Skytree Python SDK Quick Start

Release 15.3.0

SKYTREE®

Notices

Copyright

Disclaimer

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

Trademarks

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

Contact Support:

- Log in to support.skytree.net.
- Send questions to support@skytree.net.
- Call 408.392.9300 and select option 3.

Revision Date: December 21, 2015

SKYTREE®

Contents

SKYTREE®

About this Document

This document describes how to get started using the Skytree Python SDK, which supports Python 2.7. For detailed information about all of the functions available through the SDK, refer to the *Skytree Python SDK* document.

Using the included `income.data` file, this document will take you through the following steps:

1. Preparing your environment
    - Installing the Python SDK
    - Importing the `skytree` and `skytree.prediction` modules
2. Authenticating
3. Creating a Project
4. Adding a dataset to the project
5. Transforming the dataset to produce a final dataset
    - Splitting the dataset into training and testing datasets
    - Adding a unique ID column
    - Setting the ID column
6. Commiting the dataset
7. Training a model based on the transformed dataset
8. Testing the model
9. Viewing results

Assumptions

- This document assumes that you already have a Skytree user account set up by your Admin and that your Admin has provided you with a valid e-mail address, password, and hostname. This information is required when authenticating.
- Setuptools must already be bootstrapped. The Python SDK cannot install without setuptools. Any version compatible with Python 2.7 is supported. At the time of this writing, the current setuptools version was 15.2. Refer to the python.org site for installation instructions.
- This document assumes that the actions performed here will be done in the default environment.

Step 1. Preparing Your Environment

You must prepare your environment before you can use the Skytree Python SDK. In order to do this, you must know the location of the Skytree Python SDK directory. Contact your Admin for this information.

1. Open a terminal window and run the following commands to install/upgrade the Skytree Python SDK. Note that this also installs an `/examples` folder, which includes sample datasets, including the `income.data` file that will be used here.

   ```
   cd <python_sdk_directory>
   python setup.py install
   ```

   Note that the above command assumes you have sudo access. If you do not, then run the following:

   ```
   python setup.py install --user
   ```

2. In Python, import the following modules:

   ```
   $ python
   >>> import skytree
   >>> import skytree.prediction
   >>> from skytree.prediction import gbt
   ```

Step 2. Authenticating

Using the information provided by your Admin, run the following function to log in and begin using the Skytree Python SDK. You will need to include a valid e-mail address, password, and hostname (including `http://` or `https://`).

```
skytree.authenticate("<email>", "<password>", hostname="<host_url>")
```

For example, the command below creates a user whose e-mail address is `jack@skytree.net`. Jack's password is `JackPwd!`. The hostname is `http://192.168.0.0:8080/v1`.

```
skytree.authenticate(
    "jack@skytree.net",
    "JackPwd!"
    hostname="http://192.168.0.0:8080/v1"
)
```

Note: Your password must be at least eight characters in length and must include at least one uppercase character, one lowercase character, and one number. Special characters are permitted. Also, `http://` or `https://` is required in the hostname.

Step 3. Creating a Project

Skytree makes use of project-based machine learning. Whether you are trying to detect fraud or predict user retention, the datasets, models, and results are stored and saved in the individual projects.

The example below shows how to create a new project called `New Customers`. The description of the project is `Track leads`. The Project Object is defined as `project`. This object will be referenced when adding datasets to this project.

```
project = skytree.create_project("New Customers", "Track leads")
```

Step 4. Adding a Dataset to a Project

Using the New Customers project that you just created, the example below shows how to add the `income.data` dataset to a Dataset Object called `dataset`. The path to this dataset is in the `examples` folder, which was installed during Step 1. (Refer to *Preparing Your Environment* (page 2).) The `income.data` file includes a header row, its entries are separated by commas, and missing values are indicated with a `?`. The Project Object for this is `project`. This imported file will not have an included ID column. This will be set in an upcoming transform example.

```
dataset = project.create_dataset(
      path="examples/income.data",
      name="income.data",
      has_header=True,
      separator=",",
      missing_value="?"
)
dataset.ready()
```

In the above example, `create_dataset()` makes the REST API call and returns as soon as the data is uploaded, returning a Dataset Object (`dataset` in the example above). This kicks off the dataset creation process on the server. The `ready()` function is a blocking call that polls the server continuously until the resource (`dataset`) is available for use (i.e, has a `READY` status). Models, tests, and datasets must be `READY` before they can be used.

Step 5. Transforming the Dataset

Transform 1 - Split the dataset into train and test datasets

In order to properly evaluate a machine learning model, it needs to be tested on data not used to train it. We will start by splitting the datasets into train and test datasets, keeping 70% of the data for training and 30% for testing.

```
train, test = dataset.split([0.7, 0.3], names=["train", "test"])
train.ready()
test.ready()
```

Transform 2 - Add a unique ID column

The income dataset that was split did not include the optional ID column. The `add_unique_id_column()` function adds a column named `IDs` to the `test` dataset and returns a new Dataset Object, `uniqueid`. The name for this transformed dataset is `NewIDColumn`.

```
uniqueid = test.add_unique_id_column(
   "IDs", name="NewIdColumn"
).ready()
```

Transform 3 - Set the ID column

Now that the new ID column was created, use the `set_as_id()` transform to set the ID column as the ID column for Predict and Evaluate.

```
dataset2 = uniqueid.set_as_id("IDs", name="NewIncomeId").ready()
```

Step 6. Committing a Dataset

When a dataset is initially added to a project, the server uses a sample of this dataset for all operations. This is done to enable a more interactive user experience while prototyping. Before building or testing a model, the server will automatically process the full dataset. If you want to trigger the full process manually prior to building or testing a model, use the `commit()` function to recompute the metadata across the entire dataset. In the example below, a commit is done on `dataset2`.

```
dataset2.commit().ready()
```

You can run the following to verify the status of the dataset has changed to committed.

```
for dataset in project.list_datasets:
        print dataset.name
        print dataset.status
```

The output should be the following, showing the most recently updated dataset at the top. This shows that the dataset was successfully committed.

```
NewIncomeId
{u'message': u'COMMITTED: Data Set has been validated and is
   ready to use', u'code': u'READY'}
NewIdColumn
{u'message': u'SAMPLED: Data Set has been validated and is
   ready to use', u'code': u'READY'}
test
{u'message': u'SAMPLED: Data Set has been validated and is
   ready to use', u'code': u'READY'}
train
{u'message': u'SAMPLED: Data Set has been validated and is
   ready to use', u'code': u'READY'}
income.data
{u'message': u'SAMPLED: Data Set has been validated and is
   ready to use', u'code': u'READY'}
```

Step 7. Training a Model

The following example shows how to build a simple GBT model over 100 trees with a tree depth of 4. The Dataset object `train` indicates that the transformed and then committed dataset will be used to train this model. The GBT module automatically recognizes this as a classification problem because the `objective_column` (specified as "`yearly-income`") is categorical rather than continuous.

Create the Configuration Object

```
gbtconfig = gbt.Config()
gbtconfig.num_trees = 100
gbtconfig.tree_depth = 4
```

Specify the Configuration Object when creating the model

This configuration requires the Dataset object (`train`), the objective column (`"yearly-income"`), the Configuration Object (`gbtconfig`), and an optional name for the model (`"GbtModel"`). The Model Object for this is defined as `model`.

```
model = skytree.prediction.learn(
    train, "yearly-income", gbtconfig, name="GbtModel"
).ready()
```

Step 8. Testing a Model

This example shows how to test the GBT model using the `test()` function. Specifically, this tests the GBT model that you just built (`model`) against the `dataset2` dataset. `dataset2` includes an ID column that was added during the transformation. (Note that datasets that do not have an ID column cannot be referenced in a test.)

```
results = model.test(dataset2, name="TestPredict").ready()
```

Step 9. Viewing Results

After the test has been generated, you can use the `get_probabilities()` and `get_labels()` functions to retrieve the associated labels and probabilities. The Test object for retrieving these is `results`.

View the summary

```
print results.summary()
```

This displays a summary of the test results.

```
Skytree classification score summary
------------------------
False positive rate: 0.0499594
Precision:           0.80861
Recall:              0.663122
Accuracy:            0.880764
F score:             0.728675
Confusion matrix:
---
Class -1: Wrong: 369. Right: 7017. Total 7386.
Class  1: Wrong: 792. Right: 1559. Total 2351.
Overall : Wrong: 1161. Right: 8576. Total 9737.
---
Gini:                0.864919
Capture Deviation:   0.0593205
------------------------
```

Get the Probabilities

```
probs = results.get_probabilities()
```

Print the top Probabilities

```
for (id, probability) in results.get_probabilities():
        print (id, probability)
```

The output will be similar to below.

```
('0', 0.3607444411175226)
('2', 0.023960168573792007)
('4', 0.8332312221859394)
('6', 0.70833187123728)
('8', 0.46353463624366076)
('10', 0.04634450089245616)
('12', 0.01623127871765511)
('14', 0.010303023796566994)
('16', 0.3753666698975811)
...
```

Get the Labels

```
labels = results.get_labels()
```

Print the top 20 Labels

```
for (id, label) in results.get_labels():
        print (id, label)
```

The output will be similar to below.

```
('0', -1)
('2', -1)
('4', 1)
('6', 1)
('8', -1)
('10', -1)
('12', -1)
('14', -1)
('16', -1)
...
```

What's Next?

Now that you have finished your first Python SDK example, you are ready to begin creating models using your own data. Refer to the *Skytree Python SDK Guide* for information on all of the features, model building methods, and configuration options available in Skytree.