# Daimensions™ Docker Quickstart

## Requirements for installing a standalone Daimensions docker container:

- A machine with docker installed. (For instructions on how to install docker, please refer to https://docs.docker.com/install/)
- A Docker Hub credential to download the image from Brainome repository
- User login privilege to the server on which docker is running
- Access to your data on the server
- python3 installed on your machine (to run predictors)

**NOTE**: Daimensions™ standalone docker runs locally and **does not upload any customer data to Brainome's cloud servers**.

## Installing or Updating Daimensions' docker image

### Step 1: Download the btc-docker script to your server.

btc-docker is used to run Daimension's docker container. It will automatically download the master container, create a custom user container and run Daimensions on your container. Download and save btc-docker at: https://download.brainome.net/btc-docker/latest/btc-docker The script is used to install the docker image AND to run btc.

### Step 2 : Install Daimensions's docker on your server

From a terminal on the server, run btc-docker to install OR update the Daimensions's image from the docker hub repository:

```
server: chmod +x btc-docker
server:./btc-docker -update-server
Updating existing docker setup
Updating brainome/btc_local_gpu:latest
Authenticating with existing credentials...
Login Succeeded
latest: Pulling from brainome/btc_local_gpu
7ddbc47eeb70: Pull complete
c1bbdc448b72: Pull complete
8c3b70e39044: Pull complete
. . . lines deleted ...
```

```
ef30acc59017: Pull complete
Digest:
sha256:e798879b6d335b0ad198fd3dbdfa4d45158a6e196d93387ab6efd02088d602e2
Status: Downloaded newer image for brainome/btc_local_gpu:latest
docker.io/brainome/btc_local_gpu:latest
Creating user docker image btc-user
Docker image btc-bertrand was updated properly.
```

**IMPORTANT: Use the same Docker Hub credentials that were used to register your Daimensions trial with Brainome.**

NOTE: The server needs to be able to connect to hub.docker.com to download the image.

## Step 3a: Installing the license file - Offline

Unzip the license file that was emailed to you in your **working directory**.

```
server: unzip license.zip
server: ls -l .daimensions.key
```

## Step 3b: Installing the license file - Online

If you have internet access, you do not need to manually download the license file. It will be downloaded automatically the first time your run btc by entering your credentials (email and password)

## Step 4: Rename btc-docker and place it in your path

For ease of use, we recommend renaming btc-docker to “btc” and placing it in a location that is in your command path. For instance, if you have a /usr/local/bin directory on your machine, we suggest doing the following:

```
server: sudo cp btc-docker /usr/local/bin/btc
server: sudo chmod a+rx /usr/local/bin/btc
```

# Running Daimensions™

For this tutorial, we will use a data file from OpenML.org located at:
https://www.openml.org/data/get_csv/53488/spectrometer.arff

## Step 1: Download the example data file into your data directory

```
curl https://www.openml.org/data/get_csv/53488/spectrometer.arff -o spectrometer.csv
```

NOTE: Version v0.96 of Daimensions supports only data files in CSV format.

## Step 2: Get your measurements

The "-measureonly" option performs measurement only on the data set:

```
btc -measureonly spectrometer.csv
Brainome Daimensions(tm) 0.96 Copyright (c) 2019, 2020 by Brainome, Inc. All Rights
Reserved.
Licensed to: Your Name
Expiration date: 2020-08-30 (89 days left)
Number of threads: 1
Maximum file size: 1GB
Connected to: daimensions.brainome.ai

Note: Assuming machine learner type QC for compiling efficiency (not accuracy).
Note: Choice can be overridden with -f parameter.
Data:
Number of instances: 531
Number of attributes: 102
Number of classes: 2
Class balance: 89.64% 10.17%

Learnability:
Best guess accuracy: 89.64%
Capacity progression (# of decision points): [3, 5, 6, 6, 7, 7]
Quick Clustering: 100 parameters
Estimated Memory Equivalent Capacity for Neural Networks: 729 parameters

Risk that model needs to overfit for high accuracies...
using Quick clustering: 37.66%
using Neural Networks: 77.80%
```

```
Expected Generalization...
using Quick clustering: 5.31 bits/bit
using a Neural Network: 0.73 bits/bit

Recommendations:
Note: Maybe enough data to generalize. [yellow]
Note: Quick Clustering may outperform Neural Networks. Try with -f QC.

Time estimate for a Neural Network:
Estimated time to architect: 0d 0h 0m 1s
Estimated time to prime (subject to change after model architecting): 0d 0h 3m 43s

Time estimate for Quick Clustering:
Estimated time to prime a quick classifier: a few seconds
```

Congratulations! You have successfully run Daimensions measurements!

Because we did not specify a model type, notice that Daimensions selected a clustering network (QC) as the default model for this dataset to minimize the risk of overfitting the data. You can override this using the -f option. “-f NN” for neural networks or “-f QC” for clustering networks.

## Step 3: Build a model

When building models, Daimensions™ offers multiple options:.

Create a python neural network called predict.py:

```
btc -v -v -f NN spectrometer.csv -o predict.py
```

Create a python clustering predictor called predict.py

```
btc -v -v -f QC spectrometer.csv -o predict.py
```

The  “effort” option will increase compute time to lead to better accuracy. (for smaller sets)

```
btc -v -v -f NN spectrometer.csv -o predict.py -e 5
```

The “-e 5” effort will take 5 times as long to run and possibly improve the accuracy. It can be used with QC models and NN models.

For a complete list of all options (all options available will print):

```
btc -v
```

**IMPORTANT! BTC assumes that the last data column is the predictor. If it is not, specify the column name using the -target option:**

```
btc -v -v -f NN spectrometer.csv -o predict.py -target binaryClass
```

For a complete tutorial on using Daimensions, please refers to the Daimensions™ ' User Manual

## Step 4: Run the predictor

Once you have created your predictor, you can use it to validate an existing data set or predict with new data. With the predictor created in the example above, you can validate the existing data spectrometer.csv set by running:

```
python3 predict.py -validate spectrometer.csv
```

Alternatively, if you have a new dataset (without the prediction target column), you can predict the outcome by running:

```
python3 predict.py newdata.csv
```