'created_at': '2018-02-01T11:11:17.560360',
   'id': 'ea6f3f80-0772-11e8-9abe-77bc32e350c5',
   'mbytes': 0.0,
   'name': 'artifact-1',
   'type': 'Run'}]
```

DarwinSdk.lookup_artifact_name(*artifact_name*)

Get information for an artifact specified by its name.

Parameter:

- *artifact* - specifies an artifact by its name

Returns:

`(True, <job-info>) or (False, <error-message>)`

Example:

```
In [31]: s.lookup_artifact_name('artifact-1')
Out[31]:
(True,
 {'created_at': '2018-02-01T11:11:17.560360',
  'mbytes': 0.0,
  'name': 'artifact-1',
  'type': 'Run'})
```

DarwinSdk.lookup_limits()

Get a client's metadata. A client is the current user or service in context.

Parameters: None

Returns:

```
(True, <client-info>) or (False, <error-message>)
```

Example

```
In [21]: s.lookup_limits()
Out[21]:
(True,
 {'job_limit': None,
  'model_limit': None,
  'tier': 0,
  'upload_limit': None,
  'user_limit': None,
  'username': None})
```

DarwinSdk.lookup_dataset()

Get the dataset(s) metadata for a user. The user is the current user or service in the current context. This is useful for listing all created datasets.

Parameters: None

Returns:

```
(True, <list-of-dataset-info>) or (False, <error-message>)
```

Example

```
In [4]: s.lookup_dataset()
Out[4]:
(True,
 [{'categorical': None,
   'imbalanced': None,
   'mbytes': 0.020199775695800781,
   'minimum_recommended_train_time': "string"
   'name': 'unittest-cancer-dataset2',
   'sequential': None,
   'updated_at': '2018-01-31T15:37:28.310994'},
  {'categorical': None,
   'imbalanced': None,
   'mbytes': 0.02019977569580078,
   'minimum_recommended_train_time': "string"
   'name': 'cancer-train',
   'sequential': None,
   'updated_at': '2018-02-01T10:52:06.076279'}])
```

DarwinSdk.lookup_dataset_name(*dataset_name*)

Get a specific dataset's metadata.

Parameters:

- *dataset_name* - The name of the dataset. The name of a dataset is established in the **upload_dataset**() method.

Returns:

```
(True, <dataset-info>) or (False, <error-message>)
```

Example

```
In [36]: s.lookup_dataset_name('cancer-train')
Out[36]:
(True,
 {'categorical': None,
  'imbalanced': None,
  'mbytes': 0.02019977569580078,
  'minimum_recommended_train_time': "string"
  'sequential': None,
  'updated_at': '2018-02-01T10:52:06.076279'})
```

DarwinSdk.lookup_model()

Get the model(s) metadata for a user. The user is the current user or service in the current context. This is useful for listing all models.

Parameters: None

Returns:

```
(True, <list-of-model-info>) or (False, <error-message>)
```

Example

```
In [37]: s.lookup_model()
Out[37]:
```

```
(True,
 [{'generations': 0,
   'loss': 2.0,
   'name': 'cancer-model',
   'parameters': {'target': 'Diagnosis'},
   'trained_on': ['cancer-train'],
   'updated_at': '2018-02-01T10:53:50.443166',
   '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}
 }]
)
```

DarwinSdk.lookup_model_name(*model_name*)

Get a specific model's metadata. The name of a model is established in the *create_model()* method.

Parameters:

- *model_name* - The name of the model

Returns:

    (True, <model-info>) or (False, <error-message>)

Example

```
In [40]: s.lookup_model_name('cancer-model')
Out[40]:
(True,
 [{'generations': 0,
  'loss': 2.0,
  'parameters': {'target': 'Diagnosis'},
  'trained_on': ['cancer-train'],
  'updated_at': '2018-02-01T10:53:50.443166',
  '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}
 }]
)
```

DarwinSdk.lookup_tier()

Get metadata for all tiers. A tier specifies certain usage limits such as *number of models* and *datasets*.

Parameters: None

Returns:

```
(True, <list-of-tier-info>) or (False, <error-message>)
```

Example

```
In [41]: s.lookup_tier()
Out[41]:
(True,
 [{'job_limit': None,
   'model_limit': None,
   'tier': 0,
   'upload_limit': None,
   'user_limit': None},
  {'job_limit': 10000,
   'model_limit': 10000,
   'tier': 1,
   'upload_limit': 10000,
   'user_limit': 1000}])
```

DarwinSdk.lookup_tier_num(*tier_num*)

Get a specific tier's metadata. A tier specifies certain usage limits such as the *number of models* or *datasets*.

Parameters:

- *tier_num* - The number of the tier

Returns:

```
(True, <tier-info>) or (False, <error-message>)
```

Example

```
In [44]: s.lookup_tier_num(1)
Out[44]:
(True,
 {'job_limit': 10000,
  'model_limit': 10000,
  'tier': 1,
  'upload_limit': 10000,
  'user_limit': 1000})
```

DarwinSdk.lookup_user()

Returns information for users that were created with the current api_key.
Note: Each customer site is assigned a *unique api_key*. All users from that site have the same api_key.

Parameters: None

Returns:

`(True, <list-of-user-info>) or (False, <error-message>)`

Example

```
In [25]: s.lookup_user()
Out[25]:
(True,
 [{'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrchhS9...3zFA',
   'created_at': '2018-01-03T12:54:30.653478',
   'parent_id': '2516b462-df85-11e7-bdd1-e37424c63ea4',
   'tier': 0,
   'username': 'testuser2'},
  {'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrchhS9...3zFA',
   'created_at': '2018-01-03T13:14:36.188371',
   'parent_id': '2516b462-df85-11e7-bdd1-e37424c63ea4',
   'tier': 0,
   'username': 'testuser5'},
  {'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrchhS9...3zFA',
   'created_at': '2018-01-03T13:21:21.099148',
   'parent_id': '2516b462-df85-11e7-bdd1-e37424c63ea4',
   'tier': 0,
   'username': 'testuser6'}])
```

DarwinSdk.lookup_username(*username*)

Returns information for a user.

Notes:

- The user in question should have been created using the current api_key.
- Each customer site is assigned a *unique api_key*. All users from that site have the same api_key.

Parameters: None

Returns:

`(True, <user-info>) or (False, <error-message>)`

Example

```
In [26]: s.lookup_username('testuser2')
Out[26]:
(True,
 [{'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrchhS9...3zFA',
   'created_at': '2018-01-03T12:54:30.653478',
   'parent_id': '2516b462-df85-11e7-bdd1-e37424c63ea4',
   'tier': 0,
   'username': 'testuser2'}])
```

DarwinSdk.display_population(*model_name*)

Get a specific model's population data. The name of the model is established in the **create_model()** method.

Parameters:

- *model_name* - The name of the model

Returns:

`(True, <population-info>) or (False, <error-message>)`

Example

```
In [40]: s.display_population('cancer-model')
Out[40]:
        (True,
        {
          "population": {
            "model_types": {
              "DeepNeuralNetwork": {
                "model_description": [
                  {
                    "layer 1": {
                      "type": "LinearLayer",
                      "parameters": {
                        "activation": "leakyrelu",
                        "numunits": 221
                      }
                    }
                  },
                  {
                    "layer 2": {
                      "type": "LinearLayer",
                      "parameters": {
                        "activation": "relu",
                        "numunits": 2
                      }
                    }
                  }
                ],
                "loss_function": "CrossEntropy",
                "fitness": 1.9667300770467946
              },
              "RandomForest": {
                "model_description": {
                  "type": "RandomForestClassifier",
                  "parameters": {
                    "bootstrap": true,
                    ....
                  }
                },
                "loss_function": "CrossEntropy",
                "fitness": 1.9321841524601422
              }
            }
          }
        })
```

Datasets and artifact methods

DarwinSdk.upload_dataset(*dataset*, *dataset_name=None*)

Upload a dataset.

Note: Supported file formats are .csv and .h5.

Note: The maximum size that can be uploaded is 10GB. Files larger than ~2GB can be processed by analyze_data() only. Model creation might not be successful for files larger than ~2GB until Big Data is fully supported. Analyze_data() is the only method that supports Big Data.

Parameters:

- *dataset*- Path to dataset
- *dataset_name* - Name to be given to dataset, or defaults to filename

Returns:

```
(True, {dataset_name: <name-given-to-dataset>})  or (False, <error-message>)
```

Example

```
In [5]: s.upload_dataset('sets/cancer_train.csv', 'unittest-cancer-dataset')
Out[5]:
(True,
 {'dataset_name': 'unittest-cancer-dataset'})
```

DarwinSdk.download_dataset(*dataset_name*)

Download a dataset artifact given its name.

Parameters:

- *dataset_name* - Name of the dataset to be downloaded.

Returns:

```
(True, <path-to-file>) or (True, <python-list>) or (False, <error-message>)
```

Example

```
In [5]: s.download_dataset('cancer-cleandata3', \
    artifact_path='/Users/username/Downloads/artifacts')
Out[5]:
(True,
  {'filename': \
  '/Users/username/Downloads/artifacts/cancer-cleandata3-cleaned-8m38g07j.csv'})
```

DarwinSdk.delete_dataset(*dataset_name*)

Delete the named dataset.

Parameters:

- *dataset_name* - Name of the dataset to be deleted.

Returns:

```
(True, None)  or (False, <error-message>)
```

Example

```
In [6]: s.delete_dataset('unittest-cancer-dataset')
Out[6]:
(True, None)
```

DarwinSdk.download_model(*model_name*)

Download a supervised model given its name.

Parameters:

- *model_name* - Name of the model to be downloaded.
- *path* - (optional) Relative or absolute path of the directory to download the model to. This directory must already exist prior to model download. If the path is not 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: *DeepNeuralNetwork, RandomForest, GradientBoosted*.
- *model_format* - (optional) Format in which the model is to be downloaded. Possible values include: *json, onnx*.

Returns:

```
(True, None)  or (False, <error-message>)
```

Example

```
In [6]: s.download_model('my-model-name', path='Users/auser/Downloads/mymodel')
Out[6]:
(True, None)
% ls -l ~/Downloads/mymodel
total 272
-rw-r--r-- 1 auser staff 58609 Oct 10 15:55 data_profiler
-rw-r--r-- 1 auser staff 75507 Oct 10 15:55 model
```

DarwinSdk.download_artifact(*artifact_name*, *artifact_path=None*)

Download artifact given its name. The methods that return artifacts are:

- *analyze_data()*
- *analyze_model()*
- *analyze_predictions()*
- *run_model()*

Note: The artifact for *analyze_model()* is a pandas Series. The artifact displays a two-column series where the name of the feature is in the first column and the second column is a number between 0 and 1 indicating how much that feature influenced the model's predictions over the entire dataset that the model was trained on.

Note: The artifact for *analyze_predictions* is a pandas DataFrame. The artifact has one column for each feature that indicates how much that feature influenced the model's prediction, plus additional columns for the average model prediction ("base_value"), and the model prediction for each row ("predicted_value" for regression or "predicted-class" and "predicted_probability" for classification).

Parameters:

- *artifact_name* - Name of the artifact to download.
- *artifact_path*: *(optional)* Relative path of the directory to download the artifact to (only applicable for the artifacts where a temporary file is created). This directory must already exist prior to artifact download.

Returns:

```
(True, <path-to-file>) or (True, <python-list>) or (False, <error-message>)
```

Example *run_model()* or prediction artifact

```
In [16]: s.download_artifact('5da17d64be9c4441899316edb9afd403')
Out[16]:
(True,     Diagnosis  prob_BENIGN  prob_MALIGNANT
 0      BENIGN     0.999400    6.002134e-04
 1      BENIGN     1.000000    3.600000e-09
 2      BENIGN     0.999999    8.689000e-07
 3      BENIGN     1.000000    2.500000e-09
 4   MALIGNANT     0.004159    9.958413e-01
 5   MALIGNANT     0.002674    9.973264e-01
 ..        ...          ...             ...

92  MALIGNANT     0.002499    9.975013e-01
93      BENIGN     1.000000    5.250000e-08
94      BENIGN     1.000000    3.100000e-08
95      BENIGN     0.999901    9.866350e-05
96      BENIGN     1.000000    9.230000e-08
97  MALIGNANT     0.003884    9.961160e-01
98  MALIGNANT     0.002777    9.972232e-01
99  MALIGNANT     0.003686    9.963139e-01

[100 rows x 3 columns])
```

Example *analyze_data()* artifact

```
In [19]: s.download_artifact('923338b7512f4770b239e1b53406cfa6')
Out[19]:
(True,                         col_name     col_type   drop is_cat      max  \
 0                           Code        int64    True  False  8233704
 1                Clump Thickness  categorical  False   True     None
 2        Uniformity of Cell Size  categorical  False   True     None
 3       Uniformity of Cell Shape  categorical  False   True     None
 4              Marginal Adhesion  categorical  False   True     None
 5    Single Epithelial Cell Size  categorical  False   True     None
 6                    Bare Nuclei     LongType  False  False       10
 7                Bland Chromatin  categorical  False   True     None
 8                Normal Nucleoli  categorical  False   True     None
 9                        Mitoses  categorical  False   True     None
 10                     Diagnosis  categorical  False   True     None

                        mean                   min  missing num_uniques  \
 0        1044171.0667779633                 61634      0.0         559
 1         4.555926544240401                  None      0.0          10
 2        3.2153589315525877                  None      0.0          10
 3         3.287145242070117                  None      0.0          10
 4        2.8597662771285477                  None      0.0          10
 5         3.290484140233723                  None      0.0          10
 6   -2.30969249670820768E17  -9223372036854775808      0.0          11
 7        3.5208681135225377                  None      0.0          10
 8          2.96661101836394                  None      0.0          10
 9        1.6076794657762938                  None      0.0           9
 10                     None                  None      0.0           2
```

```
    scalable                   stddev                          uniques
 0       True       414096.3687689267                             None
 1      False       2.887487844960718  [9, 1, 5, 2, 6, 3, 10, 7, 4, 8]
 2      False       3.044601202894244  [9, 1, 5, 2, 6, 3, 10, 7, 4, 8]
 3      False      2.9710450562657416  [9, 1, 5, 2, 6, 3, 10, 7, 4, 8]
 4      False       2.873655092520189  [9, 1, 5, 2, 6, 3, 10, 7, 4, 8]
 5      False      2.2751587689827613  [9, 1, 5, 2, 6, 3, 10, 7, 4, 8]
 6       True  1.44237363952833229E18                             None
 7      False       2.369500020847775  [9, 1, 5, 2, 6, 3, 10, 7, 4, 8]
 8      False      3.0844664820475916  [9, 1, 5, 2, 6, 3, 10, 7, 4, 8]
 9      False      1.7343686380557295     [1, 5, 2, 6, 3, 10, 7, 4, 8]
 10     False                    None          ['BENIGN', 'MALIGNANT']  )
```

Example *analyze_model()* or prediction artifact

```
  In [5]: s.download_artifact('6e4861de29424cb7ad09e467d1869c17',\
   'path_to_download_dir/')
  Out[5]:
  True RM            0.216088
CRIM          0.141956
LSTAT         0.134069
DIS           0.104101
PTRATIO       0.089905
AGE           0.078864
NOX           0.074132
B             0.067823
TAX           0.045741
INDUS         0.023659
ZN            0.011041
RAD = 4.0     0.009464
RAD = 5.0     0.001577
RAD = 6.0     0.001577
RAD = 24.0    0.000000
RAD = 3.0     0.000000
RAD = 7.0     0.000000
CHAS = 1.0    0.000000
RAD = 8.0     0.000000
RAD = 2.0     0.000000
dtype: float64
```

Example *analyze_predictions()* artifact

```
In [8]: (code, fis) = s.download_artifact('34b461c7a52a48318e982068f87e6562',\
 'path_to_download_dir/')

In [9]: fis.head()
Out[9]:    ##Sample return for regression, has predicted_value column
```

```
        AGE         B  CHAS = 1.0      CRIM       DIS     INDUS     LSTAT  \
0  0.000000  0.000000    0.000000 -0.664664 -0.923219 -0.720941  2.328635
1 -1.220243 -0.648893    0.000000  0.000000  1.187539 -0.630767  3.506132
2 -0.456561 -0.226880   -0.424802  0.000000 -0.077616 -0.333270 -0.292705
3 -0.195096  0.352712    0.000000 -1.867664 -0.152037  0.273082 -3.583178
4  0.632119  0.079678    0.000000  0.076080 -0.488128 -0.016690 -0.102031

        NOX   PTRATIO  RAD = 2.0       ...         RAD = 4.0  RAD = 5.0  \
0 -0.342404  0.224360        0.0       ...         -0.641678  -0.570788
1 -0.556636 -2.168356        0.0       ...          0.000000  -0.741561
2  0.000000  1.458677        0.0       ...          0.000000  -0.340486
3 -0.945060 -1.068743        0.0       ...          0.000000   0.217991
4  0.309544  0.298940        0.0       ...          0.000000  -0.047708

   RAD = 6.0  RAD = 7.0  RAD = 8.0        RM       TAX        ZN  base_value  \
0        0.0        0.0        0.0 -1.835851 -0.563795 -0.600155    21.63455
1        0.0        0.0        0.0 -1.016655 -0.699813 -0.727181    21.63455
2        0.0        0.0        0.0 -1.137559  0.000000 -0.310209    21.63455
3        0.0        0.0        0.0 -1.220045  0.156790  0.256763    21.63455
4        0.0        0.0        0.0 -0.999328 -0.149627 -0.045493    21.63455

   predicted_value
0        24.620939
1        26.128595
2        24.200972
3        11.255393
4        21.982929

[5 rows x 22 columns]

Out[9]: ##Sample return for classification, returns predicted_class as well

   petal length (cm)  petal width (cm)  sepal length (cm)  sepal width (cm)  \
0           0.217699          0.424209           0.026237          0.005834
1           0.292612          0.315358           0.019236         -0.014442
2           0.325615          0.329229           0.003208          0.016954
3           0.232265          0.410938           0.043014          0.004154
4           0.317190          0.339065           0.015227          0.003523

   base_value  predicted_value predicted_class
0    0.309628         0.983607       virginica
1    0.365378         0.978142      versicolor
2    0.324994         1.000000          setosa
3    0.309628         1.000000       virginica
4    0.324994         1.000000          setosa
```

DarwinSdk.delete_artifact(*artifact_name*)

Delete the artifact given its name.

Parameters:

- *artifact_name* - Name of the artifact to be deleted.

Returns:

```
(True, None)  or (False, <error-message>)
```

Example

```
In [8]: s.delete_artifact('6c482eac9f894cdb9b0e1e487e41730a')
Out[8]:
(True, None)
```

Data Analysis and Data Cleaning methods

DarwinSdk.analyze_data(*dataset_name, **kwargs*)

Analyze the dataset given its *name*. Basic statistics about the data are returned. This method supports Big Data (greater than 2GB) although *upload_dataset()* is artificially limited to 10GB for version 1.4.

Note: Please contact us if you have data greater than 10GB. We would like to see a sampling of the large datasets that you’d like to see supported.

Parameters:

dataset_name - The name of the dataset to be analyzed.

***kwargs* - variable number of keyword arguments, described below:

- *job_name* - (optional) If not specified, a uuid will be created as the job_name.
- *artifact_name*: (optional) If not specified, a uuid will be created as the *artifact_name*.
- *max_unique_values*: Expected input/type: *integer*. Default value of 15. 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.

Returns:

```
(True, {"job_name”: <string>, "artifact_name”: <string>})  or (False, <error-message>)
```

Statistics included in the artifact:

- *col_name* - name of the column (any periods (’.’) in the column name will be replaced by underscores (’_’))
- *col_type* - type of column

- *drop* - returns True if column is dropped for modeling. Also returns True if the number of unique values is greater than the number defined in *max_unique_values* (default of 15) or if it has more than 80% missing values or has a standard deviation of 0.
- *is_cat* - returns True for categorical otherwise returns False
- *max* - column maximum
- *mean* - column mean
- *min* - column minimum
- *missing* - percentage of missing values
- *num_uniques* - number of unique values if the distinct count is less than the number defined in *max_unique_values* (default of 15), otherwise the value is the approximate number of unique values.
- *scalable* - returns True if column is scalable
- *stddev* - column standard deviation
- *uniques* - actual unique values if there are less than the number defined in *max_unique_values* (default of 15). Otherwise, nothing is returned, see *num_uniques* for the approximate number of unique values.

Example

```
In [6]: s.analyze_data('boston')
Out[6]:
(True,
 {'artifact_name': 'db968d77d2c4444ab731777d01e5e0c0',
  'job_name': '8c12f0df4c39485f9a488fa63196e00c'})

In [8]: s.download_artifact('db968d77d2c4444ab731777d01e5e0c0')
Out[8]:
(True,
                        col_name     col_type   drop is_cat          max   \
 0                           PID   StringType   True  False  2205663001_
 1                        ST_NUM   StringType   True  False          999
 2                       ST_NAME   StringType   True  False       ZELLER
 3                   ST_NAME_SUF   StringType   True  False           XT
 4                       ZIPCODE   StringType   True  False       02467_
 5                Assessed_Value        int64   True  False     23095700
 6                      Lot_Area        int64   True  False       107158
 7                    Gross_Area        int64   True  False        23335
 8                   Living_Area        int64   True  False        21711
 9                Owner_Occupied  categorical  False   True         None
 10                   Year_Built        int64   True  False         2016
 11             Number_of_Floors      float64  False  False          5.0
 12        Total_Number_of_Rooms        int64   True  False           27
 13           Number_of_Bedrooms  categorical  False   True         None
 14         Number_of_Full_Baths  categorical  False   True         None
 15         Number_of_Half_Baths  categorical  False   True         None
 16           Number_of_Kitchens  categorical  False   True         None
 17                       Has_AC  categorical  False   True         None
 18         Number_of_Fireplaces  categorical  False   True         None
```

```
19  Year_Since_Remodel_or_Build        int64   True  False          307
20               Year_Remodeled   StringType   True  False  Unremodeled
21               Structure_Type  categorical  False   True         None
22               Building_Style   StringType   True  False    Victorian
23                    Roof_Type  categorical  False   True         None
24              Exterior_Finish  categorical  False   True         None
25          Main_Bathroom_Style  categorical  False   True         None
26           Main_Kitchen_Style  categorical  False   True         None
27                 Heating_type  categorical  False   True         None
28           Exterior_Condition  categorical  False   True         None
29            Overall_Condition  categorical  False   True         None
30           Interior_Condition  categorical  False   True         None
31              Interior_Finish  categorical  False   True         None
32                         View  categorical  False   True         None

                   mean          min   missing num_uniques scalable  \
0                  None  0100021000_  0.000000       28578     True
1    122.09705524787249        1005R  0.010223        1922     True
2                  None   ABBOTSFORD  0.000000        2246     True
3                  None           ST  0.003015          21     True
4                  None       02108_  0.000000          28     True
5     534716.6815977456       101300  0.000000        7737     True
6     5116.273150271971          375  0.000000        8342     True
7    2931.1126220591127          510  0.000000        4472     True
8    1752.7717084999017          332  0.000000        3169     True
9    0.8408480241169146         None  0.000000           2    False
10    1926.970935185792         1710  0.000000         225     True
11   1.8748115866046269          1.0  0.000000           9     True
12    7.233632610262796            2  0.000000          26     True
13   3.3851169801428664         None  0.000000          12    False
14   1.4273543482534898         None  0.000000          10    False
15   0.5716953928828888         None  0.000000           7    False
16   1.0287043711907726         None  0.000000           4    False
17  0.18733206632151517         None  0.000000           2    False
18    0.590995478078511         None  0.000000          13    False
19    60.88419948882627            1  0.000000         190     True
20   2000.3376960831488         1890  0.000000          82     True
21                 None         None  0.000000           5    False
22                 None     Bi-Level  0.000000          17     True
23                 None         None  0.000000           7    False
24                 None         None  0.000000          13    False
25                 None         None  0.000000           4    False
26                 None         None  0.000000           4    False
27                 None         None  0.000000           6    False
28                 None         None  0.000000           5    False
```

```
29                 None         None   0.000000            5    False
30                 None         None   0.000000            5    False
31                 None         None   0.000000            3    False
32                 None         None   0.000000            5    False

                 stddev                                             uniques
0                  None                                                None
1     294.1511958893473                                                None
2                  None                                                None
3                  None                                                None
4                  None                                                None
5     634750.7826113638                                                None
6     3218.286557124007                                                None
7    1069.3847598444354                                                None
8     758.9874732061347                                                None
9    0.3658237412175791                                              [0, 1]
10     34.9170355483078                                                None
11   0.5737101635770085                                                None
12   1.8082562295656077                                                None
13   1.0095185504254367             [12, 9, 1, 5, 2, 6, 3, 10, 7, 4, 11, 8]
14   0.6850264359951297                     [12, 9, 1, 5, 2, 6, 3, 7, 4, 8]
15   0.5645602408681473                               [0, 1, 5, 2, 6, 3, 4]
16  0.17162236936210065                                        [0, 1, 2, 3]
17   0.3901842537872663                                              [0, 1]
18   0.8584446055814273          [0, 12, 9, 1, 5, 2, 6, 3, 10, 7, 4, 11, 8]
19   43.323487380439225                                                None
20   13.578956800881818                                                None
21                 None  ['Residential', 'Wood/Frame', 'Unknown', 'Bric...
22                 None                                                None
23                 None  ['Shed', 'Gambrel', 'Flat', 'Other', 'Mansard'...
24                 None  ['Cement Board', 'Frame/Clapboard', 'Wood Shak...
25                 None  ['Semi-Modern', 'Luxury', 'No Remodeling', 'Mo...
26                 None  ['Semi-Modern', 'Luxury', 'No Remodeling', 'Mo...
27                 None  ['Electric', 'Other', 'None', 'Hot Water', 'Sp...
28                 None   ['Poor', 'Good', 'Excellent', 'Average', 'Fair']
29                 None   ['Poor', 'Good', 'Excellent', 'Average', 'Fair']
30                 None   ['Poor', 'Good', 'Excellent', 'Average', 'Fair']
31                 None              ['Elaborate', 'Normal', 'Substandard']
32                 None   ['Poor', 'Good', 'Excellent', 'Average', 'Fair'] )
```

DarwinSdk.clean_data(*dataset_name, **kwargs*)

Clean the dataset given its name. 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 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.

Parameters:

- *dataset_name* - The name of the dataset to be analyzed.
- ***kwargs* - variable number of keyword arguments, described below:
  - *job_name* - (optional) If not specified, a uuid will be created as the job_name.
  - *artifact_name*: (optional) If not specified, a uuid will be created as the *artifact_name*.
  - *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.
  - *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

  - *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.

Example

```
In [6]: s.clean_data('cancer', artifact_name='cancer-cleandata')
Out[6]:
    (True,
     {'artifact_name': 'cancer-cleandata',
      'job_name': 'baa07cf9a7734c8da2afcf805b2d0571'})

In [8]: s.download_dataset('cancer-cleandata', \
    artifact_path='/Users/username/Downloads/artifacts')
Out[8]:
  (True,
   {'filename': \
    '/Users/username/Downloads/artifacts/cancer-cleandata3-cleaned-8m38g07j.csv'})
```

Modeling and analysis methods

DarwinSdk.create_model(*dataset_names*, ***kwargs*)

Create a model trained on the dataset identified by dataset_names. You must clean the data using *clean_data()* The name of a model is specified in a parameter in kwargs.
Note: If no name is specified, the model is named with a *uuid-like* name.

Parameters:

dataset_names - A single dataset name as a string or a list of dataset string names to be used for training. The maximum file size is 500 MB for unsupervised and NBM and 10 GB for supervised.

***kwargs* - variable number of keyword arguments, described in *parameters*.

parameters -

- *model_name*: The string identifier of the model to be trained. If no name is specified, the model is named with a *uuid-like* name.
- *job_name*: If no name is specified, the job is named with a *uuid-like* 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): 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 can 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

- *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**.
- *clustering* (unsupervised only): Enables clustering for unsupervised problems. If False, detects outliers.
- *n_clusters* (unsupervised only): Expected input/type: *integer*. Specifies the number of clusters. **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.
- *loss_fn_name*: 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.
- *fitness_fn_name*: Specify the fitness function. This represents the name of the fitness function used for evolution of the model population during training. Possible values include: *"Accuracy"*, *"F1"*, *"R2"*, *"MSE"*. "F1" is the default for classification and "R2" is the default for regression problems. "Accuracy" and "F1" are for classification only. "R2" and "MSE" are for regression only.
- *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"`

Returns:

```
(True, {'job_id': <uuid1>, model_name: <model_name>}) or (False, <error-message>)
```

Example

```
In [10]: s.create_model('cancer-data', model_name="cancer-\
model", max_train_time="00:01", max_epochs=0)
Out[10]:
(True,
 {'job_id': 'f5124576a4f34e5c9ab3499770455509',
  'model_name': 'cancer-model'})
```

DarwinSdk.delete_model(*model_name*)

Delete a model named by *model_name*.

Parameters:

- *model_name* - Name of the model to be deleted.

Returns:

```
(True, None) or (False, <error-message>)
```

Example

```
In [5]: s.delete_model('unittest-cancer-model')
Out[5]: (True, None)
```

DarwinSdk.resume_training_model(*model_name, dataset_names, **kwargs*)

Resume training for a model on the dataset(s) identified by *dataset_names*.

Parameters:

- *model_name* - Name of the model to be trained.
- *dataset_name*- Name of dataset(s) used for training.
- ***kwargs* - variable number of keyword arguments, described below:.
  - *job_name* - If not specified, a uuid is created as the *job_name*.
  - *max_train_time* - If not specified, the *default* is used.

Returns:

```
(True, {"job_id""<uuid>", "model_name": "<model_name>"}) or (False, <error-message>)
```

Example

```
In [8]: s.resume_training_model('unittest-cancer-model', 'unittest-cancer-\
dataset', max_train_time="00:01")
Out[8]:
(True, {"job_id": "4e59ffc425e047e1a3b872f1e7396976", "model_name": "unittest-\
cancer-model"})
```

DarwinSdk.analyze_model(*model_name, job_name*=*None, artifact_name*=*None*)

Analyze the universal feature importances for a particular model given the model name.

Parameters:

- *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 importance for. If this is not specified, the feature importance will be over all classes/clusters.
- *model_type*: (optional) Model type from the population. Possible values include: *DeepNeuralNetwork, RandomForest, GradientBoosted.*

Returns:

```
(True, {"job_name”: <string>, "artifact_name”: <string>})  or (False, <error-message>)
```

Example

```
In [5]: s.analyze_model('unittest-cancer-model')
Out [5]:
(True, {'artifact_id': '71a8ae55f2934014b45c13a3975f419c', 'job_id': \
'4e59ffc425e047e1a3b872f1e7396976'})
```

DarwinSdk.analyze_predictions(*model_name, dataset_name, job_name*=*None, artifact_name*=*None*)

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.

Parameters:

- *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*.
- *start_index* - (optional) Index to start at in the dataset when analyzing model predictions.
- *end_index* - (optional) Index to stop at in the dataset when analyzing model predictions.
- *model_type*: (optional) Model type from the population. Possible values include: *DeepNeuralNetwork, RandomForest, GradientBoosted.*

Returns:

```
(True, {"job_name”: <string>, "artifact_name”: <string>}) or (False, <error-message>)
```

Example

```
In [5]: s.analyze_predictions('model_name', 'dataset_name')
Out [5]:
(True, {'artifact_name': '71a8ae55f2934014b45c13a3975f419c', 'job_name': \
'4e59ffc425e047e1a3b872f1e7396976'})
```

DarwinSdk.run_model(*dataset_name, model_name, job_name*=*None, artifact_name*=*None*)

Run the model given its name and a dataset to use. Use **upload_dataset**() to upload a data set.

Parameters:

- *dataset_name* - The name of a dataset to use for running the model.
- *model_name* - The name of the model to run.
- *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.) (optional) A boolean (True/False) indicating whether the model is supervised or not, for example, set this to *False* for *unsupervised.*
- *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*.
- *model_type* - (optional) Model type of the model to be downloaded. Possible values include: *DeepNeuralNetwork, RandomForest, GradientBoosted.*

Returns:

```
(True, {"job_name”: <string>, "artifact_name”: <string>}) or (False, <error-message>)
```

Example

```
[In [9]: s.run_model('unittest-cancer-testdataset', 'unittest-cancer-model')
Out [9]:
(True, {'artifact_id': '6c482eac9f894cdb9b0e1e487e41730a', 'job_id': \
'1696e03c8165404c8e05685ea68baa3c'})
```

Convenience methods

DarwinSdk.delete_all_datasets()

Deletes user datasets. This method deletes all datasets in the current user or service context.
Note: Use *lookup_dataset()* to view/verify the datasets for deletion.

Parameters: None

Returns:

```
(True, None) or (False, <error-message>)
```

DarwinSdk.delete_all_models()

Delete all models for a user. This method will delete all models in the current user's or service's context.
Note: Use *lookup_model()* to review and verify that you want to delete all listed models.

Parameters: None

Returns:

```
(True, None) or (False, <error-message>)
```

DarwinSdk.delete_all_artifacts()

Delete all artifacts for a user. This method will delete all artifacts in the current user's or service's context.
Note: Use *lookup_artifact()* to review and verify that you want to delete all listed artifacts.

Parameters: None

Returns:

```
(True, None) or (False, <error-message>)
```

DarwinSdk.wait_for_job*(job_name, time_limit=600)*

Synchronously wait for a job to complete, limited by *time_limit* that defaults to 600 seconds.

Parameters:

- *job_name* - The id for the job
- *time_limit* - (optional) defaults to 600 seconds

Returns:

```
(True, None) or (False, <error-message>)
```

Reference

- SDK modeling example
- Revision table

SDK modeling example

The following example shows the Darwin SDK performing a modeling process:

```
  ---
 In [1]: from amb_sdk.sdk import DarwinSdk

In [2]: s = DarwinSdk()

In [3]: s.auth_login_user('your-username', 'your-password')
Out[3]:
(True,
 'Bearer eyJ0eXAiOiJK...A8sj4pAzX1FpMMscwY_rMJbnGo0YQ_4')

In [4]: s.upload_dataset('sets/cancer_train.csv', 'mydata')
Out[4]: (True, {'dataset_name': 'mydata'})

In [5]: s.clean_data('mydata', target='Diagnosis')
Out[5]:
(True,
 {'job_name': '801ee7e95dfd4380b7be76332ead5036',
  'artifact_name': '97fad4a3598f41068eadd84df26a6eaa'})

In [6]: s.wait_for_job('801ee7e95dfd4380b7be76332ead5036')
{'status': 'Complete', 'starttime': '2019-01-16T11:28:09.779535',\
 'endtime': '2019-01-16T11:28:12.613227', 'percent_complete': 100,\
  'job_type': 'CleanDataTiny', 'loss': None, 'generations': None,\
   'dataset_names': ['mydata'], 'artifact_names': \
   ['97fad4a3598f41068eadd84df26a6eaa'], 'model_name': None,\
    'job_error': ''}
Out[6]: (True, 'Job completed')

In [7]: s.create_model(dataset_names='mydata', model_name='my-model')
Out[7]:
(True,
 {'job_name': '2bbf5dc050b6499a9e19e0c6173a2821',
  'job_id': '2fa8953e-19b4-11e9-a52a-1b252aa286fd',
  'model_name': 'my-model'})

In [8]: s.wait_for_job('2bbf5dc050b6499a9e19e0c6173a2821')
{'status': 'Running', 'starttime': '2019-01-16T11:28:49.588621',\
 'endtime': None, 'percent_complete': 0, 'job_type': 'TrainModel',\
  'loss': 0.4303114712238312, 'generations': 2, 'dataset_names': ['mydata'],\
   'artifact_names': None, 'model_name': 'my-model', 'job_error': ''}
{'status': 'Running', 'starttime': '2019-01-16T11:28:49.588621',
 'endtime': None, 'percent_complete': 3, 'job_type': 'TrainModel',\
  'loss': 0.18398252129554749, 'generations': 2, 'dataset_names': ['mydata'],\
   'artifact_names': None, 'model_name': 'my-model', 'job_error': ''}
{'status': 'Running', 'starttime': '2019-01-16T11:28:49.588621',\
 'endtime': None, 'percent_complete': 8, 'job_type': 'TrainModel',\
  'loss': 0.41190358996391296, 'generations': 4, 'dataset_names': ['mydata'],\
   'artifact_names': None, 'model_name': 'my-model', 'job_error': ''}
{'status': 'Complete', 'starttime': '2019-01-16T11:28:49.588621',\
 'endtime': '2019-01-16T11:29:55.502275', 'percent_complete': 100,\
  'job_type': 'TrainModel', 'loss': 0.41190358996391296, 'generations': 4,\
   'dataset_names': ['mydata'], 'artifact_names': None,\
    'model_name': 'my-model', 'job_error': ''}
Out[8]: (True, 'Job completed')

In [9]: s.upload_dataset('sets/cancer_test.csv', 'mytestdata')
Out[9]: (True, {'dataset_name': 'mytestdata'})

In [10]: s.clean_data('mytestdata', model_name='my-model')
Out[10]:
(True,
 {'job_name': '625f2cc5c6e2437d808b158ad66bfefc',
  'artifact_name': '17bad7c5426c4166afa4fe70eb0ff8a1'})

In [11]: s.wait_for_job('625f2cc5c6e2437d808b158ad66bfefc')
{'status': 'Complete', 'starttime': '2019-01-16T11:34:11.567819',\
 'endtime': '2019-01-16T11:34:13.283641', 'percent_complete': 100,\
  'job_type': 'CleanDataTiny', 'loss': None, 'generations': None,\
   'dataset_names': ['mytestdata'], 'artifact_names': ['17bad726...4fe70eb0ff8a1'],\
    'model_name': None, 'job_error': ''}
Out[11]: (True, 'Job completed')

In [12]: s.run_model('mytestdata', 'my-model')
Out[12]:
(True,
 {'job_name': '75d8bf61689346fda84b430f5fe1be58',
  'artifact_name': '20e87cda3ef24cd18f065ccaf87e8ca4'})

In [13]: s.wait_for_job('75d8bf61689346fda84b430f5fe1be58')
{'status': 'Complete', 'starttime': '2019-01-16T11:35:33.1109',\
 'endtime': '2019-01-16T11:35:34.891138', 'percent_complete': 100,\
  'job_type': 'RunModel', 'loss': 0.41190358996391296, 'generations': 4,\
   'dataset_names': ['mytestdata'], 'artifact_names': ['20e87cda3...065ccaf8ca4'],\
    'model_name': 'my-model', 'job_error': ''}
Out[13]: (True, 'Job completed')

In [14]: s.download_artifact('20e87cda3ef24cd18f065ccaf87e8ca4')
Out[14]:
(True,     Diagnosis  prob_BENIGN  prob_MALIGNANT
 0      BENIGN     0.665922        0.334078
 1      BENIGN     0.676795        0.323205
 2      BENIGN     0.676795        0.323205
 3      BENIGN     0.676795        0.323205
 4   MALIGNANT     0.107175        0.892825
 5   MALIGNANT     0.049802        0.950199
 6   MALIGNANT     0.002107        0.997893
 7      BENIGN     0.676795        0.323205
 ...
 95     BENIGN     0.676795        0.323205
 96     BENIGN     0.676795        0.323205
 97  MALIGNANT     0.050580        0.949420
 98  MALIGNANT     0.032286        0.967714
 99  MALIGNANT     0.052880        0.947120

 [100 rows x 3 columns])
```

Revision Table

Version	Date	Notes
v 1.0	05-Feb-2018	Initial Release
v 1.2	28-Mar-2018	Added: • DarwinSdk.auth_change_password • DarwinSdk.delete_job • DarwinSdk.stop_job • DarwinSdk.lookup_user • DarwinSdk.lookup_username • DarwinSdk.auth_delete_user Name change: lookup_client to lookup_limits

Version	Date	Notes
v 1.3	23-May-2018	Added: • DarwinSdk.auth_reset_password • DarwinSdk.auth_set_email • DarwinSdk.analyze_predictions Updated endpoints: • DarwinSdk.auth_register_user • DarwinSdk.analyze_model
v 1.3.1	14-Jun-2018	Fixed issues only. See Release Notes.
v 1.4	31-Jul-2018	• Island Models implemented to allow model types to reproduce at their own speeds • User selectable loss functions • Output model confidence value • Specify download paths for artifacts • Parameter validation • Stored data is encrypted • **DarwinSdk.lookup_model()** and **DarwinSdk.lookup_model_name(*model_name*)** calls display model description
v 1.5	15-Oct-2018	New endpoints: • DarwinSdk.clean_data • DarwinSdk.download_dataset • DarwinSdk.download_model Updated endpoints: • DarwinSdk.analyze_data • DarwinSdk.download_artifact • DarwinSdk.create_model • DarwinSdk.lookup_model
v 1.6	16-Jan-2019	New endpoints: • DarwinSdk.display_population • DarwinSdk.delete_all_artifacts Updated endpoints: • DarwinSdk.analyze_data • DarwinSdk.download_artifact • DarwinSdk.create_model • DarwinSdk.clean_data