
dcbench

Data Centric AI

Nov 18, 2021

CONTENTS

1	What is dcbench?	1
2	API Walkthrough	3
2.1	Task	3
2.2	Problem	3
2.3	Artifact	4
2.4	Solution	5
3	Tasks	7
3.1	Minimal Data Selection	7
3.2	Slice Discovery	8
3.3	Data Cleaning on a Budget	9
4	Installing dcbench	11
4.1	Installing from branch	11
4.2	Installing from clone	11
5	Configuring dcbench	13
5.1	Configuring with YAML	13
5.2	Configuring Programmatically	13
6	dcbench package	15
6.1	Subpackages	15
6.2	Submodules	24
6.3	dcbench.constants module	24
6.4	dcbench.version module	24
6.5	Module contents	24
	Python Module Index	31
	Index	33

WHAT IS DCBENCH?

This benchmark evaluates the steps in your machine learning workflow beyond model training and tuning. This includes feature cleaning, slice discovery, and coreset selection. We call these “data-centric” tasks because they’re focused on exploring and manipulating data – not training models. `dcbench` supports a growing number of them:

- *Minimal Data Selection*: Find the smallest subset of training data on which a fixed model architecture achieves accuracy above a threshold.
- *Slice Discovery*: Identify subgroups on which a model underperforms.
- `budgetclean`: Given a fixed budget, clean input features of training data to improve model performance.

`dcbench` includes tasks that look very different from one another: the inputs and outputs of the slice discovery task are not the same as those of the minimal data cleaning task. However, we think it important that researchers and practitioners be able to run evaluations on data-centric tasks across the ML lifecycle without having to learn a bunch of different APIs or rewrite evaluation scripts.

So, `dcbench` is designed to be a common home for these diverse, but related, tasks. In `dcbench` all of these tasks are structured in a similar manner and they are supported by a common Python API that makes it easy to download data, run evaluations, and compare methods.

API WALKTHROUGH

```
pip install dcbench
```

2.1 Task

dcbench supports a diverse set of data-centric tasks (e.g. *Slice Discovery*). You can explore the supported tasks in the documentation (*Tasks*) or via the Python API:

```
In [1]: import dcbench

In [2]: dcbench.tasks
Out[2]:
```

	name	
↪ summary		
minidata	Minimal Data Selection	Given a large training dataset, what is the
↪ sm...		
slice_discovery	Slice Discovery	Machine learnings models that achieve high
↪ ove...		
budgetclean	Data Cleaning on a Budget	When it comes to data preparation, data
↪ clean...		

In the dcbench API, each task is represented by a `dcbench.Task` object that can be accessed by `task_id` (e.g. `dcbench.slice_discovery`). These task objects hold metadata about the task and hold pointers to task-specific `dcbench.Problem` and `dcbench.Solution` subclasses, discussed below.

2.2 Problem

Each task features a collection of *problems* (i.e. instances of the task). For example, the *Slice Discovery* task includes hundreds of problems across a number of different datasets. We can explore a task's problems in dcbench:

```
In [3]: dcbench.tasks["slice_discovery"].problems
Out[3]:
```

	alpha	dataset	...	slice_names	target_name
p_117306	0.0171	imagenet	...	[craft.n.02]	vehicle.n.01
p_117341	0.0171	imagenet	...	[cart.n.01]	vehicle.n.01
p_117406	0.0171	imagenet	...	[rocket.n.01]	vehicle.n.01
p_117634	0.0171	imagenet	...	[barrow.n.03]	vehicle.n.01

(continues on next page)

(continued from previous page)

p_117980	0.0171	imagenet	...	[bicycle.n.01]	vehicle.n.01
p_118007	0.0171	imagenet	...	[wagon.n.01]	vehicle.n.01
p_118045	0.0171	imagenet	...	[motorcycle.n.01]	vehicle.n.01
p_118259	0.0171	imagenet	...	[hat.n.01]	clothing.n.01
p_118311	0.0171	imagenet	...	[shirt.n.01]	clothing.n.01
p_118660	0.0171	imagenet	...	[menu.n.02]	food.n.01
p_118716	0.0171	imagenet	...	[alcohol.n.01]	food.n.01
p_118843	0.0171	imagenet	...	[concoction.n.01]	food.n.01
p_118895	0.0171	imagenet	...	[cup.n.06]	food.n.01
p_118919	0.0171	imagenet	...	[hay.n.01]	food.n.01
p_118949	0.0171	imagenet	...	[punch.n.02]	food.n.01
p_118970	0.0171	imagenet	...	[beverage.n.01]	food.n.01
p_119029	0.0171	imagenet	...	[wine.n.01]	food.n.01
p_119061	0.0171	imagenet	...	[fare.n.04]	food.n.01
p_119075	0.0171	imagenet	...	[feed.n.01]	food.n.01
p_119216	0.0171	imagenet	...	[chime.n.01]	musical_instrument.n.01

[20 rows x 6 columns]

All of a task's problems share the same structure and use the same evaluation scripts. This is specified via task-specific subclasses of `dcbench.Problem` (e.g. `SliceDiscoveryProblem`). The problems themselves are instances of these subclasses. We can access a problem using its id:

```
In [4]: problem = dcbench.tasks["slice_discovery"].problems["p_118919"]
```

```
In [5]: problem
```

```
Out[5]: SliceDiscoveryProblem(artifacts={'activations': 'DataPanelArtifact', 'base_
↳ dataset': 'VisionDatasetArtifact', 'clip': 'DataPanelArtifact', 'model': 'ModelArtifact
↳ ', 'test_predictions': 'DataPanelArtifact', 'test_slices': 'DataPanelArtifact', 'val_
↳ predictions': 'DataPanelArtifact'}, attributes={'alpha': 0.01709975946676697, 'dataset
↳ ': 'imagenet', 'n_pred_slices': 5, 'slice_category': 'rare', 'slice_names': ['hay.n.01
↳ '], 'target_name': 'food.n.01'})
```

2.3 Artifact

Each *problem* is made up of a set of artifacts: a dataset with features to clean, a dataset and a model to perform error analysis on. In `dcbench`, these artifacts are represented by instances of `dcbench.Artifact`. We can think of each *Problem* object as a container for *Artifact* objects.

```
In [6]: problem.artifacts
```

```
Out[6]:
```

```
{'activations': <dcbench.common.artifact.DataPanelArtifact at 0x7fd38c6fc7d0>,
'base_dataset': <dcbench.common.artifact.VisionDatasetArtifact at 0x7fd38c7d3b10>,
'clip': <dcbench.common.artifact.DataPanelArtifact at 0x7fd38c6f44d0>,
'model': <dcbench.common.artifact.ModelArtifact at 0x7fd38c6fc850>,
'test_predictions': <dcbench.common.artifact.DataPanelArtifact at 0x7fd38c6fc890>,
'test_slices': <dcbench.common.artifact.DataPanelArtifact at 0x7fd38c6fc8d0>,
'val_predictions': <dcbench.common.artifact.DataPanelArtifact at 0x7fd38c6fc910>}
```

Note that *Artifact* objects don't actually hold their underlying data in memory. Instead, they hold pointers to where the *Artifact* lives in `dcbench` cloud storage and, if it's been downloaded, where it lives locally on disk. This makes

the *Problem* objects very lightweight.

dcbench includes loading functionality for each artifact type. To load an artifact into memory we can use `load()`. Note that this will also download the artifact to disk if it hasn't yet been downloaded.

```
In [7]: problem.artifacts["model"]  
Out[7]: <dcbench.common.artifact.ModelArtifact at 0x7fd38c6fc850>
```

Easier yet, we can use the index operator directly on *Problem* objects to both fetch the artifact and load it into memory.

```
In [8]: problem["activations"] # shorthand for problem.artifacts["model"].load()  
Out[8]: DataPanel(nrows: 9044, ncols: 3)
```

Downloading to Disk

By default, dcbench downloads artifacts to `~/dcbench` but this can be configured by creating a `dcbench-config.yaml` as described in [Configuring dcbench](#). To download an *Artifact* via the Python API, use `Artifact.download()`. You can also download all the artifacts in a problem with `Problem.download()`.

2.4 Solution

3.1 Minimal Data Selection

Task Details

Task ID minidata

Problems 1

Given a large training dataset, what is the smallest subset you can sample that still achieves some threshold of performance.

Classes: `dcbench.MinidataProblem` `dcbench.MinidataSolution`

Cloud Storage

We recommend downloading Artifacts through the Python API, but you can also explore the Artifacts on the [Google Cloud Console](#).

3.1.1 Problem Artifacts

name	type	description
train_data	<code>dcbench.DataPanelArtifact</code>	A DataPanel of train examples with columns <code>id</code> , <code>input</code> , and <code>target</code> .
val_data	<code>dcbench.DataPanelArtifact</code>	A DataPanel of validation examples with columns <code>id</code> , <code>input</code> , and <code>target</code> .
test_data	<code>dcbench.DataPanelArtifact</code>	A DataPanel of test examples with columns <code>id</code> , <code>input</code> , and <code>target</code> .

3.1.2 Solution Artifacts

name	type	description
train_ids	<code>dcbench.YAMLArtifact</code>	A list of train example ids from the <code>id</code> column of <code>train_data</code> .

3.2 Slice Discovery

Task Details

Task ID `slice_discovery`

Problems 20

Machine learning models that achieve high overall accuracy often make systematic errors on important subgroups (or *slices*) of data. When working with high-dimensional inputs (e.g. images, audio) where data slices are often unlabeled, identifying underperforming slices is challenging. In this task, we'll develop automated slice discovery methods that mine unstructured data for underperforming slices.

Classes: `dcbench.SliceDiscoveryProblem` `dcbench.SliceDiscoverySolution`

Cloud Storage

We recommend downloading Artifacts through the Python API, but you can also explore the Artifacts on the [Google Cloud Console](#).

3.2.1 Problem Artifacts

name	type	description
val_predictions	<code>dcbench.DataPanelArtifact</code>	A <code>DataPanel</code> of the model's predictions with columns <code>id</code> , <code>target</code> , and <code>probs</code> .
test_predictions	<code>dcbench.DataPanelArtifact</code>	A <code>DataPanel</code> of the model's predictions with columns <code>id</code> , <code>target</code> , and <code>probs</code> .
test_slices	<code>dcbench.DataPanelArtifact</code>	A <code>DataPanel</code> of the ground truth slice labels with columns <code>id</code> , <code>slices</code> .
activations	<code>dcbench.DataPanelArtifact</code>	A <code>DataPanel</code> of the model's activations with columns <code>id</code> , <code>act</code> .
model	<code>dcbench.ModelArtifact</code>	A trained PyTorch model to audit.
base_dataset	<code>dcbench.VisionDatasetArtifact</code>	A <code>DataPanel</code> representing the base dataset with columns <code>id</code> and <code>image</code> .
clip	<code>dcbench.DataPanelArtifact</code>	A <code>DataPanel</code> of the image embeddings from OpenAI's CLIP model

3.2.2 Solution Artifacts

name	type	description
pred_slices	<code>dcbench.DataPanelArtifact</code>	A <code>DataPanel</code> of predicted slice labels with columns <code>id</code> and <code>pred_slices</code> .

3.3 Data Cleaning on a Budget

Task Details

Task ID budgetclean

Problems 144

When it comes to data preparation, data cleaning is an essential yet quite costly task. If we are given a fixed cleaning budget, the challenge is to find the training data examples that would bring the biggest positive impact on model performance if we were to clean them.

Classes: `dcbench.BudgetcleanProblem` `dcbench.BudgetcleanSolution`

Cloud Storage

We recommend downloading Artifacts through the Python API, but you can also explore the Artifacts on the [Google Cloud Console](#).

3.3.1 Problem Artifacts

name	type	description
X_train_dirty	<code>dcbench.CSVArtifact</code>	(‘Features of the dirty training dataset which we need to clean. Each dirty cell contains an embedded list of clean candidate values.’)
X_train_clean	<code>dcbench.CSVArtifact</code>	Features of the clean training dataset where each dirty value from the dirty dataset is replaced with the correct clean candidate.
y_train	<code>dcbench.CSVArtifact</code>	Labels of the training dataset.
X_val	<code>dcbench.CSVArtifact</code>	Feature of the validation dataset which can be used to guide the cleaning optimization process.
y_val	<code>dcbench.CSVArtifact</code>	Labels of the validation dataset.
X_test	<code>dcbench.CSVArtifact</code>	(‘Features of the test dataset used to produce the final evaluation score of the model.’)
y_test	<code>dcbench.CSVArtifact</code>	Labels of the test dataset.

3.3.2 Solution Artifacts

name	type	description
idx_selected	<i>dcbench.CSVArtifact</i>	

INSTALLING DCBENCH

This section describes how to install the `dcbench` Python package.

```
pip install dcbench
```

Optional

Some parts of `dcbench` rely on optional dependencies. If you know which optional dependencies you'd like to install, you can do so using something like `pip install dcbench[dev]` instead. See `setup.py` for a full list of optional dependencies.

4.1 Installing from branch

To install from a specific branch use the command below, replacing `main` with the name of any branch in the `dcbench` repository.

```
pip install "dcbench @ git+https://github.com/data-centric-ai/dcbench@main"
```

4.2 Installing from clone

You can install from a clone of the `dcbench` repo with:

```
git clone https://github.com/data-centric-ai/dcbench.git
cd dcbench
pip install -e .
```


CONFIGURING DCBENCH

Several aspects of `dcbench` behavior can be configured by the user. For example, one may wish to change the directory in which `dcbench` downloads artifacts (by default this is `~/dcbench`).

You can see the current state of the `dcbench` configuration with:

```
In [1]: import dcbench

In [2]: dcbench.config
Out[2]: DCBenchConfig(local_dir='/home/docs/.dcbench', public_bucket_name='dcbench',
↳ hidden_bucket_name='dcbench-hidden', celeba_dir='/home/docs/.dcbench/datasets/celeba',
↳ imagenet_dir='/home/docs/.dcbench/datasets/imagenet')
```

5.1 Configuring with YAML

To change the configuration create a YAML file, like the one below:

Then set the environment variable `DCBENCH_CONFIG` to point to the file:

```
export DCBENCH_CONFIG="/path/to/dcbench-config.yaml"
```

If you're using a `conda`, you can permanently set this variable for your environment:

```
conda env config vars set DCBENCH_CONFIG="path/to/dcbench-config.yaml"
conda activate env_name # need to reactivate the environment
```

5.2 Configuring Programmatically

You can also update the config programmatically, though unlike the YAML method above, these changes will not persist beyond the lifetime of your program.

```
dcbench.config.local_dir = "/path/to/storage"
dcbench.config.public_bucket_name = "dcbench-test"
```


DCBENCH PACKAGE

6.1 Subpackages

6.1.1 dcbench.common package

Submodules

dcbench.common.artifact module

class `Artifact`(*artifact_id*, ***kwargs*)

Bases: `abc.ABC`

Parameters `artifact_id` (*str*) –

Return type `None`

DEFAULT_EXT: `str = ''`

download(*force=False*)

Parameters `force` (*bool*) –

classmethod `from_data`(*data*, *artifact_id=None*)

Parameters

- `data` (*Any*) –

- `artifact_id` (*Optional[str]*) –

static `from_yaml`(*loader*, *node*)

Parameters `loader` (*yaml.loader.Loader*) –

property `is_downloaded`: `bool`

property `is_uploaded`: `bool`

isdir: `bool = False`

abstract `load`()

Return type `Any`

property `local_path`: `str`

```
property remote_url: str
```

```
abstract save(data)
```

```
Parameters data (Any) –
```

```
Return type None
```

```
static to_yaml(dumper, data)
```

```
Parameters
```

- *dumper* (*yaml.dumper.Dumper*) –
- *data* (*dcbench.common.artifact.Artifact*) –

```
upload(force=False, bucket=None)
```

```
Parameters
```

- *force* (*bool*) –
- *bucket* (*Optional[storage.Bucket]*) –

```
class ArtifactContainer(id, artifacts, attributes=None)
```

```
Bases: abc.ABC, collections.abc.Mapping, dcbench.common.table.RowMixin
```

```
Parameters
```

- *id* (*str*) –
- *artifacts* (*Mapping[str, Artifact]*) –
- *attributes* (*Mapping[str, BASIC_TYPE]*) –

```
artifact_specs: Mapping[str, ArtifactSpec]
```

```
container_type: str
```

```
download(force=False)
```

```
Parameters force (bool) –
```

```
Return type bool
```

```
classmethod from_artifacts(artifacts, attributes=None, container_id=None)
```

```
Parameters
```

- *artifacts* (*Mapping[str, Artifact]*) –
- *attributes* (*Mapping[str, BASIC_TYPE]*) –
- *container_id* (*str*) –

```
static from_yaml(loader, node)
```

```
Parameters loader (yaml.loader.Loader) –
```

```
property is_downloaded: bool
```

```
property is_uploaded: bool
```

```
task_id: str = 'none'
static to_yaml(dumper, data)
```

Parameters

- **dumper** (`yaml.dumper.Dumper`) –
- **data** (`dcbench.common.artifact.ArtifactContainer`) –

```
upload(force=False, bucket=None)
```

Parameters

- **force** (`bool`) –
- **bucket** (`Optional[storage.Bucket]`) –

```
class ArtifactSpec(description: 'str', artifact_type: 'type')
Bases: object
```

Parameters

- **description** (`str`) –
- **artifact_type** (`type`) –

Return type `None`

artifact_type: `type`

description: `str`

```
class CSVArtifact(artifact_id, **kwargs)
Bases: dcbench.common.artifact.Artifact
```

Parameters **artifact_id** (`str`) –

Return type `None`

DEFAULT_EXT: `str = 'csv'`

load()

Return type `pandas.core.frame.DataFrame`

save(`data`)

Parameters **data** (`pandas.core.frame.DataFrame`) –

Return type `None`

```
class DataPanelArtifact(artifact_id, **kwargs)
Bases: dcbench.common.artifact.Artifact
```

Parameters **artifact_id** (`str`) –

Return type `None`

DEFAULT_EXT: `str = 'mk'`

isdir: `bool = True`

load()

Return type pandas.core.frame.DataFrame

save(*data*)

Parameters *data* (meerkat.datapanel.DataPanel) –

Return type None

class ModelArtifact(*artifact_id*, ***kwargs*)

Bases: [dcbench.common.artifact.Artifact](#)

Parameters *artifact_id* (str) –

Return type None

DEFAULT_EXT: str = 'pt'

load()

Return type dcbench.common.modeling.Model

save(*data*)

Parameters *data* (dcbench.common.modeling.Model) –

Return type None

class VisionDatasetArtifact(*artifact_id*, ***kwargs*)

Bases: [dcbench.common.artifact.DataPanelArtifact](#)

Parameters *artifact_id* (str) –

Return type None

COLUMN_SUBSETS = {'celeba': ['id', 'image', 'identity', 'split'], 'imagenet': ['id', 'image', 'name', 'synset']}

DEFAULT_EXT: str = 'mk'

download(*force=False*)

Parameters *force* (bool) –

classmethod **from_name**(*name*)

Parameters *name* (str) –

isdir: bool = True

class YAMLAartifact(*artifact_id*, ***kwargs*)

Bases: [dcbench.common.artifact.Artifact](#)

Parameters *artifact_id* (str) –

Return type None

DEFAULT_EXT: str = 'yaml'

load()

Return type pandas.core.frame.DataFrame

save(*data*)

Parameters *data* (*Any*) –

Return type None

dcbench.common.download_utils module

This file contains utility functions for downloading datasets. The code in this file is taken from the torchvision package, specifically, <https://github.com/pytorch/vision/blob/master/torchvision/datasets/utils.py>. We package it here to avoid users having to install the rest of torchvision. It is licensed under the following license:

BSD 3-Clause License

Copyright (c) Soumith Chintala 2016, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

calculate_md5(*fpath*, *chunk_size*=1048576)

Parameters

- **fpath** (*str*) –
- **chunk_size** (*int*) –

Return type str

check_integrity(*fpath*, *md5*=None)

Parameters

- **fpath** (*str*) –
- **md5** (*Optional[str]*) –

Return type bool

check_md5(*fpath*, *md5*, ***kwargs*)

Parameters

- **fpath** (*str*) –
- **md5** (*str*) –
- **kwargs** (*Any*) –

Return type `bool`

download_and_extract_archive(*url*, *download_root*, *extract_root=None*, *filename=None*, *md5=None*, *remove_finished=False*, *size=None*)

Parameters

- **url** (*str*) –
- **download_root** (*str*) –
- **extract_root** (*Optional[str]*) –
- **filename** (*Optional[str]*) –
- **md5** (*Optional[str]*) –
- **remove_finished** (*bool*) –
- **size** (*Optional[int]*) –

Return type `None`

download_file_from_google_drive(*file_id*, *root*, *filename=None*, *md5=None*)

Download a Google Drive file from and place it in root.

Parameters

- **file_id** (*str*) – id of file to be downloaded
- **root** (*str*) – Directory to place downloaded file in
- **filename** (*str*, *optional*) – Name to save the file under. If None, use the id of the file.
- **md5** (*str*, *optional*) – MD5 checksum of the download. If None, do not check

download_url(*url*, *root*, *filename=None*, *md5=None*, *size=None*)

Download a file from a url and place it in root.

Parameters

- **url** (*str*) – URL to download file from
- **root** (*str*) – Directory to place downloaded file in
- **filename** (*str*, *optional*) – Name to save the file under. If None, use the basename of the URL
- **md5** (*str*, *optional*) – MD5 checksum of the download. If None, do not check
- **size** (*Optional[int]*) –

Return type `None`

extract_archive(*from_path*, *to_path=None*, *remove_finished=False*)

Parameters

- **from_path** (*str*) –

- **to_path** (*Optional[str]*) –
- **remove_finished** (*bool*) –

Return type *None*

gen_bar_updater (*total*)

Return type *Callable[[int, int, int], None]*

iterable_to_str (*iterable*)

Parameters **iterable** (*Iterable*) –

Return type *str*

list_dir (*root, prefix=False*)

List all directories at a given root.

Parameters

- **root** (*str*) – Path to directory whose folders need to be listed
- **prefix** (*bool, optional*) – If true, prepends the path to each result, otherwise only returns the name of the directories found

Return type *List[str]*

list_files (*root, suffix, prefix=False*)

List all files ending with a suffix at a given root.

Parameters

- **root** (*str*) – Path to directory whose folders need to be listed
- **suffix** (*str or tuple*) – Suffix of the files to match, e.g. `‘.png’` or `(‘.jpg’, ‘.png’)`. It uses the Python `“str.endswith”` method and is passed directly
- **prefix** (*bool, optional*) – If true, prepends the path to each result, otherwise only returns the name of the files found

Return type *List[str]*

verify_str_arg (*value, arg=None, valid_values=None, custom_msg=None*)

Parameters

- **value** (*dcbench.common.download_utils.T*) –
- **arg** (*Optional[str]*) –
- **valid_values** (*Optional[Iterable[dcbench.common.download_utils.T]]*) –
- **custom_msg** (*Optional[str]*) –

Return type *dcbench.common.download_utils.T*

dcbench.common.method module**class Method**(*config=None, **kwargs*)Bases: `abc.ABC`**Parameters** *config* (*dict*) –**class Config**(*n_slices: int = 5, emb_group: str = 'main', emb: str = 'emb', xmodal_emb: str = 'emb'*)Bases: `object`**Parameters**

- *n_slices* (*int*) –
- *emb_group* (*str*) –
- *emb* (*str*) –
- *xmodal_emb* (*str*) –

Return type `None`*emb: str = 'emb'**emb_group: str = 'main'**n_slices: int = 5**xmodal_emb: str = 'emb'**RESOURCES_REQUIRED = {'cpu': 1, 'custom_resources': {'ram_gb': 4}}***dcbench.common.problem module****class Problem**(*id, artifacts, attributes=None*)Bases: `dcbench.common.artifact.ArtifactContainer`**Parameters**

- *id* (*str*) –
- *artifacts* (*Mapping[str, Artifact]*) –
- *attributes* (*Mapping[str, BASIC_TYPE]*) –

*artifact_specs: Mapping[str, ArtifactSpec]**container_type: str = 'problem'**abstract evaluate(solution)***Parameters** *solution* (*Solution*) –*name: str**solution_class: type**summary: str*

dcbench.common.result module

```
class Result
    Bases: object
```

dcbench.common.solution module

```
class Result(source)
    Bases: Mapping
    static load(path)

        Parameters path (str) –
        Return type dcbench.common.solution.Result

    save(path)

        Parameters path (str) –
        Return type None

class Solution(id, artifacts, attributes=None)
    Bases: dcbench.common.artifact.ArtifactContainer
    Parameters
        • id (str) –
        • artifacts (Mapping[str, Artifact]) –
        • attributes (Mapping[str, BASIC_TYPE]) –
    artifact_specs: Mapping[str, ArtifactSpec]
    container_type: str = 'solution'
```

dcbench.common.solve module**dcbench.common.utils module****Module contents****6.1.2 dcbench.tasks package****Subpackages****dcbench.tasks.slice package****Submodules****dcbench.tasks.slice.build_problems module****Module contents**

Module contents

6.2 Submodules

6.3 dcbench.constants module

6.4 dcbench.version module

6.5 Module contents

The dcbench module is a collection for benchmarks that test various aspects of data preparation and handling in the context of AI workflows.

class `Artifact`(*artifact_id*, ***kwargs*)

Bases: `abc.ABC`

Parameters `artifact_id` (*str*) –

Return type `None`

DEFAULT_EXT: `str = ''`

download(*force=False*)

Parameters `force` (*bool*) –

classmethod `from_data`(*data*, *artifact_id=None*)

Parameters

- `data` (*Any*) –

- `artifact_id` (*Optional[str]*) –

static `from_yaml`(*loader*, *node*)

Parameters `loader` (*yaml.loader.Loader*) –

property `is_downloaded`: `bool`

property `is_uploaded`: `bool`

`isdir`: `bool = False`

abstract `load`()

Return type `Any`

property `local_path`: `str`

property `remote_url`: `str`

abstract `save`(*data*)

Parameters `data` (*Any*) –

Return type `None`

`static to_yaml(dumper, data)`

Parameters

- `dumper` (`yaml.Dumper`) –
- `data` (`dcbench.common.artifact.Artifact`) –

`upload(force=False, bucket=None)`

Parameters

- `force` (`bool`) –
- `bucket` (`Optional[storage.Bucket]`) –

`class BudgetcleanProblem(id, artifacts, attributes=None)`

Bases: `dcbench.common.problem.Problem`

Parameters

- `id` (`str`) –
- `artifacts` (`Mapping[str, Artifact]`) –
- `attributes` (`Mapping[str, BASIC_TYPE]`) –

```
artifact_specs: Mapping[str, dcbench.common.artifact.ArtifactSpec] = {'X_test':
ArtifactSpec(description=('Features of the test dataset used to produce the final
evaluation score of the model.'), artifact_type=<class
'dcbench.common.artifact.CSVArtifact'>), 'X_train_clean':
ArtifactSpec(description='Features of the clean training dataset where each dirty
value from the dirty dataset is replaced with the correct clean candidate.',
artifact_type=<class 'dcbench.common.artifact.CSVArtifact'>), 'X_train_dirty':
ArtifactSpec(description=('Features of the dirty training dataset which we need to
clean. Each dirty cell contains an embedded list of clean candidate values.'),
artifact_type=<class 'dcbench.common.artifact.CSVArtifact'>), 'X_val':
ArtifactSpec(description='Feature of the validation dataset which can be used to
guide the cleaning optimization process.', artifact_type=<class
'dcbench.common.artifact.CSVArtifact'>), 'y_test': ArtifactSpec(description='Labels
of the test dataset.', artifact_type=<class 'dcbench.common.artifact.CSVArtifact'>),
'y_train': ArtifactSpec(description='Labels of the training dataset.',
artifact_type=<class 'dcbench.common.artifact.CSVArtifact'>), 'y_val':
ArtifactSpec(description='Labels of the validation dataset.', artifact_type=<class
'dcbench.common.artifact.CSVArtifact'>)}
```

`evaluate(solution)`

Parameters `solution` (`dcbench.tasks.budgetclean.problem.BudgetcleanSolution`)

–

Return type `dcbench.common.result.Result`

`classmethod from_id(scenario_id)`

Parameters `scenario_id` (`str`) –

`classmethod list()`

`solve(idx_selected, **kwargs)`

Parameters

- `idx_selected` (*Any*) –
- `kwargs` (*Any*) –

Return type `dcbench.common.solution.Solution`

`task_id: str = 'budgetclean'`

`class CSVArtifact(artifact_id, **kwargs)`

Bases: `dcbench.common.artifact.Artifact`

Parameters `artifact_id` (*str*) –

Return type `None`

`DEFAULT_EXT: str = 'csv'`

`load()`

Return type `pandas.core.frame.DataFrame`

`save(data)`

Parameters `data` (`pandas.core.frame.DataFrame`) –

Return type `None`

`class DataPanelArtifact(artifact_id, **kwargs)`

Bases: `dcbench.common.artifact.Artifact`

Parameters `artifact_id` (*str*) –

Return type `None`

`DEFAULT_EXT: str = 'mk'`

`isdir: bool = True`

`load()`

Return type `pandas.core.frame.DataFrame`

`save(data)`

Parameters `data` (`meerkat.datapanel.DataPanel`) –

Return type `None`

`class MiniDataProblem(id, artifacts, attributes=None)`

Bases: `dcbench.common.problem.Problem`

Parameters

- `id` (*str*) –
- `artifacts` (`Mapping[str, Artifact]`) –
- `attributes` (`Mapping[str, BASIC_TYPE]`) –

```

artifact_specs: Mapping[str, dcbench.common.artifact.ArtifactSpec] = {'test_data':
ArtifactSpec(description='A DataPanel of test examples with columns ``id``,
``input``, and ``target``.', artifact_type=<class
'dcbench.common.artifact.DataPanelArtifact'>), 'train_data':
ArtifactSpec(description='A DataPanel of train examples with columns ``id``,
``input``, and ``target``.', artifact_type=<class
'dcbench.common.artifact.DataPanelArtifact'>), 'val_data':
ArtifactSpec(description='A DataPanel of validation examples with columns ``id``,
``input``, and ``target``.', artifact_type=<class
'dcbench.common.artifact.DataPanelArtifact'>)}

evaluate(solution)

Parameters solution (dcbench.common.solution.Solution) –
task_id: str = 'minidata'

class ModelArtifact(artifact_id, **kwargs)
Bases: dcbench.common.artifact.Artifact

Parameters artifact_id (str) –
Return type None
DEFAULT_EXT: str = 'pt'
load()

Return type dcbench.common.modeling.Model
save(data)

Parameters data (dcbench.common.modeling.Model) –
Return type None

class Problem(id, artifacts, attributes=None)
Bases: dcbench.common.artifact.ArtifactContainer

Parameters
• id (str) –
• artifacts (Mapping[str, Artifact]) –
• attributes (Mapping[str, BASIC_TYPE]) –
container_type: str = 'problem'
abstract evaluate(solution)

Parameters solution (Solution) –
name: str
solution_class: type
summary: str

class SliceDiscoveryProblem(id, artifacts, attributes=None)
Bases: dcbench.common.problem.Problem

```

Parameters

- `id` (*str*) –
- `artifacts` (*Mapping*[*str*, *Artifact*]) –
- `attributes` (*Mapping*[*str*, *BASIC_TYPE*]) –

```
artifact_specs: Mapping[str, dcbench.common.artifact.ArtifactSpec] = {'activations':
ArtifactSpec(description="A DataPanel of the model's activations with columns
`id`,`act`", artifact_type=<class 'dcbench.common.artifact.DataPanelArtifact'>),
'base_dataset': ArtifactSpec(description="A DataPanel representing the base dataset
with columns `id` and `image`.", artifact_type=<class
'dcbench.common.artifact.VisionDatasetArtifact'>), 'clip':
ArtifactSpec(description="A DataPanel of the image embeddings from OpenAI's CLIP
model", artifact_type=<class 'dcbench.common.artifact.DataPanelArtifact'>), 'model':
ArtifactSpec(description="A trained PyTorch model to audit.", artifact_type=<class
'dcbench.common.artifact.ModelArtifact'>), 'test_predictions':
ArtifactSpec(description="A DataPanel of the model's predictions with columns
`id`,`target`, and `probs.`", artifact_type=<class
'dcbench.common.artifact.DataPanelArtifact'>), 'test_slices':
ArtifactSpec(description="A DataPanel of the ground truth slice labels with columns
`id`, `slices`.", artifact_type=<class
'dcbench.common.artifact.DataPanelArtifact'>), 'val_predictions':
ArtifactSpec(description="A DataPanel of the model's predictions with columns
`id`,`target`, and `probs.`", artifact_type=<class
'dcbench.common.artifact.DataPanelArtifact'>)}

```

`evaluate`(*solution*)

Parameters *solution* (*dcbench.tasks.slice_discovery.problem.SliceDiscoverySolution*) –

Return type dict

`solve`(*pred_slices_dp*)

Parameters *pred_slices_dp* (*meerkat.datapanel.DataPanel*) –

Return type *dcbench.tasks.slice_discovery.problem.SliceDiscoverySolution*

`task_id`: str = 'slice_discovery'

class *Solution*(*id*, *artifacts*, *attributes*=None)

Bases: *dcbench.common.artifact.ArtifactContainer*

Parameters

- `id` (*str*) –
- `artifacts` (*Mapping*[*str*, *Artifact*]) –
- `attributes` (*Mapping*[*str*, *BASIC_TYPE*]) –

`container_type`: str = 'solution'

class *Task*(*task_id*, *name*, *summary*, *problem_class*, *solution_class*, *baselines*=Empty DataFrame Columns: []
Index: [])

Bases: *dcbench.common.table.RowMixin*

Task(task_id: str, name: str, summary: str, problem_class: type, solution_class: type, baselines: dcbench.common.table.Table = Empty DataFrame Columns: [] Index: [])

Parameters

- **task_id** (*str*) –
- **name** (*str*) –
- **summary** (*str*) –
- **problem_class** (*type*) –
- **solution_class** (*type*) –
- **baselines** (*dcbench.common.table.Table*) –

Return type None

baselines: dcbench.common.table.Table = Empty DataFrame Columns: [] Index: []

download_problems(*include_artifacts=False*)

Parameters **include_artifacts** (*bool*) –

property local_problems_path

name: str

problem_class: type

property problems

property problems_path

property remote_problems_url

solution_class: type

summary: str

task_id: str

upload_problems(*include_artifacts=False*)

Parameters **include_artifacts** (*bool*) –

write_problems(*containers*)

Parameters **containers** (*Sequence[dcbench.common.artifact.ArtifactContainer]*)

–

class VisionDatasetArtifact(*artifact_id, **kwargs*)

Bases: dcbench.common.artifact.DataPanelArtifact

Parameters **artifact_id** (*str*) –

Return type None

COLUMN_SUBSETS = {'celeba': ['id', 'image', 'identity', 'split'], 'imagenet': ['id', 'image', 'name', 'synset']}

DEFAULT_EXT: str = 'mk'

download(*force=False*)

Parameters *force* (*bool*) –

classmethod *from_name* (*name*)

Parameters *name* (*str*) –

isdir: *bool* = *True*

class *YAMLArtifact* (*artifact_id*, ***kwargs*)

Bases: *dcbench.common.artifact.Artifact*

Parameters *artifact_id* (*str*) –

Return type *None*

DEFAULT_EXT: *str* = *'yaml'*

load()

Return type *pandas.core.frame.DataFrame*

save (*data*)

Parameters *data* (*Any*) –

Return type *None*

PYTHON MODULE INDEX

d

- `dcbench`, 24
- `dcbench.common`, 23
 - `dcbench.common.artifact`, 15
 - `dcbench.common.download_utils`, 19
 - `dcbench.common.method`, 22
 - `dcbench.common.problem`, 22
 - `dcbench.common.result`, 23
 - `dcbench.common.solution`, 23
 - `dcbench.common.solve`, 23
 - `dcbench.common.utils`, 23
- `dcbench.constants`, 24
- `dcbench.tasks`, 24
- `dcbench.version`, 24

A

Artifact (*class in dcbench*), 24
 Artifact (*class in dcbench.common.artifact*), 15
 artifact_specs (*ArtifactContainer attribute*), 16
 artifact_specs (*BudgetcleanProblem attribute*), 25
 artifact_specs (*MiniDataProblem attribute*), 26
 artifact_specs (*Problem attribute*), 22
 artifact_specs (*SliceDiscoveryProblem attribute*), 28
 artifact_specs (*Solution attribute*), 23
 artifact_type (*ArtifactSpec attribute*), 17
 ArtifactContainer (*class in dcbench.common.artifact*), 16
 ArtifactSpec (*class in dcbench.common.artifact*), 17

B

baselines (*Task attribute*), 29
 BudgetcleanProblem (*class in dcbench*), 25

C

calculate_md5() (*in module dcbench.common.download_utils*), 19
 check_integrity() (*in module dcbench.common.download_utils*), 19
 check_md5() (*in module dcbench.common.download_utils*), 19
 COLUMN_SUBSETS (*VisionDatasetArtifact attribute*), 18, 29
 container_type (*ArtifactContainer attribute*), 16
 container_type (*Problem attribute*), 22, 27
 container_type (*Solution attribute*), 23, 28
 CSVArtifact (*class in dcbench*), 26
 CSVArtifact (*class in dcbench.common.artifact*), 17

D

DataPanelArtifact (*class in dcbench*), 26
 DataPanelArtifact (*class in dcbench.common.artifact*), 17
 dcbench
 module, 24
 dcbench.common
 module, 23
 dcbench.common.artifact

 module, 15
 dcbench.common.download_utils
 module, 19
 dcbench.common.method
 module, 22
 dcbench.common.problem
 module, 22
 dcbench.common.result
 module, 23
 dcbench.common.solution
 module, 23
 dcbench.common.solve
 module, 23
 dcbench.common.utils
 module, 23
 dcbench.constants
 module, 24
 dcbench.tasks
 module, 24
 dcbench.version
 module, 24
 DEFAULT_EXT (*Artifact attribute*), 15, 24
 DEFAULT_EXT (*CSVArtifact attribute*), 17, 26
 DEFAULT_EXT (*DataPanelArtifact attribute*), 17, 26
 DEFAULT_EXT (*ModelArtifact attribute*), 18, 27
 DEFAULT_EXT (*VisionDatasetArtifact attribute*), 18, 29
 DEFAULT_EXT (*YAMLArtifact attribute*), 18, 30
 description (*ArtifactSpec attribute*), 17
 download() (*Artifact method*), 15, 24
 download() (*ArtifactContainer method*), 16
 download() (*VisionDatasetArtifact method*), 18, 29
 download_and_extract_archive() (*in module dcbench.common.download_utils*), 20
 download_file_from_google_drive() (*in module dcbench.common.download_utils*), 20
 download_problems() (*Task method*), 29
 download_url() (*in module dcbench.common.download_utils*), 20

E

emb (*Method.Config attribute*), 22
 emb_group (*Method.Config attribute*), 22

evaluate() (*BudgetcleanProblem* method), 25
 evaluate() (*MiniDataProblem* method), 27
 evaluate() (*Problem* method), 22, 27
 evaluate() (*SliceDiscoveryProblem* method), 28
 extract_archive() (in module
 dcbench.common.download_utils), 20

F

from_artifacts() (*ArtifactContainer* class method),
 16
 from_data() (*Artifact* class method), 15, 24
 from_id() (*BudgetcleanProblem* class method), 25
 from_name() (*VisionDatasetArtifact* class method), 18,
 30
 from_yaml() (*Artifact* static method), 15, 24
 from_yaml() (*ArtifactContainer* static method), 16

G

gen_bar_updater() (in module
 dcbench.common.download_utils), 21

I

is_downloaded (*Artifact* property), 15, 24
 is_downloaded (*ArtifactContainer* property), 16
 is_uploaded (*Artifact* property), 15, 24
 is_uploaded (*ArtifactContainer* property), 16
 isdir (*Artifact* attribute), 15, 24
 isdir (*DataPanelArtifact* attribute), 17, 26
 isdir (*VisionDatasetArtifact* attribute), 18, 30
 iterable_to_str() (in module
 dcbench.common.download_utils), 21

L

list() (*BudgetcleanProblem* class method), 25
 list_dir() (in module
 dcbench.common.download_utils), 21
 list_files() (in module
 dcbench.common.download_utils), 21
 load() (*Artifact* method), 15, 24
 load() (*CSVArtifact* method), 17, 26
 load() (*DataPanelArtifact* method), 17, 26
 load() (*ModelArtifact* method), 18, 27
 load() (*Result* static method), 23
 load() (*YAMLArtifact* method), 18, 30
 local_path (*Artifact* property), 15, 24
 local_problems_path (*Task* property), 29

M

Method (class in *dcbench.common.method*), 22
 Method.Config (class in *dcbench.common.method*), 22
 MiniDataProblem (class in *dcbench*), 26
 ModelArtifact (class in *dcbench*), 27
 ModelArtifact (class in *dcbench.common.artifact*), 18

module

dcbench, 24
dcbench.common, 23
dcbench.common.artifact, 15
dcbench.common.download_utils, 19
dcbench.common.method, 22
dcbench.common.problem, 22
dcbench.common.result, 23
dcbench.common.solution, 23
dcbench.common.solve, 23
dcbench.common.utils, 23
dcbench.constants, 24
dcbench.tasks, 24
dcbench.version, 24

N

n_slices (*Method.Config* attribute), 22
 name (*Problem* attribute), 22, 27
 name (*Task* attribute), 29

P

Problem (class in *dcbench*), 27
Problem (class in *dcbench.common.problem*), 22
 problem_class (*Task* attribute), 29
 problems (*Task* property), 29
 problems_path (*Task* property), 29

R

remote_problems_url (*Task* property), 29
 remote_url (*Artifact* property), 15, 24
 RESOURCES_REQUIRED (*Method* attribute), 22
 Result (class in *dcbench.common.result*), 23
 Result (class in *dcbench.common.solution*), 23

S

save() (*Artifact* method), 16, 24
 save() (*CSVArtifact* method), 17, 26
 save() (*DataPanelArtifact* method), 18, 26
 save() (*ModelArtifact* method), 18, 27
 save() (*Result* method), 23
 save() (*YAMLArtifact* method), 18, 30
 SliceDiscoveryProblem (class in *dcbench*), 27
 Solution (class in *dcbench*), 28
 Solution (class in *dcbench.common.solution*), 23
 solution_class (*Problem* attribute), 22, 27
 solution_class (*Task* attribute), 29
 solve() (*BudgetcleanProblem* method), 26
 solve() (*SliceDiscoveryProblem* method), 28
 summary (*Problem* attribute), 22, 27
 summary (*Task* attribute), 29

T

Task (class in *dcbench*), 28

`task_id` (*ArtifactContainer* attribute), 16
`task_id` (*BudgetcleanProblem* attribute), 26
`task_id` (*MiniDataProblem* attribute), 27
`task_id` (*SliceDiscoveryProblem* attribute), 28
`task_id` (*Task* attribute), 29
`to_yaml()` (*Artifact* static method), 16, 25
`to_yaml()` (*ArtifactContainer* static method), 17

U

`upload()` (*Artifact* method), 16, 25
`upload()` (*ArtifactContainer* method), 17
`upload_problems()` (*Task* method), 29

V

`verify_str_arg()` (in module
 dcbench.common.download_utils), 21
VisionDatasetArtifact (class in *dcbench*), 29
VisionDatasetArtifact (class in
 dcbench.common.artifact), 18

W

`write_problems()` (*Task* method), 29

X

`xmodal_emb` (*Method.Config* attribute), 22

Y

YAMLArtifact (class in *dcbench*), 30
YAMLArtifact (class in *dcbench.common.artifact*), 18