Darwin ™ Python SDK Guide

A SparkCognition™ Education Document

v. 1.46.0 - January 2020

SparkCognition Darwin Python SDK Guide

Contents

About this guide **3**

Expectation 3

Darwin overview **3**

Accessing the API 4

Darwin SDK interface **4**

Setup Darwin SDK 4

Set up Users 5

Set up Admin account 5

Set up User accounts 5

Darwin SDK methods **7**

URL Get/Set methods 7

DarwinSdk.**get_info**() 7

DarwinSdk.**get_sdk_version**() 7

DarwinSdk.**get_url**() 8

DarwinSdk.**set_url**(*url, version='v1'*) 8

Authentication methods 9

DarwinSdk.**auth_register**(*password, api_key, email*) 9

DarwinSdk.**auth_login**(*password, api_key*) 9

DarwinSdk.**auth_register_user**(username, *password, email*) 10

DarwinSdk.**auth_set_email**(*username, email*) 10

DarwinSdk.**auth_login_user**(*username, password*) 11

DarwinSdk.**auth_change_password**(*curpass, newpass*) 11

DarwinSdk.**auth_reset_password**(*username*) 12

DarwinSdk.**auth_delete_user**(*username*) 12

DarwinSdk.**disable_ssl_cert_check**() 13

DarwinSdk.**enable_ssl_cert_check**() 13

Job status methods 14

DarwinSdk.**lookup_job_status***(age=None, status=None)* 14

DarwinSdk.**lookup_job_status_name**(*job_name*) 14

DarwinSdk.**delete_job**(*job_name*) 15

DarwinSdk.**stop_job**(*job_name*) 16

Lookup methods 16

DarwinSdk.**lookup_artifact**(*type=None*) . . . 16
DarwinSdk.**lookup_artifact_name**(*artifact_name*) . . . 17
DarwinSdk.**lookup_limits**() . . . 17
DarwinSdk.**lookup_dataset**() . . . 18
DarwinSdk.**lookup_dataset_name**(*dataset_name*) . . . 19
DarwinSdk.**lookup_model**() . . . 19
DarwinSdk.**lookup_model_name**(*model_name*) . . . 20
DarwinSdk.**lookup_tier**() . . . 20
DarwinSdk.**lookup_tier_num**(*tier_num*) . . . 21
DarwinSdk.**lookup_user**() . . . 21
DarwinSdk.**lookup_username**(*username*) . . . 22
DarwinSdk.**display_population**(*model_name*) . . . 23
Datasets and artifact methods . . . 24
DarwinSdk.**upload_dataset**(*dataset, dataset_name=None*) . . . 24
DarwinSdk.**download_dataset**(*dataset_name*) . . . 24
DarwinSdk.**delete_dataset**(*dataset_name*) . . . 25
DarwinSdk.**download_model**(*model_name*) . . . 25
DarwinSdk.**download_artifact**(*artifact_name, artifact_path=None*) . . . 26
DarwinSdk.**delete_artifact**(*artifact_name*) . . . 31
Data Analysis and Data Cleaning methods . . . 31
DarwinSdk.**analyze_data**(*dataset_name, **kwargs*) . . . 31
DarwinSdk.**clean_data**(*dataset_name, **kwargs*) . . . 35
Modeling and analysis methods . . . 36
DarwinSdk.**create_model**(*dataset_names, **kwargs*) . . . 36
DarwinSdk.**delete_model**(*model_name*) . . . 40
DarwinSdk.**resume_training_model**(*model_name, dataset_names, **kwargs*) . . . 40
DarwinSdk.**analyze_model**(*model_name, job_name=None, artifact_name=None*) . . . 40
DarwinSdk.**analyze_predictions**(*model_name, dataset_name, job_name=None, artifact_name=None*) . . . 41
DarwinSdk.**run_model**(*dataset_name, model_name, job_name=None, artifact_name=None*) . 42
DarwinSdk.**align_forecasting_predictions**(*model_name, data, predictions*) . . . 43
Convenience methods . . . 43
DarwinSdk.**delete_all_datasets**() . . . 43
DarwinSdk.**delete_all_models**() . . . 44
DarwinSdk.**delete_all_artifacts**() . . . 44
DarwinSdk.**wait_for_job***(job_name, time_limit=600)* . . . 44
DarwinSdk.**help**() . . . 44

Reference **46**
SDK modeling examples . . . 46
Supervised modeling example . . . 46
Unsupervised modeling example . . . 52
NBM modeling example . . . 55
Forecasting modeling example . . . 63
Contact Support . . . 67
Revision Table . . . 67

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 (**...**).

The Darwin SDK has an independent version number to allow for release outside of the normal Darwin product release window. As of this printing, the Darwin SDK is at version 1.46.0.

The documentation for this version of Darwin includes:

- The *Darwin Release Notes*, version 2.0.5
- The *Darwin User Interface Guide*, version 2.0.5
- The *Darwin API User Guide*, version 1.36.1
- The *Darwin Python SDK User Guide*, version 1.46.0
- The *Darwin RTE User Guide*, version 3.0.0

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

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
- Analyze training data
- Clean training data
- Create model
- Wait for job to complete
- Upload test data
- Clean test data
- Run the model
- Wait for job to complete
- Download the 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 (no more than 4000) and data samples that span rows. Plan your data file so it fits this expectation to help prevent errors.

See the SDK examples for modeling examples of supervised, unsupervised, normal behavior modeling (NBM), and forecasting problems.

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 set up the Darwin SDK, unless you have already set up your service and created users using the API.
  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 darwin-sdk
   ```

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/

   Note: Ensure you have a trailing slash (/) on the production URL.

   ON-PREM ONLY: For on-prem installations, the product URL will be in the form:

   ```
   https://customerdomainname.customerdomain.com:8000/v1/
   ```

   Note: On-prem installation must add port 8000 to the product URL.

8. Verify the connection.

   Use `get_url()` and `set_url()` to verify connection to the correct Darwin service. See the URL Get/Set methods below for more information.

Set up Users

Before you can set up any user accounts, you need to know your api key, also known as an admin key. This key can be obtained from SparkCognition support or your IT manager. The api key is a long string, for example:

```
'RsJ74ZS5AvwznbHh0AfVSgrchhS9KxACDy3jefaQnxb9f6QTSDBFmhnGa0cOIWtNAIFRAG9ToOTpi0mn'
```

Set up Admin account

Register the api key using the `auth_register()` method.

The purpose of this method is to create a password and an email address for the Darwin admin account. This method must be invoked once for each api key to establish an admin account for that key.

Example

```
>>> from amb_sdk.sdk import DarwinSdk
>>> s = DarwinSdk()
>>> s.auth_register('adminpassword', 'iRgwut4kGs0ymULiuKtMd0WFvBYLMWSj16q2uysQeteq\
UvcysnPojRpfycLVHa2IlN1IlrfEk1YMA', 'admin@company.com')
(True,'Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE1MTU1MzM4NjEsImlh\
dCI6MTUxNTUzMDI2MS ... F56xZQiBT-89nrRz1nIXD5LfawHIj_MlUHQqM36vU')
```

Set up User accounts

While you can use the SDK as an admin, it is more convenient to create additional user accounts so that you can have certain datasets/models be owned by specific users. Perform the following to create additional user accounts:

Log in to the *service* as an admin. In the following example, you need to enter your admin password and the api key.

Example

```
>>> s.auth_login('adminpassword', 'iRgwut4kGs0ymULiuKtMd0WFvBYLMWSj16q2uysQeteq\
UvcysnPojRpfycLVHa2IlN1IlrfEk1YMA')
(True,'Bearer iLCJhbGciOeyJ0eXAiOiJKV1QiJIUzI1NiJ9.eyJleHAiOjE1MTU1MzQxNzIsImlh\
dUxNTUCI6MTzMD ... UQQfoXqYFKJSoRXXDNPE985-a08cE6_o')
```

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 2 hours, if the DarwinSdk object session life time exceeds 2 hours, the SDK will request another auth token until the session ends.

Register a new user by calling the `auth_register_user()` method. You need to input the username, password, and email address for the new user.

Example

```
>>> s.auth_register_user('user1', 'user1-password', 'user1@company.com')
(True,
 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiJkNjY0MmJjOC1iMmU5LTQxO\
DctODFlNS00YjI2MD ... 5zMp_1FfxU')
```

You can repeat this procedure for additional users.

The user can now log in by using the `auth_login_user()` method. The user needs to input the username and password.

Example

```
    >>> s.auth_login_user('user1', 'user1-password')
    (True,
     'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiI3NGYzYmUxZS0yOTlmLTRhN\
    zMtODU5ZC01NGRmM2F ... u1zGCeCONA')
```

The user is now logged in and can perform other functions. See the following sections for other SDK methods.

Darwin SDK methods

URL Get/Set methods

DarwinSdk.get_info()

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

Parameters: None

Returns:

```
(True, {available_routes: {}, local: False})
```

Example

```
In [29]: s.get_info()

Out[29]: (True,
{'available_routes': {'Info': True,
'Auth': True,
'Job': True,
'Metadata': True,
'Train': True,
'Risk': True,
'Upload': True,
'Download': True,
'Analyze': True,
'Run': True,
'Admin': True,
'Clean': True,
'Model': True,
'Dataset': True,
'Population': True},
'local': False, 'api_version': '1.36.1'})
```

DarwinSdk.get_sdk_version()

Get the version of the SDK.

Parameters: None

Returns:

```
(True, '1.46.0')
```

Example

```
In [8]: s.get_sdk_version()

Out[8]: (True, '1.46.0')
```

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 for SaaS

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

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

Example for On-prem

```
In [9]: s.set_url('https://customerdomainname.customerdomain.com:8000/v1/')

Out[9]: (True, 'https://customerdomainname.customerdomain.com:8000/v1/')
```

Authentication methods

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

Register the api key, also known as an admin key, as a service and establish an admin account. The purpose of this method is to set a password and an email address for the Darwin Admin account. This method is invoked only once for each api key to establish a password and Admin account. After registration, the admin can log in to the service using the *auth_login*() method.

Parameters:

- *password* - The service level password for the admin
- *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 2 hours, if the DarwinSdk object session life time exceeds 2 hours, the SDK will request another auth token until the session ends.

Example

```
In [4]: s.auth_register('adminpassword', 'RsJ74ZS5AvwznbHh0AfVSgrchhS9KxACDy\
3jefaQnxb9f6QTSDBFmhnGa0cOIWtNAIFRAG9ToOTpi0mnEo3zFA', 'admin@company.com')

Out[4]:
(True,
 'Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiO...iSdU8xlF4yJk')
```

DarwinSdk.auth_login(*password, api_key*)

Log in to the *service* as an admin.
Note: A service must have a password set using `auth_register()` to login successfully.

Parameters:

- *password* - The service level password for the admin
- *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 2 hours, if the DarwinSdk object session life time exceeds the 2 hour limit, the SDK will acquire another auth token until the session ends.

Example

```
In [5]: s.auth_login('adminpassword',
'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 2 hours, if the DarwinSdk object session life time exceeds the 2 hour limit, the SDK will acquire another auth token until the session ends.

Example

```
In [8]: s.auth_register_user('user1', 'user1-password', 'user1@company.com')

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. For cloud installations, this email address will be used for password resets and other notifications. For on-prem installations, password resets are executed using a new default password, there is no email sent. If you do not know the default password, contact Darwin support.

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 2 hours, if the DarwinSdk object session life time exceeds the 2 hour limit, the SDK will acquire another auth token until the session ends.

Example

```
In [9]: s.auth_login_user('user1', 'user1-password')


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. For cloud installations, an email will be generated with a temporary password. For on-prem installations, password resets are executed using a new default password, there is no email sent. If you do not know the default password, contact Darwin support.

Example

```
In [10]: s.auth_change_password('user1-password', 'user1-newpassword')


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

DarwinSdk.auth_reset_password(*username*)

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

Parameter:

- *username* - Username to reset password for.

Returns:

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

Example

```
In [8]: s.auth_reset_password('user1')


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

DarwinSdk.auth_delete_user(*username*)

Remove/Unregister a user. This can only be performed by an admin account.

Parameter:

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

Returns:

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

Example

```
In [8]: s.lookup_username('testuser2')

Out[8]:
(True,
 [{'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrch...Eo3zFA',
   'created_at': '2020-01-03T12:54:30.653478',
   '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')
```

DarwinSdk.disable_ssl_cert_check()

For on-prem customers, this disables the SSL certificate check when calling `auth_login_user(<username>, <password>`

If you want to re-enable the SSL certificate check, call the `enable_ssl_cert_check()` method.

Parameters: None

DarwinSdk.enable_ssl_cert_check()

For on-prem customers, this enables the SSL certificate check when calling `auth_login_user(<username>, <password>`

If you want to disable the SSL certificate check, call the `disable_ssl_cert_check()` method.

Parameters: None

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': '2020-02-01T10:53:50.451598',
   'generations': 0,
   'job_name': 'eeef500d629e4a2185eb8af6e18a83b4',
   'job_type': 'TrainModel',
   'loss': 2.0,
   'model_name': 'cancer-model',
   'percent_complete': 100,
   'starttime': '2020-02-01T10:52:42.280929',
   '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': '2020-02-01T10:52:42.280929',
  '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': '2020-03-06T14:23:59.975793',
  'generations': None,
  'job_error': '',
  'job_type': 'AnalyzeData',
  'loss': None,
  'model_name': None,
  'percent_complete': 100,
  'starttime': '2020-03-06T14:23:57.18095',
  '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 running job. The job will not stop right away, but it will stop when the current generation is complete.

Parameter:

- *job_name* - The name of the job you want to stop.

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': '2020-02-01T11:09:55.731040',
   'id': 'b9a9205a-0772-11e8-a003-3b1c8766dad0',
   'mbytes': 0.0,
   'name': '8a63e21030d1483abb0f892963c1728f',
   'type': 'Run'},
  {'created_at': '2020-02-01T11:11:17.560360',
   '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_name* - specify 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': '2020-02-01T11:11:17.560360',
  '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,
 [{ 'id': {},
    'categorical': None,
   'imbalanced': None,
   'mbytes': 0.02019977569580078,
   'minimum_recommended_train_time': "string"
   'name': 'unittest-cancer-dataset2',
   'sequential': None,
   'updated_at': '2020-01-31T15:37:28.310994'},
  {'id': {},
   'categorical': None,
   'imbalanced': None,
   'mbytes': 0.02019977569580078,
   'minimum_recommended_train_time': "string"
   'name': 'cancer-train',
   'sequential': None,
   'updated_at': '2020-02-01T10:52:06.076279'}])
```

DarwinSdk.lookup_dataset_name(*dataset_name*)

Get a specific dataset's metadata.

Parameters:

- *dataset_name* - The dataset name. The dataset name 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': '2020-02-01T10:52:06.076279'})
```

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': '2020-02-01T10:53:50.443166',
   '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': '2020-02-01T10:53:50.443166',
  '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': '2020-01-03T12:54:30.653478',
   'parent_id': '2516b462-df85-11e7-bdd1-e37424c63ea4',
   'tier': 0,
   'username': 'testuser2'},
  {'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrchhS9...3zFA',
   'created_at': '2020-01-03T13:14:36.188371',
   'parent_id': '2516b462-df85-11e7-bdd1-e37424c63ea4',
   'tier': 0,
   'username': 'testuser5'},
  {'client_api_key': 'RsJ74ZS5AvwznbHh0AfVSgrchhS9...3zFA',
   'created_at': '2020-01-03T13:21:21.099148',
   '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': '2020-01-03T12:54:30.653478',
   '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 due to only supporting uploading data via http.

Note: Datasets uploaded via the SDK cannot be used in the user interface. If you plan on using the UI of Darwin, you will need to upload the dataset using the UI functionality.

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.
- *artifact_path* - (optional) Path to the download directory for the artifact

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 the following: *DeepNeuralNetwork, RandomForest, GradientBoosted.*

- *model_format* - (optional) Format in which the model is to be downloaded. Possible values include: *json*, *onnx*. The ONNX format is only available for neural network models.

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 [97]: s.download_artifact('1a38f1af934c4cbabb9136ee94f72718')

Out[97]:
(True,
                                count   drop  is_date low_samples max  \
name
Code                            599  False    False      []  8233704.0
 Clump Thickness                599  False    False      []       10.0
 Uniformity of Cell Size        599  False    False      []       10.0
 Uniformity of Cell Shape       599  False    False      []       10.0
 Marginal Adhesion              599  False    False      []       10.0
 Single Epithelial Cell Size    599  False    False      []       10.0
 Bare Nuclei                    599  False    False      []       10.0
 Bland Chromatin                599  False    False      []       10.0
 Normal Nucleoli                599  False    False      []       10.0
 Mitoses                        599  False    False      []       10.0
Diagnosis                       599  False    False      []        NaN


mean           min           missing  num_uniques        stddev  \
1.044171e+06  6.163400e+04      0.0           557     4.140964e+05
```

```
4.555927e+00  1.000000e+00      0.0           10     2.887488e+00
3.215359e+00  1.000000e+00      0.0           10     3.044601e+00
3.287145e+00  1.000000e+00      0.0           10     2.971045e+00
2.859766e+00  1.000000e+00      0.0           10     2.873655e+00
3.290484e+00  1.000000e+00      0.0           10     2.275159e+00
-2.309692e+17 -9.223372e+18     0.0           11     1.442374e+18
3.520868e+00  1.000000e+00      0.0           10     2.369500e+00
2.966611e+00  1.000000e+00      0.0           10     3.084466e+00
1.607679e+00  1.000000e+00      0.0            9     1.734369e+00
NaN           NaN               0.0            2           NaN

top_unique_values                   treatment    type   type_note \
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
    []                               numeric     int64
[[BENIGN, 379], [MALIGNANT, 220]]  categorical  object

uniques
[61634, 63375, 76389, 95719, 128059, 142932, 1...
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[-9223372036854775808, 1, 2, 3, 4, 5, 6, 7, 8,...
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
   [1, 2, 3, 4, 5, 6, 7, 8, 10]
            [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
```

Example *run_model()* forecasting artifact

The forecasting artifact returns the future values based on the forecast_horizon value. In the following case, the forecast_horizon value was 3.

The first row of the output is the `seq_len` + 1 row.

```
In [33]:
s.download_artifact('6ae83c0d59c9411a8f3f596a33057834')
Out[33]:
(True,          future_1      future_2      future_3
 0    11306.876953  10307.818359   9128.345703
 1    11661.523438  10513.156250   9195.193359
 2     8693.410156   7886.320801   7096.866211
 3     6402.957031   5710.714844   5166.185547
 4     4489.059570   4121.243164   3995.698242
 5     2979.267578   3142.290039   3468.699219

 ..            ...           ...           ...

 958  26081.808594  26612.792969  26809.789062
 959  26810.261719  26978.746094  27026.882812
 960  27688.652344  27256.580078  27052.052734
 961  26898.109375  26352.546875  26050.978516
```

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.

Note: *upload_dataset()* is currently artificially limited to 10GB due to only supporting uploading data via http. 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.

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

Parameters:

dataset_name - (required) 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*.
- *target*: String denoting target prediction column in input data.
- *char_encoding*: The character encoding of the dataset. The default value is `utf-8`. If your dataset has a different encoding, set the value with this parameter. For a list of possible values, click here.

- *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 and a target. When you run *clean_data()* before running a model, you must specify a dataset_name and a model_name. *clean_data()* can also be used for visualizing what Darwin would do with the dataset or for when you want to use the cleaned data outside of Darwin.

Parameters:

- *dataset_name* - (required) 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.
  - *index*: String denoting the date/time column name to use as an index.
  - *impute*: String alias that indicates how to fill in missing values in input data.

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

ALIAS	DESCRIPTION	COMPLEXITY
'median'	Median Fill: Computes the median value of all non-missing examples in a column to fill in missing examples. While the result is interpretable in terms of the input space for categorical variables, the approach might not be appropriate for non-ordinal data.	Linear Fast
'Linear'	Linear Interpolation Fill: Interpolation using a Linear function. Useful for timeseries or sequential data.	Linear Fast

Example

```
In [16]: s.clean_data('pacman-cancer', target='Diagnosis')
Out[16]:
    (True,
        {'job_name': '77b47b391b0d483699fe0741cc52e6ad',
        'job_id': '72756658-a63b-11e9-824c-c75a5cadd55f',
        'profile_name': '60f9429a0a2141d9ad98c3d073b8425f',
        'profile_id': '7276d1d2-a63b-11e9-824c-6be79b2f3967'})
```

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 - (required) 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.

Ensure that the dataset used supports the task that Darwin is to solve. For example, if you want to forecast future results, make sure that your data is in a consistent time-series format with regular intervals between data points. The data doesn't have to necessarily have time stamps, but in this case, it is assumed that each row has a regular time interval between them.

fit_profile_name: (required) This is the *profile_name* that is generated from the *clean_data* call.

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

parameters -

- *val_size*: Portion of the dataset to be used as a validation set during training, expressed as a decimal that is greater than 0 and less than 1. Default value is 0.2 (i.e., 20%).
- *cv_kfold*: k-fold cross-validation, where k is the number of groups that a given data sample is to be split into for training/validation. Default is 1 for non-timeseries data or 3 for timeseries data. Maximum value allowed is 10. This parameter is not currently supported for forecasting model creation.

- *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*: 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 can result in slower model evolution. If you want to see the LSTM and TCN models used during training or if you want to treat your problem as a time series problem, you must set `recurrent=True`.
- *anomaly*: Setting this parameter to **True** indicates that an isolation forest should be built for anomaly detection. If set to **True**, clustering will automatically be interpreted as **False**.
- *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.
- *forecast_horizon* (forecasting only): Integer indicating how long in the future you want to forecast predictions. For example, if you have 6 months of time-series data and each row represents a 1 day interval and you want to predict the next week of data, you should set `forecast_horizon=7`. It is not recommended to have a `forecast_horizon` value greater than 20.

  Note: For best results, be sure that the minimum time (in minutes) to train the model is 10 times the value specified in *forecast_horizon* or 30 minutes, whichever is longer.

  Note: Ensure that your training data is 5 times greater than the `forecast_horizon`, otherwise an error will be generated.
- *anomaly_prior* (unsupervised only): Expected input/type: *between [0,1]*. Significance level at which a point is defined as anomalous. This is only used for unsupervised problems if *clustering* is disabled.
- *class_weights*: A string to indicate how relatively important each class is for predictive correctness. This is done by providing a numeric value to each class. Note that the class name is case-sensitive. The following is an example *class_weights* setting:

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

  The reason that you'd want different class weights would be to account for the fact that the reward/cost for classifying a certain class differs from the other(s). For example, the "cost" of misclassifying a malignant tumor is much higher than misclassifying a benign tumor.
- *loss_fn_name*: Specify the loss function. Possible values include: *"CrossEntropy", "MSE", "BCE", "L1", "NLL", "BCEWithLogits", "SmoothL1".*

  "CrossEntropy", *"BCE"*, and *"BCEWithLogits"* can be used for classification data, while all others can be used for regression data. The default value is *CrossEntropy* if this field is left empty.

- *fitness_fn_name*: Specify the fitness function. This represents the name of the fitness function used for evolution of the model population during training.

  For classification problems, possible values include:

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

  For regression problems, possible values include:

  - `r2` - (R^2) (default) Measures how closely the data maps to the fitted regression line. It is also known as the coefficient of determination and is useful for mapping the relationships that exist in data.

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

- *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 [18]: s.create_model('pacman-cancer', \
  fit_profile_name='60f9429a0a2141d9ad98c3d073b8425f', max_train_time='00:02')
Out[18]:
  (True,
    {'job_name': 'b29d680547f94d1e87bf0e6ae6913ae0',
    'job_id': '81a68bde-a63b-11e9-aa27-2ba9c50e485f',
    'model_name': '1f871bd9b0b3405680a2bce0c3b2b226'})
```

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 [39]: s.resume_training_model('1f871bd9b0b3405680a2bce0c3b2b226', \
    'pacman-cancer', max_train_time='00:01')
Out[39]:
    (True,
        {'job_name': '30ba83c61814459f95b668a8316e647f',
         'job_id': '66e3ef92-a63d-11e9-9792-a38b3282194d',
         'model_name': '1f871bd9b0b3405680a2bce0c3b2b226'})
```

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

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

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

Parameters:

- *model_name* - (required) The name of the model to be analyzed.
- *job_name* - (optional) If not specified, a uuid is created as the *job_name*.
- *artifact_name* - (optional) If not specified, a uuid is created as the *artifact_name*.
- *category_name* - (optional) The name of the class for supervised or cluster for unsupervised to get feature 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 100 MB.

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

Parameters:

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

- *model_type*: (optional) Model type from the population. Possible values include: *DeepNeuralNetwork*, *RandomForest*, *GradientBoosted*.

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) User can specify a model type to use for their prediction. If nothing is defined, the SDK will use the best model type. Possible values include:
  - `DeepNeuralNetwork`: The run_model command will pick the best performing neural network to use when running the prediction.
  - `RandomForest`: The run_model command will pick the best performing sklearn random forest to use when running the prediction.
  - `GradientBoosted`: The run_model command will pick the best performing sklearn gradient boosted model to use when running the prediction.

Returns:

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

Example

```
In [21]: s.run_model('pacman-cancer', '1f871bd9b0b3405680a2bce0c3b2b226')
Out[21]:
    (True,
     {'job_name': 'dfd758d43bda429cb19b4d9460db689d',
      'job_id': '612b7b2a-a63c-11e9-acd6-7b9b48091c5f',
      'artifact_name': '2f8502b4e3494ba3b5ce2133e066ec1a'})
```

DarwinSdk.align_forecasting_predictions(*model_name, data, predictions*)

Align data and predictions for a forecasting model. Useful for computing performance metrics and plotting predictions vs actual data. This method will lookup several parameters of the forecasting model and reshape the prediction columns named `future_*` in a way that they align with the raw input data.

Parameters:

- *model_name* - The name of the forecasting model.
- *data* - pandas.DataFrame containing data from a Darwin dataset.
- *predictions* - pandas.DataFrame containing forecasting predictions returned from DarwinSdk.**run_model**(dataset, ...)

Returns:

Tuple of (*aligned_data, aligned_preds*) where

- *aligned_data* - pandas.DataFrame containing the data aligned to the forecasting model's predictions.
- *aligned_preds* - pandas.DataFrame containing the forecasting model's predictions corresponding to each row of *aligned_data*. The length will be the same as *aligned_data*.

You can now evaluate the forecasting model using any scikit-learn metric as in the following example, assuming your target column is a variable named `target`.

Example

```
from sklearn.metrics import r2_score
print(r2_score(aligned_preds[target], aligned_preds))
```

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. If the *time_limit* is reached, your job will continue to run but **wait_for_job** will discontinue monitoring it. You can re-run **wait_for_job** or modify the *time_limit* parameter.

Parameters:

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

Returns:

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

DarwinSdk.help()

Shows all the methods available.

Parameters: None

Example

```
In [5]: s.help()
Out [5]:
analyze_data (self, dataset_name, **kwargs)
analyze_model (self, model_name, job_name=None, artifact_name=None, \
  category_name=None, model_type=None)
analyze_predictions (self, model_name, dataset_name, job_name=None, \
  artifact_name=None, model_type=None)
auth_change_password (self, curpass, newpass)
auth_delete_user (self, username)
auth_login (self, password, api_key)
auth_login_user (self, username, password)
auth_register (self, password, api_key, email)
auth_register_user (self, username, password, email)
auth_reset_password (self, username)
auth_set_email (self, username, email)
clean_data (self, dataset_name, **kwargs)
create_model (self, dataset_names, **kwargs)
delete_all_artifacts (self)
delete_all_datasets (self)
delete_all_models (self)
delete_artifact (self, artifact_name)
delete_dataset (self, dataset_name)
delete_job (self, job_name)
delete_model (self, model_name)
disable_ssl_cert_check (self)
display_population (self, model_name)
download_artifact (self, artifact_name, artifact_path=None)
download_dataset (self, dataset_name, file_part=None, artifact_path=None)
download_model (self, model_name, path=None, model_type=None, model_format=None)
enable_ssl_cert_check (self)
get_info (self)
get_url (self)
lookup_artifact (self, type=None)
lookup_artifact_name (self, artifact_name)
lookup_dataset (self)
lookup_dataset_name (self, dataset_name)
lookup_job_status (self, age=None, status=None)
lookup_job_status_name (self, job_name)
lookup_limits (self)
lookup_model (self)
lookup_model_name (self, model_name)
lookup_tier (self)
lookup_tier_num (self, tier_num)
lookup_user (self)
lookup_username (self, username)
```

```
resume_training_model (self, model_name, dataset_names, **kwargs)
run_model (self, dataset_name, model_name, **kwargs)
set_url (self, url, version='v1')
stop_job (self, job_name)
upload_dataset (self, dataset_path, dataset_name=None, has_header=True)
wait_for_job (self, job_name, time_limit=600)
```

Reference

- SDK modeling example
- Revision table

SDK modeling examples

The following section details modeling examples for the following types of problems:

- Supervised
- Unsupervised
- NBM
- Forecasting

Supervised modeling example

```
---
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 [14]: s.upload_dataset('sets/cancer_train.csv', 'pacman-cancer')
Out[14]: (True, {'dataset_name': 'pacman-cancer'})

In [15]:  s.analyze_data('pacman-cancer')
Out[15]:
(True,
{'job_name': '3b3a54324a68427583ccae1194822fdd',
 'job_id': '688d5aba-a63b-11e9-b969-d3fc6b14b182',
 'artifact_name': '5b38f0d797cd45c5a7081a0c1b02ccad'})

In [16]: s.clean_data('pacman-cancer', target='Diagnosis')
```

```
Out[16]:
(True,
{'job_name': '77b47b391b0d483699fe0741cc52e6ad',
 'job_id': '72756658-a63b-11e9-824c-c75a5cadd55f',
 'profile_name': '60f9429a0a2141d9ad98c3d073b8425f',
 'profile_id': '7276d1d2-a63b-11e9-824c-6be79b2f3967'})

In [18]: s.create_model('pacman-cancer', \
 fit_profile_name='60f9429a0a2141d9ad98c3d073b8425f', max_train_time='00:02')
Out[18]:
(True,
{'job_name': 'b29d680547f94d1e87bf0e6ae6913ae0',
 'job_id': '81a68bde-a63b-11e9-aa27-2ba9c50e485f',
 'model_name': '1f871bd9b0b3405680a2bce0c3b2b226'})

In [19]: s.wait_for_job('b29d680547f94d1e87bf0e6ae6913ae0')
{'status': 'Running', 'starttime': '2019-07-14T08:30:11.995459', \
 'endtime': None, 'percent_complete': 0, 'job_type': 'TrainModel', \
 'loss': None, 'generations': 0, 'dataset_names': ['pacman-cancer'], \
 'artifact_names': None, 'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', \
 'job_error': ''}
{'status': 'Running', 'starttime': '2019-07-14T08:30:11.995459', \
'endtime': None, 'percent_complete': 50, 'job_type': 'TrainModel', \
'loss': 0.20505395531654358, 'generations': 1, 'dataset_names': ['pacman-cancer'], \
'artifact_names': None, 'model_name': '1f871bd9b0b3405680a2bce0c3b2b226',\
 'job_error': ''}
...
'generations': 29, 'dataset_names': ['pacman-cancer'], 'artifact_names': None, \
'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', 'job_error': ''}
{'status': 'Complete', 'starttime': '2019-07-14T08:30:11.995459', \
'endtime': '2019-07-14T08:34:40.40471', 'percent_complete': 100, \
'job_type': 'TrainModel', 'loss': 0.20505395531654358, 'generations': 31, \
'dataset_names': ['pacman-cancer'], 'artifact_names': None, \
'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', 'job_error': ''}
Out[19]: (True, 'Job completed')

In [21]: s.run_model('pacman-cancer', '1f871bd9b0b3405680a2bce0c3b2b226')
Out[21]:
(True,
{'job_name': 'dfd758d43bda429cb19b4d9460db689d',
 'job_id': '612b7b2a-a63c-11e9-acd6-7b9b48091c5f',
 'artifact_name': '2f8502b4e3494ba3b5ce2133e066ec1a'})

In [22]: s.wait_for_job('dfd758d43bda429cb19b4d9460db689d')
{'status': 'Complete', 'starttime': '2019-07-14T08:36:26.997233', \
 'endtime': '2019-07-14T08:36:40.418987', 'percent_complete': 100, \
 'job_type': 'RunModel', 'loss': 0.20505395531654358, 'generations': 31, \
 'dataset_names': ['pacman-cancer'], \
 'artifact_names': ['2f8502b4e3494ba3b5ce2133e066ec1a'], \
 'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', 'job_error': ''}
Out[22]: (True, 'Job completed')

In [23]: s.download_artifact('2f8502b4e3494ba3b5ce2133e066ec1a')
Out[23]:
(True,      Diagnosis  prob_BENIGN  prob_MALIGNANT
     0       BENIGN     0.896724        0.103276
     1    MALIGNANT     0.313960        0.686040
    ...
     597     BENIGN     0.886773        0.113227
     598     BENIGN     0.931960        0.068040

    [599 rows x 3 columns])

In [26]: s.upload_dataset('sets/cancer_test.csv', 'pacman-cancertest')
Out[26]: (True, {'dataset_name': 'pacman-cancertest'})

In [27]: s.clean_data('pacman-cancertest', \
model_name='1f871bd9b0b3405680a2bce0c3b2b226')
Out[27]:
(True,
{'job_name': '462ade77431d4fde92b780ddc00573d9',
 'job_id': 'a868dd5c-a63c-11e9-94d1-8b9399879e43',
 'artifact_name': 'd8d4e61a4c624e70abb8bf66fde42e45',
 'artifact_id': 'a86a8a30-a63c-11e9-94d1-93c25fc085f5'})

In [28]: s.run_model('pacman-cancertest', \
'1f871bd9b0b3405680a2bce0c3b2b226')
Out[28]:
(True,
    {'job_name': '015bc7fa826c4f36b5ca2b4e8b27dba0',
     'job_id': 'b3eff6c4-a63c-11e9-bfd4-2387a41a7f36',
     'artifact_name': 'f5e5de3ac8bc413385d94ff9203ed919'})

In [29]: s.wait_for_job('015bc7fa826c4f36b5ca2b4e8b27dba0')
{'status': 'Complete', 'starttime': '2019-07-15T12:53:34.483154', \
'endtime': '2019-07-15T12:53:42.266985', 'percent_complete': 100, \
'job_type': 'RunModel', 'loss': 0.4324471354484558, 'generations': 21, \
'dataset_names': ['pacman-cancer'], \
'artifact_names': ['f5e5de3ac8bc413385d94ff9203ed919'], \
'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', 'job_error': ''}
```

```
Out[29]: (True, 'Job completed')

In [30]: s.download_artifact('f5e5de3ac8bc413385d94ff9203ed919')
Out[30]:
(True,      Diagnosis  prob_BENIGN  prob_MALIGNANT
     0       BENIGN     0.816393        0.183607
     1       BENIGN     0.947398        0.052602
     2       BENIGN     0.947646        0.052354
     3       BENIGN     0.947398        0.052602
     4    MALIGNANT     0.189687        0.810313
     5    MALIGNANT     0.256924        0.743076
    ...
     98   MALIGNANT     0.225788        0.774212
     99   MALIGNANT     0.202293        0.797707
     100     BENIGN     0.816393        0.183607
     101     BENIGN     0.816393        0.183607

    [102 rows x 3 columns])


In [31]: s.analyze_model('1f871bd9b0b3405680a2bce0c3b2b226')
Out[31]:
    (True,
     {'job_name': '173bc091175c41c3a03d2cef9b4344fb',
      'job_id': 'eef624aa-a63c-11e9-94d1-2bccd93a7796',
      'artifact_name': '4d4684c1956844df8b412119637e890b'})

In [32]: s.wait_for_job('173bc091175c41c3a03d2cef9b4344fb')
{'status': 'Complete', 'starttime': '2019-07-14T08:40:24.885663', \
'endtime': '2019-07-14T08:40:26.360908', 'percent_complete': 100, \
'job_type': 'AnalyzeModel', 'loss': 0.20505395531654358, 'generations': 31, \
'dataset_names': None, 'artifact_names': ['4d4684c1956844df8b412119637e890b'], \
'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', 'job_error': ''}
Out[32]: (True, 'Job completed')

In [33]: s.download_artifact('4d4684c1956844df8b412119637e890b')
Out[33]:
(True, Diagnosis = BENIGN            0.143460
      Single Epithelial Cell Size    0.141226
      Bland Chromatin                0.114085
      Normal Nucleoli                0.112659
      Mitoses                        0.100054
      Marginal Adhesion              0.091280
      Uniformity of Cell Size        0.070170
      Uniformity of Cell Shape       0.066997
     Code                            0.058587
      Clump Thickness                0.053780
     Diagnosis = MALIGNANT           0.046033
      Bare Nuclei                    0.001669
     dtype: float64)

In [34]:  s.analyze_predictions('1f871bd9b0b3405680a2bce0c3b2b226', \
         'pacman-cancertest')
Out[34]:
    (True,
     {'job_name': 'b8e93f64e00f41819886cccbd2cad488',
      'job_id': 'f9f081a0-a729-11e9-8035-c3c7e048165a',
      'artifact_name': 'cbb36947ba5e4511846a80668207c77c'})

In [35]: s.wait_for_job('b8e93f64e00f41819886cccbd2cad488')
{'status': 'Running', 'starttime': '2019-07-15T12:57:14.029058', \
'endtime': None, 'percent_complete': 0, 'job_type': 'AnalyzePredictions', \
'loss': 0.4324471354484558, 'generations': 21, 'dataset_names': None, \
'artifact_names': ['cbb36947ba5e4511846a80668207c77c'], \
'model_name': '9f3ffa24162448158bdb1c0f3fe8c21e', 'job_error': ''}
{'status': 'Running', 'starttime': '2019-07-15T12:57:14.029058', \
'endtime': None, 'percent_complete': 0, 'job_type': 'AnalyzePredictions', \
'loss': 0.4324471354484558, 'generations': 21, 'dataset_names': None, \
'artifact_names': ['cbb36947ba5e4511846a80668207c77c'], \
'model_name': '9f3ffa24162448158bdb1c0f3fe8c21e', 'job_error': ''}
{'status': 'Complete', 'starttime': '2019-07-15T12:57:14.029058', \
'endtime': '2019-07-15T13:05:26.715496', 'percent_complete': 100, \
'job_type': 'AnalyzePredictions', 'loss': 0.4324471354484558, 'generations': 21, \
'dataset_names': None, 'artifact_names': ['cbb36947ba5e4511846a80668207c77c'], \
'model_name': '9f3ffa24162448158bdb1c0f3fe8c21e', 'job_error': ''}
Out[35]: (True, 'Job completed')

In [36]: s.download_artifact('cbb36947ba5e4511846a80668207c77c')
Out[36]:
(True,
Code_shap  Clump Thickness_shap   Uniformity of Cell Size_shap   Uniformity of ... \
     0     0.001568               0.029792                       0.068253
     1     0.000836               0.039068                       0.052454
     2     0.000584               0.039056                       0.052201
     3     0.002181               0.037539                       0.052797
     4    -0.001198               0.012781                       0.124094

    ...
     98   -0.001878               0.000495                       0.131678
     99   -0.000813               0.000495                       0.141285
     100   0.001568               0.029792                          0.068253
     101   0.001568               0.029792                          0.068253

     [102 rows x 13 columns])

In [39]: s.resume_training_model('1f871bd9b0b3405680a2bce0c3b2b226', \
'pacman-cancer', max_train_time='00:01')
Out[39]:
    (True,
     {'job_name': '30ba83c61814459f95b668a8316e647f',
      'job_id': '66e3ef92-a63d-11e9-9792-a38b3282194d',
      'model_name': '1f871bd9b0b3405680a2bce0c3b2b226'})

In [40]: s.wait_for_job('30ba83c61814459f95b668a8316e647f')
{'status': 'Running', 'starttime': '2019-07-14T08:43:46.090825', 'endtime': None, \
'percent_complete': 0, 'job_type': 'UpdateModel', 'loss': 0.20505395531654358, \
'generations': 31, 'dataset_names': ['pacman-cancer'], 'artifact_names': None, \
'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', 'job_error': ''}

... ...

{'status': 'Complete', 'starttime': '2019-07-14T08:43:46.090825', 'endtime': \
'2019-07-14T08:46:07.389587', 'percent_complete': 100, 'job_type': 'UpdateModel', \
'loss': 0.20505395531654358, 'generations': 52, 'dataset_names': ['pacman-cancer'], \
'artifact_names': None, 'model_name': '1f871bd9b0b3405680a2bce0c3b2b226', \
'job_error': ''}
Out[40]: (True, 'Job completed')

In [41]: s.run_model('pacman-cancer', '1f871bd9b0b3405680a2bce0c3b2b226')
Out[41]:
    (True,
     {'job_name': 'cc623240c5b1453eb73cdcdc93777f55',
      'job_id': 'c4380d18-a63d-11e9-85df-2337b425c294',
      'artifact_name': '48d74e4277944412ac032879fc23c5ba'})



In [45]: s.download_artifact('48d74e4277944412ac032879fc23c5ba')
Out[45]:
(True,      Diagnosis  prob_BENIGN  prob_MALIGNANT
     0        BENIGN     0.920420        0.079580
     1     MALIGNANT     0.461758        0.538242
    '''
     597      BENIGN     0.834559        0.165441
     598      BENIGN     0.949016        0.050984
```

```
   [599 rows x 3 columns])
```

Unsupervised modeling example

```
 ---
 In [47]: s.upload_dataset('sets/pulsars.csv', 'pacman-pulsars')
 Out[47]: (True, {'dataset_name': 'pacman-pulsars'})

 In [48]: s.analyze_data('pacman-pulsars')
 Out[48]:
 (True,
  {'job_name': '294ed354f3484b2ebebe658033284128',
   'job_id': '80193b9c-a63e-11e9-85df-fba3623db3bc',
   'artifact_name': 'b3d8dacf77fc409b9dd48030b2dda07b'})

 In [49]: s.clean_data('pacman-pulsars')
 Out[49]:
 (True,
  {'job_name': '454f781b2a02403ea74d915a4b6b530c',
   'job_id': '8763baf8-a63e-11e9-900b-4335c8f0f324',
   'artifact_name': 'b9820a205bf140b1ae24bc6b1b133d2d',
   'artifact_id': '876560d8-a63e-11e9-900b-7f0b539a80a9'})

 In [50]: s.create_model('pacman-pulsars',\
  fit_profile_name='b9820a205bf140b1ae24bc6b1b133d2d')
 Out[50]:
 (True,
  {'job_name': '2a9a1e55f8e34ddd828326ecee2b42f4',
   'job_id': '97a92592-a63e-11e9-a08f-cf415e11311a',
   'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265'})

 In [51]: s.wait_for_job('2a9a1e55f8e34ddd828326ecee2b42f4')
 {'status': 'Running', 'starttime': '2019-07-14T08:52:17.412985', 'endtime': None,\
  'percent_complete': 0, 'job_type': 'TrainModel', 'loss': None, 'generations': 0, \
  'dataset_names': ['pacman-pulsars'], 'artifact_names': None, \
  'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}
 {'status': 'Running', 'starttime': '2019-07-14T08:52:17.412985', 'endtime': None,\
  'percent_complete': 0, 'job_type': 'TrainModel', 'loss': None, 'generations': 0,\
   'dataset_names': ['pacman-pulsars'], 'artifact_names': None,\
    'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}

    ...
```

```
{'status': 'Complete', 'starttime': '2019-07-14T08:52:17.412985', \
'endtime': '2019-07-14T08:53:53.736499', 'percent_complete': 100, \
'job_type': 'TrainModel', 'loss': None, 'generations': 0, \
 'dataset_names': ['pacman-pulsars'], 'artifact_names': None, \
 'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}
Out[51]: (True, 'Job completed')

In [53]: s.run_model('pacman-pulsars', 'fa1b82cab28c46cdac3b44c8e8bc1265')
Out[53]:
(True,
 {'job_name': 'c72ef79400014a3fb9e0a5821adf0826',
  'job_id': '0b58fb52-a63f-11e9-9f7b-53a24e4e1e2a',
  'artifact_name': '05380df9e93c4650ab6f7e5e67e23f72'})

In [54]: s.wait_for_job('c72ef79400014a3fb9e0a5821adf0826')
{'status': 'Complete', 'starttime': '2019-07-14T08:55:31.501449', \
'endtime': '2019-07-14T08:55:34.679054', 'percent_complete': 100, \
'job_type': 'RunModel', 'loss': None, 'generations': 0, \
'dataset_names': ['pacman-pulsars'],\
'artifact_names': ['05380df9e93c4650ab6f7e5e67e23f72'], \
'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}
Out[54]: (True, 'Job completed')

In [55]: s.download_artifact('05380df9e93c4650ab6f7e5e67e23f72')
Out[55]:
(True,
 {'filename': '/var/folders/wc/w7ktf3392_s7c6t2djlhyb9/T/artifact-6ac2m9yp.csv'})

n [17]: s.analyze_model('b7d28ee423d4430fafa51a017be827ac')
Out[17]:
(True,
 {'job_name': 'b36800c1a47f4c559c231d155a12fd85',
  'job_id': '8c5dfd70-a8f5-11e9-88ca-173db83c1239',
  'artifact_name': 'a417260dfabc409a8204fa1ceeae112f'})

In [18]: s.wait_for_job('b36800c1a47f4c559c231d155a12fd85')
{'status': 'Complete', 'starttime': '2019-07-17T19:46:58.691115',\
 'endtime': '2019-07-17T19:47:02.152927', 'percent_complete': 100,\
  'job_type': 'AnalyzeModel', 'loss': None, 'generations': 0,\
   'dataset_names': None, 'artifact_names': ['a417260dfabc409a8204fa1ceeae112f'],\
    'model_name': 'b7d28ee423d4430fafa51a017be827ac', 'job_error': ''}
Out[18]: (True, 'Job completed')

In [19]: s.download_artifact('a417260dfabc409a8204fa1ceeae112f')
Out[19]:
```

```
(True, kurt_dmsnr      0.0
 skew_dmsnr      0.0
 skew_profile    0.0
 kurt_profile    0.0
 class           0.0
 std_profile     0.0
 mean_dmsnr      0.0
 mean_profile    0.0
 std_dmsnr       0.0
 dtype: float64)

In [22]: s.upload_dataset('sets/pulsars_predict.csv', 'pulsars-test')\
  # Need to trim original dataset to have fewer than 500 rows.
Out[22]: (True, {'dataset_name': 'pulsars-test'})

In [23]: s.clean_data('pulsars-test', model_name='b7d28ee423d4430fafa51a017be827ac')
Out[23]:
(True,
 {'job_name': '8324da8dea734455a73daeeddd3e0b5f',
  'job_id': 'fc2a4974-a8f5-11e9-9074-7fa762e40db7',
  'profile_name': 'bccde471e8514ef59b0b106fa7af6be9',
  'profile_id': 'fc2c0e76-a8f5-11e9-9074-13d39d00d68d'})

In [24]: s.analyze_predictions ('b7d28ee423d4430fafa51a017be827ac', 'pulsars-test')
Out[24]:
(True,
 {'job_name': 'def26ef5be3a4d5b822542fd125c8600',
  'job_id': '12995024-a8f6-11e9-89b8-cf7654542d20',
  'artifact_name': 'd726a45761a1431d8bbe381c4f4f2782'})

In [25]: s.wait_for_job('def26ef5be3a4d5b822542fd125c8600')
{'status': 'Complete', 'starttime': '2019-07-17T19:50:43.895873',\
 'endtime': '2019-07-17T19:50:49.098101', 'percent_complete': 100,\
  'job_type': 'AnalyzePredictions', 'loss': None, 'generations': 0,\
   'dataset_names': None, 'artifact_names': ['d726a45761a1431d8bbe381c4f4f2782'],\
    'model_name': 'b7d28ee423d4430fafa51a017be827ac', 'job_error': ''}
Out[25]: (True, 'Job completed')
In [26]: s.download_artifact('d726a45761a1431d8bbe381c4f4f2782')
Out[26]:
(True,
    std_dmsnr_shap  mean_dmsnr_shap  kurt_dmsnr_shap  mean_profile_shap  ... \
 0             0.0              0.0              0.0                0.0
 1             0.0              0.0              0.0                0.0
 2             0.0              0.0              0.0                0.0
 3             0.0              0.0              0.0                0.0
 4              0.0               0.0               0.0                0.0
 5              0.0               0.0               0.0                0.0
 6              0.0               0.0               0.0                0.0
 7              0.0               0.0               0.0                0.0
 8              0.0               0.0               0.0                0.0
 [9 rows x 12 columns])
```

NBM modeling example

```
---
In [13]: s.upload_dataset('sets/SmokyT025_full_raw.csv', 'pacman-smoky')
Out[13]: (True, {'dataset_name': 'pacman-smoky'})

In [14]: s.analyze_data('pacman-smoky')
Out[14]:
(True,
 {'job_name': 'af7d01c94e774b3aa5648bf675cf990f',
  'job_id': 'fc4e1f30-a895-11e9-be40-8b36979e9d84',
  'artifact_name': 'c267aa0daa43407fbb59f44c3644d2b8'})

In [15]: s.clean_data('pacman-smoky', index='timestamp')
Out[15]:
(True,
 {'job_name': '1950b29aef9a42afb1eb17125cea38a5',
  'job_id': '567ddbe4-a896-11e9-b2c6-c3499b5ebff0',
  'profile_name': '1700ee4ecb854ba8977a06f85efd1644',
  'profile_id': '567f38e0-a896-11e9-b2c6-770cfb81b384'})

In [18]: s.create_model('pacman-smoky', \
  fit_profile_name='1700ee4ecb854ba8977a06f85efd1644',\
  max_train_time='00:02', recurrent=False, failure_dates=['08/23/2015'], nbm=True)
Out[18]:
(True,
 {'job_name': '23ce4b284df14f0ca14c0a49f63806fc',
  'job_id': '8b91e172-a896-11e9-be40-df999689405b',
  'model_name': '7e080e51cc15408492d6136e07df2a63'})

In [19]: s.wait_for_job('23ce4b284df14f0ca14c0a49f63806fc')
{'status': 'Running', 'starttime': '2019-07-17T08:26:55.165137', 'endtime': None,\
 'percent_complete': 0, 'job_type': 'TrainModel', 'loss': None, 'generations': 0,\
  'dataset_names': ['pacman-smoky'], 'artifact_names': None,\
   'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
{'status': 'Running', 'starttime': '2019-07-17T08:26:55.165137', 'endtime': None,\
 'percent_complete': 0, 'job_type': 'TrainModel', 'loss': None, 'generations': 0,\
  'dataset_names': ['pacman-smoky'], 'artifact_names': None,\
   'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
...
{'status': 'Complete', 'starttime': '2019-07-17T08:26:55.165137',\
 'endtime': '2019-07-17T08:29:11.943704', 'percent_complete': 100,\
  'job_type': 'TrainModel', 'loss': 0.054839795631057814, 'generations': 1,\
   'dataset_names': ['pacman-smoky'], 'artifact_names': None,\
    'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
Out[19]: (True, 'Job completed')

In [21]: s.run_model('pacman-smoky', '7e080e51cc15408492d6136e07df2a63')
Out[21]:
(True,
 {'job_name': '268b0fb872054ec58c0344053625e69c',
  'job_id': '1351b15a-a897-11e9-a06a-2b396d284046',
  'artifact_name': '4fd95f6ec4c442e586b941f7e2656dbf'})

In [24]: s.wait_for_job('268b0fb872054ec58c0344053625e69c')
{'status': 'Complete', 'starttime': '2019-07-17T08:30:42.913437',\
 'endtime': '2019-07-17T08:31:08.226901', 'percent_complete': 100,\
  'job_type': 'RunModel', 'loss': 0.054839795631057814, 'generations': 1,\
   'dataset_names': ['pacman-smoky'], \
   'artifact_names': ['4fd95f6ec4c442e586b941f7e2656dbf'],\
    'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
Out[24]: (True, 'Job completed')

In [25]: s.download_artifact('4fd95f6ec4c442e586b941f7e2656dbf')
Out[25]:
(True,            Risk
 0      0.000000
 1      0.000000
 2      0.000000
 3      0.000000
 4      0.000000
...
 51401  3.175640
 51402  3.153954
 51403  3.131895
 51404  3.109017
 51405  3.087970

 [51406 rows x 1 columns])

In [27]: s.upload_dataset('sets/SmokyT025_test_raw.csv', 'pacman-smokytest')
```

```
Out[27]: (True, {'dataset_name': 'pacman-smokytest'})

In [28]: s.clean_data('pacman-smokytest', \
 model_name='7e080e51cc15408492d6136e07df2a63')
Out[28]:
(True,
 {'job_name': '229b24bfc46d4601aef1ce8b017e4ff7',
  'job_id': 'bd369cca-a898-11e9-8ea7-2bc9c8ab84c4',
  'profile_name': 'b4f5cc08f980442f983a8137a756dc79',
  'profile_id': 'bd38ac40-a898-11e9-8ea7-9fb23d6a2658'})

In [29]: s.run_model('pacman-smokytest', '7e080e51cc15408492d6136e07df2a63')
Out[29]:
(True,
 {'job_name': '35d9cd3b4a3b45a99eb8dc089ef9b152',
  'job_id': 'c7d0d5ce-a898-11e9-bfef-379c562f822e',
  'artifact_name': 'f750090766f14e82b3713e3f3ef9bcb5'})

In [30]: s.wait_for_job('35d9cd3b4a3b45a99eb8dc089ef9b152')
{'status': 'Complete', 'starttime': '2019-07-17T08:42:55.232948',\
 'endtime': '2019-07-17T08:43:01.540698', 'percent_complete': 100,\
  'job_type': 'RunModel', 'loss': 0.054839795631057814, 'generations': 1,\
   'dataset_names': ['pacman-smokytest'], \
   'artifact_names': ['f750090766f14e82b3713e3f3ef9bcb5'],\
   'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
Out[30]: (True, 'Job completed')

In [31]: s.download_artifact('f750090766f14e82b3713e3f3ef9bcb5')
Out[31]:
(True,            Risk
 0      0.000000
 1      0.000000
 2      0.000000
 3      0.000000
...
 17276  3.121538
 17277  3.098760
 17278  3.077858

 [17279 rows x 1 columns])

In [32]: s.analyze_model('7e080e51cc15408492d6136e07df2a63')
Out[32]:
(True,
 {'job_name': 'e54bfd3d620d44b083f5ea65dda12aec',
    'job_id': '605a6300-a899-11e9-bfef-6fedf02afe89',
    'artifact_name': 'd8ea781de54c4303ad49e17f0d208db3'})

In [33]: s.wait_for_job('e54bfd3d620d44b083f5ea65dda12aec')
{'status': 'Running', 'starttime': '2019-07-17T08:47:11.149655', 'endtime': None,\
 'percent_complete': 0, 'job_type': 'AnalyzeModel', 'loss': 0.054839795631057814,\
  'generations': 1, 'dataset_names': None, \
  'artifact_names': ['d8ea781de54c4303ad49e17f0d208db3'],\
  'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
{'status': 'Complete', 'starttime': '2019-07-17T08:47:11.149655',\
 'endtime': '2019-07-17T08:47:27.795439', 'percent_complete': 100,\
  'job_type': 'AnalyzeModel', 'loss': 0.054839795631057814, 'generations': 1,\
   'dataset_names': None, 'artifact_names': ['d8ea781de54c4303ad49e17f0d208db3'],\
    'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}

Out[33]: (True, 'Job completed')

In [34]: s.download_artifact('d8ea781de54c4303ad49e17f0d208db3')
Out[34]:
(True, ICA                 0.290290
 PSETKW              0.095161
 GENTMPSLIPDEGC      0.075301
 AMBTMPDEGC          0.062500
 NACTMPDEGC          0.040816
 XFMRTMPPHCDEGC      0.034540
 GENTMPPHADEGC       0.032311
 YAWDIRDEG           0.031473
 GENTMPPHCDEGC       0.027232
 WDRELDEG            0.026735
 HYDRPRESBAR         0.026336
 FREQHZ              0.022709
 XFMRTMPPHADEGC      0.020440
 GENTMPPHBDEGC       0.018139
 BRGTMPGENNDEDEGC    0.017267
 WD10MDEG            0.016984
 VANV                0.015967
 PF                  0.012871
 TMPCTRLTOPDEGC      0.012038
 IBA                 0.011195
 TMPSPINNERDEGC      0.011191
 BLDANGDEG           0.010891
 WS10MMPS            0.010032
 VBNV                0.009880
 VCNV                0.008443
 XFMRTMPPHBDEGC      0.007716
 OILTMPGBXDEGC       0.007068
 GENSPDRPM           0.006877
 TMPCTRLHUBDEGC      0.006240
 PEXPKW              0.005889
 WSMPS               0.005433
 BRGTMPGBXADEGC      0.004322
 QKVAR               0.004213
 HYDRTMPDEGC         0.003860
 P10MACTKW           0.003125
 RTRSPDRPM           0.002313
 IAA                 0.002204
 dtype: float64)

In [39]: s.upload_dataset('sets/SmokyT025_test_small.csv', 'pacman-smokytestsmall')
Out[39]: (True, {'dataset_name': 'pacman-smokytestsmall'})

In [40]: s.clean_data('pacman-smokytestsmall', \
   model_name='7e080e51cc15408492d6136e07df2a63')
Out[40]:
(True,
 {'job_name': 'a9a68568b4f44fe19661f0b5d4e8822d',
  'job_id': 'f68898c4-a899-11e9-9e51-2b93b6669bbd',
  'profile_name': '5c90c01159684d6491d956f035651c6f',
  'profile_id': 'f68a2a5e-a899-11e9-9e51-5bba6c3dc2d1'})

In [41]: s.analyze_predictions ('7e080e51cc15408492d6136e07df2a63',\
   'pacman-smokytestsmall')
Out[41]:
(True,
 {'job_name': '04e5d614672a4e598276bd19a06f7df9',
  'job_id': 'fe1c8906-a899-11e9-8ea7-cfc85b8e9384',
  'artifact_name': '96fd4aa1b57043d69b658e72071c35c9'})

In [42]: s.wait_for_job('04e5d614672a4e598276bd19a06f7df9')
{'status': 'Running', 'starttime': '2019-07-17T08:51:35.824573', 'endtime': None,\
 'percent_complete': 0, 'job_type': 'AnalyzePredictions', \
 'loss': 0.054839795631057814,'generations': 1, 'dataset_names': None, \
 'artifact_names': ['96fd4aa1b57043d69b658e72071c35c9'],\
 'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
{'status': 'Running', 'starttime': '2019-07-17T08:51:35.824573', 'endtime': None,\
 'percent_complete': 0, 'job_type': 'AnalyzePredictions', \
 'loss': 0.054839795631057814, 'generations': 1, 'dataset_names': None, \
 'artifact_names': ['96fd4aa1b57043d69b658e72071c35c9'],\
 'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
 ...

In [47]: s.upload_dataset('sets/pulsars.csv', 'pacman-pulsars')
Out[47]: (True, {'dataset_name': 'pacman-pulsars'})

In [48]: s.analyze_data('pacman-pulsars')
Out[48]:
(True,
 {'job_name': '294ed354f3484b2ebebe658033284128',
  'job_id': '80193b9c-a63e-11e9-85df-fba3623db3bc',
  'artifact_name': 'b3d8dacf77fc409b9dd48030b2dda07b'})

In [49]: s.clean_data('pacman-pulsars')
Out[49]:
(True,
 {'job_name': '454f781b2a02403ea74d915a4b6b530c',
  'job_id': '8763baf8-a63e-11e9-900b-4335c8f0f324',
  'artifact_name': 'b9820a205bf140b1ae24bc6b1b133d2d',
  'artifact_id': '876560d8-a63e-11e9-900b-7f0b539a80a9'})

In [50]: s.create_model('pacman-pulsars', \
 fit_profile_name='b9820a205bf140b1ae24bc6b1b133d2d')
Out[50]:
(True,
 {'job_name': '2a9a1e55f8e34ddd828326ecee2b42f4',
  'job_id': '97a92592-a63e-11e9-a08f-cf415e11311a',
  'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265'})

In [51]: s.wait_for_job('2a9a1e55f8e34ddd828326ecee2b42f4')
{'status': 'Running', 'starttime': '2019-07-14T08:52:17.412985', 'endtime': None, \
'percent_complete': 0, 'job_type': 'TrainModel', 'loss': None, 'generations': 0, \
'dataset_names': ['pacman-pulsars'], 'artifact_names': None, \
'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}
{'status': 'Running', 'starttime': '2019-07-14T08:52:17.412985', 'endtime': None, \
'percent_complete': 0, 'job_type': 'TrainModel', 'loss': None, 'generations': 0, \
'dataset_names': ['pacman-pulsars'], 'artifact_names': None, \
'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}
....
{'status': 'Complete', 'starttime': '2019-07-14T08:52:17.412985', \
'endtime': '2019-07-14T08:53:53.736499', 'percent_complete': 100, \
'job_type': 'TrainModel', 'loss': None, 'generations': 0, \
'dataset_names': ['pacman-pulsars'], 'artifact_names': None, \
'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}
Out[51]: (True, 'Job completed')

In [53]: s.run_model('pacman-pulsars', 'fa1b82cab28c46cdac3b44c8e8bc1265')
```

```
Out[53]:
(True,
 {'job_name': 'c72ef79400014a3fb9e0a5821adf0826',
  'job_id': '0b58fb52-a63f-11e9-9f7b-53a24e4e1e2a',
  'artifact_name': '05380df9e93c4650ab6f7e5e67e23f72'})

In [54]: s.wait_for_job('c72ef79400014a3fb9e0a5821adf0826')
{'status': 'Complete', 'starttime': '2019-07-14T08:55:31.501449', \
'endtime': '2019-07-14T08:55:34.679054', 'percent_complete': 100, \
'job_type': 'RunModel', 'loss': None, 'generations': 0, \
'dataset_names': ['pacman-pulsars'], \
'artifact_names': ['05380df9e93c4650ab6f7e5e67e23f72'], \
'model_name': 'fa1b82cab28c46cdac3b44c8e8bc1265', 'job_error': ''}
Out[54]: (True, 'Job completed')

In [55]: s.download_artifact('05380df9e93c4650ab6f7e5e67e23f72')
Out[55]:
(True,
 {'filename': '/var/folders/wc/w7ktf3_h8br6t2djlhyb9/T/artifact-6ac2m9yp.csv'})

In [17]: s.analyze_model('b7d28ee423d4430fafa51a017be827ac')
Out[17]:
(True,
 {'job_name': 'b36800c1a47f4c559c231d155a12fd85',
  'job_id': '8c5dfd70-a8f5-11e9-88ca-173db83c1239',
  'artifact_name': 'a417260dfabc409a8204fa1ceeae112f'})

In [18]: s.wait_for_job('b36800c1a47f4c559c231d155a12fd85')
{'status': 'Complete', 'starttime': '2019-07-17T19:46:58.691115', \
'endtime': '2019-07-17T19:47:02.152927', 'percent_complete': 100, \
'job_type': 'AnalyzeModel', 'loss': None, 'generations': 0, \
'dataset_names': None, 'artifact_names': ['a417260dfabc409a8204fa1ceeae112f'], \
'model_name': 'b7d28ee423d4430fafa51a017be827ac', 'job_error': ''}
Out[18]: (True, 'Job completed')

In [19]: s.download_artifact('a417260dfabc409a8204fa1ceeae112f')
Out[19]:
(True, kurt_dmsnr      0.0
 skew_dmsnr      0.0
 skew_profile    0.0
 kurt_profile    0.0
 class           0.0
 std_profile     0.0
 mean_dmsnr      0.0
 mean_profile    0.0
 std_dmsnr       0.0
 dtype: float64)

In [22]: s.upload_dataset('sets/pulsars_predict.csv', 'pulsars-test') \
 # Need to trim original dataset to have fewer than 500 rows.
Out[22]: (True, {'dataset_name': 'pulsars-test'})

In [23]: s.clean_data('pulsars-test', model_name='b7d28ee423d4430fafa51a017be827ac')
Out[23]:
(True,
 {'job_name': '8324da8dea734455a73daeeddd3e0b5f',
  'job_id': 'fc2a4974-a8f5-11e9-9074-7fa762e40db7',
  'profile_name': 'bccde471e8514ef59b0b106fa7af6be9',
  'profile_id': 'fc2c0e76-a8f5-11e9-9074-13d39d00d68d'})

In [24]: s.analyze_predictions ('b7d28ee423d4430fafa51a017be827ac', \
'pulsars-test')
Out[24]:
(True,
 {'job_name': 'def26ef5be3a4d5b822542fd125c8600',
  'job_id': '12995024-a8f6-11e9-89b8-cf7654542d20',
  'artifact_name': 'd726a45761a1431d8bbe381c4f4f2782'})

In [25]: s.wait_for_job('def26ef5be3a4d5b822542fd125c8600')
{'status': 'Complete', 'starttime': '2019-07-17T19:50:43.895873', \
'endtime': '2019-07-17T19:50:49.098101', 'percent_complete': 100, \
'job_type': 'AnalyzePredictions', 'loss': None, 'generations': 0, \
'dataset_names': None, 'artifact_names': ['d726a45761a1431d8bbe381c4f4f2782'], \
'model_name': 'b7d28ee423d4430fafa51a017be827ac', 'job_error': ''}
Out[25]: (True, 'Job completed')
In [26]: s.download_artifact('d726a45761a1431d8bbe381c4f4f2782')
Out[26]:
(True,
    std_dmsnr_shap  mean_dmsnr_shap  kurt_dmsnr_shap  mean_profile_shap  ... \
 0             0.0              0.0              0.0                0.0
 1             0.0              0.0              0.0                0.0
 2             0.0              0.0              0.0                0.0
 3             0.0              0.0              0.0                0.0
 4             0.0              0.0              0.0                0.0
 5             0.0              0.0              0.0                0.0
 6             0.0              0.0              0.0                0.0
 7             0.0              0.0              0.0                0.0
 8             0.0              0.0              0.0                0.0
 [9 rows x 12 columns])
```

```
  In [27]:
  {'status': 'Complete', 'starttime': '2019-07-17T08:51:35.824573', \
   'endtime': '2019-07-17T08:55:05.619098', 'percent_complete': 100, \
   'job_type': 'AnalyzePredictions', 'loss': 0.054839795631057814, \
   'generations': 1, 'dataset_names': None,\
   'artifact_names': ['96fd4aa1b57043d69b658e72071c35c9'],\
   'model_name': '7e080e51cc15408492d6136e07df2a63', 'job_error': ''}
  Out[42]: (True, 'Job completed')

  In [43]: s.download_artifact('96fd4aa1b57043d69b658e72071c35c9')
  Out[43]:
  (True,
        GENSPDRPM_shap  VANV_shap  PSETKW_shap  P10MACTKW_shap  ... \
   0         20.060011   5.785717     0.000000       -1.762059
   1         -1.002102  -4.857683    -1.833142        0.000000
   2          0.641536   0.015167     0.000000        0.000000
   3          0.000000  -1.827971     0.000000        0.000000
   4         -5.824179   6.567310    10.847957       -5.231445
   5        -16.634360   0.000000    -8.437045       -9.929700
   ..              ...        ...          ...             ...

   396       -1.932885  -1.780289    -1.361495        0.000000
   397        5.469176  -8.030168    38.908720       -1.336821
   398       -0.598940  -8.100026    57.731217       17.159244

   [399 rows x 39 columns])
```

Forecasting modeling example

```
---
In [1]: from amb_sdk.sdk import DarwinSdk
        s = DarwinSdk()

In [4]: PATH_TO_DATASET = '../sets/'
        TRAIN_DATASET = 'nyc_taxi_train.csv'
        TEST_DATASET = 'nyc_taxi_test.csv'

        ts = '{:%Y%m%d%H%M%S}'.format(datetime.datetime.now())

In [5]: USER="username"
        PW="password"

In [17]: s.upload_dataset('../sets/nyc_taxi_train.csv')
```

```
Out[17]:
        (True, {'dataset_name': 'nyc_taxi_train.csv'})

In [18]:
        s.analyze_data('nyc_taxi_train.csv')

Out[18]:
(True,
 {'job_name': 'd879f6dd68424e46828acd405f05a332',
  'job_id': '5bec6a9a-371c-11ea-ab27-4361ffef1ed8',
  'artifact_name': '389e1d23e12f429b8eb99bbf8226251f'})

In [19]:
s.clean_data('nyc_taxi_train.csv', target = 'value')

Out[19]:
(True,
 {'job_name': '83e3494cf86f45ffb6ad9e289275a95d',
  'job_id': '5e82a468-371c-11ea-aa95-c3526d2b4370',
  'profile_name': '2cd12aaf9f03433aac01846ab61eb7d9',
  'profile_id': '5ebf292e-371c-11ea-aa95-570e857a0c83'})

In [20]:
s.create_model('nyc_taxi_train.csv',
        forecast_horizon = 3,
        fit_profile_name = '2cd12aaf9f03433aac01846ab61eb7d9')

Out[20]:
(True,
 {'job_name': 'c92fc19050a54d5b851f2844434c9bb9',
  'job_id': '7e92bb12-371c-11ea-ba16-fbb8f5911d82',
  'model_name': '9c45bf92e9884ac4a6eea33236efce00'})

In [21]:
s.wait_for_job('c92fc19050a54d5b851f2844434c9bb9')
{'status': 'Running', 'starttime': '2020-01-14T22:23:31.003464', \
'endtime': None, 'percent_complete': 0, 'job_type': 'TrainModel', \
'loss': None, 'generations': 0, 'dataset_names': ['nyc_taxi_train.csv'], \
'artifact_names': None, 'model_name': '9c45bf92e9884ac4a6eea33236efce00', \
'job_error': ''}

... ...

{'status': 'Complete', 'starttime': '2020-01-14T22:23:31.003464', \
'endtime': '2020-01-14T22:33:42.652494', 'percent_complete': 100, \
'job_type': 'TrainModel', 'loss': 0.056668907636776567, 'generations': 1, \
'dataset_names': ['nyc_taxi_train.csv'], 'artifact_names': None, \
'model_name': '9c45bf92e9884ac4a6eea33236efce00', 'job_error': ''}

Out[21]:
(True, 'Job completed')

In [22]:
s.lookup_model_name('9c45bf92e9884ac4a6eea33236efce00')
Out[22]:
(True,
 {'id': '7e82f48e-371c-11ea-ba16-4f223b72c83a',
  'name': '9c45bf92e9884ac4a6eea33236efce00',
  'type': 'Supervised',
  'problem_type': None,
  'updated_at': '2020-01-14T22:33:42.644902',
  'trained_on': ['nyc_taxi_train.csv'],
  'trained_on_id': ['5ae11074-371c-11ea-91ac-9bf950514930'],
  'loss': 0.056668907636776567,
  'complete': True,
  'generations': 1,
  'parameters': {'forecast_horizon': 3,
   'target': 'value',
   'train_time': '00:10',
   'recurrent': False,
   'max_unique_values': 50,
   'max_int_uniques': 15,
   'impute': 'mean',
   'big_data': False},
  'description': {'best_genome': [{'layer 1': {'type': 'DualAttentionRecurrentNeuralNet',
      'parameters': {'encoder_hidden_size': 78,
       'decoder_hidden_size': 79,
       'seqlength': 20}}},
    {'layer 2': {'type': 'LinearGene',
      'parameters': {'activation': 'identity', 'numunits': 3}}}],
   'recurrent': True,
   'genome_type': 'DeepNet'},
  'train_time_seconds': 612,
  'algorithm': None,
  'running_job_id': None})

In [31]:
s.run_model('nyc_taxi_test.csv',
            model_name = '9c45bf92e9884ac4a6eea33236efce00' )
Out[31]:
```

```
(True,
 {'job_name': '5c9f47ecf2da457c8e789a811959672c',
  'job_id': '05c103ee-371f-11ea-bf24-c7ed4abe719b',
  'artifact_name': '6ae83c0d59c9411a8f3f596a33057834'})
In [32]:
s.wait_for_job('5c9f47ecf2da457c8e789a811959672c')
{'status': 'Running', 'starttime': '2020-01-14T22:41:36.790956', \
'endtime': None, 'percent_complete': 0, 'job_type': 'RunModel', \
'loss': 0.056668907636776567, 'generations': 1, \
'dataset_names': ['nyc_taxi_test.csv'], 'artifact_names': \
['6ae83c0d59c9411a8f3f596a33057834'], 'model_name': \
'9c45bf92e9884ac4a6eea33236efce00', 'job_error': ''}

...

{'status': 'Complete', 'starttime': '2020-01-14T22:41:36.790956', \
'endtime': '2020-01-14T22:42:01.78274', 'percent_complete': 100, \
'job_type': 'RunModel', 'loss': 0.056668907636776567, 'generations': 1,\
 'dataset_names': ['nyc_taxi_test.csv'], 'artifact_names': \
 ['6ae83c0d59c9411a8f3f596a33057834'], 'model_name': \
 '9c45bf92e9884ac4a6eea33236efce00', 'job_error': ''}

Out[32]:
(True, 'Job completed')
In [33]:
s.download_artifact('6ae83c0d59c9411a8f3f596a33057834')
Out[33]:
(True,          future_1      future_2      future_3
 0    11306.876953  10307.818359   9128.345703
 1    11661.523438  10513.156250   9195.193359
 2     8693.410156   7886.320801   7096.866211
 3     6402.957031   5710.714844   5166.185547
 4     4489.059570   4121.243164   3995.698242
 5     2979.267578   3142.290039   3468.699219

 ..            ...           ...           ...

 958  26081.808594  26612.792969  26809.789062
 959  26810.261719  26978.746094  27026.882812
 960  27688.652344  27256.580078  27052.052734
 961  26898.109375  26352.546875  26050.978516
```

Contact Support

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

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

Revision Table

Version	Date	Notes
v 1.6	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
v 1.6.1	06-Feb-2019	Fixed issues only. See Release Notes. Added on-prem installation notes.
v 1.6.2	22-Mar-2019	New endpoints: • DarwinSdk.get_info • DarwinSdk.help Added Setup Users section. On-prem SDK users need to add port 8000 to the URL.
v 1.43.0	16-May-2019	Major change to version number to facilitate independent releases of the API New endpoints: • DarwinSdk.disable_ssl_cert_check • DarwinSdk.enable_ssl_cert_check • DarwinSdk.get_sdk_version Updated endpoints: • DarwinSdk.create_model
v 1.44.0	22-Jul-2019	Updated modeling examples for Supervised, Unsupervised, and NBM Added Analyze Training Data step to basic workflow Updated endpoints: • DarwinSdk.create_model: Added forecast_horizon, class_weights, cv_kfold, fit_profile_name
v 1.44.1	24-Sep-2019	Removed model_type parameter from run_model() Removed impute parameter from clean_data()

Version	Date	Notes
v 1.45.0	18-Dec-2019	Added model_type parameter to run_model()
		Added impute parameter to clean_data()
		Added fitness_fn_name to create_model()
		Limited to 4000 columns in dataset
v 1.46.0	30-Jan-2020	Added new method: align_forecasting_predictions()
		Removed forecast_horizon from run_model()
		Added char_encoding to analyze_data().