
SKA Science Data Challenge Scoring

Sep 08, 2021

1	sdcl	3
1.1	The Sdc1Scorer class	4
1.2	The Sdc1Score class	4
2	sdcl2	7
2.1	The Sdc2Scorer class	8
2.2	The Sdc2Score class	8
2.3	Scoring pipeline	9
3	Indices and tables	13
	Python Module Index	15
	Index	17

This package is an open-source implementation of the code used to score and rank the submissions for the SKA Science Data Challenges (SDC).

CHAPTER 1

sdcl

The original IDL code is available at: <https://astronomers.skatelescope.org/ska-science-data-challenge-1/>

To score a submission for SDC1, one should first instantiate a Scorer. This can be done via two methods depending on the format of the input data.

If your input catalogues are in text format, one should use the class method: `ska_sdc.sdcl.sdcl_scorer.SdclScorer.from_txt()`. For example:

```
from ska_sdc.sdcl import sdcl_scorer

sub_cat_path = "/path/to/submission/catalogue.txt"
truth_cat_path = "/path/to/truth/catalogue.txt"

scorer = sdcl_scorer.from_txt(sub_cat_path, truth_cat_path, freq=1400)
```

However, if your input catalogues are already dataframes, one should instantiate the constructor for `ska_sdc.sdcl.sdcl_scorer.SdclScorer` class directly:

```
from ska_sdc.sdcl import sdcl_scorer

scorer = sdcl_scorer(df1, df2, freq=1400)
```

where `df1` and `df2` are dataframes.

When the class has been instantiated, the `ska_sdc.sdcl.sdcl_scorer.SdclScorer.run()` method can be called to run the scoring pipeline:

```
result = scorer.run()
```

which returns an instance of the Score class `ska_sdc.sdcl.models.sdcl_score.SdclScore` containing all the details related to the run.

1.1 The Sdc1Scorer class

class `ska_sdc.sdc1.sdc1_scorer.Sdc1Scorer` (*sub_df, truth_df, freq*)

The SDC1 scorer class.

Parameters

- **sub_df** (`pandas.DataFrame`) – The submission catalogue DataFrame of detected sources and properties
- **truth_path** (`pandas.DataFrame`) – The truth catalogue DataFrame
- **freq** (`int`) – Image frequency band (560, 1400 or 9200 MHz)

classmethod `from_txt` (*sub_path, truth_path, freq, sub_skiprows=1, truth_skiprows=0*)

Create an SDC1 scorer class from two source catalogues in text format.

Parameters

- **sub_path** (`str`) – The path of the submission catalogue of detected sources and properties
- **truth_path** (`str`) – The path of the truth catalogue
- **freq** (`int`) – Image frequency band (560, 1400 or 9200 MHz)
- **sub_skiprows** (`int`, optional) – Number of rows to skip in submission catalogue. Defaults to 1.
- **truth_skiprows** (`int`, optional) – Number of rows to skip in truth catalogue. Defaults to 0.

run (*mode=0, train=False, detail=False*)

Run the scoring pipeline.

Parameters

- **mode** (`int`, optional) – 0 or 1 to use core or centroid positions for scoring
- **train** (`bool`, optional) – If True, will only evaluate score based on training area, else will exclude training area
- **detail** (`bool`, optional) – If True, will return the catalogue of matches and per source scores.

Returns

The calculated SDC1 score object

Return type `ska_sdc.sdc1.models.sdc1_score.Sdc1Score`

score

Get the resulting Sdc1Score object.

Returns The calculated SDC1 score object

Return type `ska_sdc.sdc1.models.sdc1_score.Sdc1Score`

1.2 The Sdc1Score class

class `ska_sdc.sdc1.models.sdc1_score.Sdc1Score` (*mode=0, train=False, detail=False*)

Simple data container class for collating data relating to an SDC1 score.

This is created by the SDC1 Scorer's run method.

acc_pc

The average score per match (%).

Returns float64

detail

If True, has returned the catalogue of matches and per source scores.

Returns bool

match_df

Dataframe of matched sources.

Returns pandas.DataFrame

mode

The position used for scoring (0==core, 1==centroid)

Returns int

n_bad

Number of candidate matches rejected during data cleansing.

Returns int

n_det

The total number of detected sources in the submission.

Returns int

n_false

Number of false detections.

Returns int

n_match

Number of candidate matches below threshold.

Returns int

score_det

The sum of the scores.

Returns float64

scores_df

Dataframe containing the scores.

Returns pandas.DataFrame

train

If True, has evaluated score based on training area, else excludes training area.

Returns bool

value

The score for the last run.

Returns float64

CHAPTER 2

sdc2

This is a skeleton framework for SDC2.

To score a submission for SDC2, one should first instantiate a Scorer. This can be done via two methods depending on the format of the input data.

If your input catalogues are in text format, one should use the class method: `ska_sdc.sdc2.sdc2_scorer.Sdc2Scorer.from_txt()`. For example:

```
from ska_sdc.sdc2 import sdc2_scorer

sub_cat_path = "/path/to/submission/catalogue.txt"
truth_cat_path = "/path/to/truth/catalogue.txt"

scorer = sdc2_scorer.from_txt(sub_cat_path, truth_cat_path)
```

However, if your input catalogues are already dataframes, one should instantiate the constructor for `ska_sdc.sdc2.sdc2_scorer.Sdc2Scorer` class directly:

```
from ska_sdc.sdc2 import sdc2_scorer

scorer = sdc2_scorer(df1, df2)
```

where `df1` and `df2` are dataframes.

When the class has been instantiated, the `ska_sdc.sdc2.sdc2_scorer.Sdc2Scorer.run()` method can be called to run the scoring pipeline:

```
result = scorer.run()
```

which returns an instance of the Score class `ska_sdc.sdc2.models.sdc2_score.Sdc2Score` containing all the details related to the run.

2.1 The Sdc2Scorer class

class `ska_sdc.sdc2.sdc2_scorer.Sdc2Scorer` (*cat_sub, cat_truth*)

The SDC2 scorer class.

Parameters

- **cat_sub** (`pandas.DataFrame`) – The submission catalogue.
- **cat_truth** (`pandas.DataFrame`) – The truth catalogue.

classmethod `from_txt` (*sub_path, truth_path, sub_skiprows=0, truth_skiprows=0*)

Create an SDC2 scorer class from two source catalogues in text format.

The catalogues must have a header row of column names that matches the expected column names in the config file.

Parameters

- **sub_path** (`str`) – Path to the submission catalogue.
- **truth_path** (`str`) – Path to the truth catalogue.
- **sub_skiprows** (`int`, optional) – Number of rows to skip in submission catalogue. Defaults to 0.
- **truth_skiprows** (`int`, optional) – Number of rows to skip in truth catalogue. Defaults to 0.

run (*train=False, detail=False*)

Run the scoring pipeline.

Returns The calculated SDC2 score object

Return type `ska_sdc.sdc2.models.sdc2_score.Sdc2Score`

score

Get the resulting Sdc2Score object.

Returns The calculated SDC2 score object

Return type `ska_sdc.sdc2.models.sdc2_score.Sdc2Score`

2.2 The Sdc2Score class

class `ska_sdc.sdc2.models.sdc2_score.Sdc2Score` (*train=False, detail=False*)

Simple data container class for collating data relating to an SDC2 score.

This is created by the SDC2 Scorer's run method.

acc_pc

The average score per match (%).

Returns `float64`

detail

If True, has returned the catalogue of matches and per source scores.

Returns `bool`

match_df

Dataframe of matched sources.

Returns `pandas.DataFrame`

n_bad
Number of candidate matches rejected during data cleansing.

Returns `int`

n_det
The total number of detected sources in the submission.

Returns `int`

n_false
Number of false detections.

Returns `int`

n_match
Number of candidate matches below threshold.

Returns `int`

score_det
The sum of the scores.

Returns `float64`

scores_df
Dataframe containing the scores.

Returns `pandas.DataFrame`

train
If True, has evaluated score based on training area, else excludes training area.

Returns `bool`

value
The score for the last run.

Returns `float64`

2.3 Scoring pipeline

The SDC scoring pipeline proceeds sequentially via four steps:

2.3.1 Crossmatch preprocessing

```
class ska_sdc.sdc2.utils.xmatch_preprocessing.XMatchPreprocessing(step_names=[])
    Prepare catalogues for crossmatching.

    __init__(step_names=[])

        Parameters step_names (list) – Name of the steps to be imported from ska_sdc.sdc2.utils.xmatch_preprocessing_steps

    preprocess (*args, **kwargs)

        A wrapper function used to sequentially call all other prerequisite crossmatching preprocessing functions.

        Returns Preprocessed catalogue.

        Return type pandas.DataFrame
```

Crossmatch preprocessing steps

```
class ska_sdc.sdc2.utils.xmatch_preprocessing_steps.XMatchPreprocessingStepStub(*args,
                                                                              **kwargs)
    Stub class for a preprocessing step.

    __init__(*args, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

    execute()
        Execute the step.

        Returns Processed catalogue.
        Return type pandas.DataFrame
```

2.3.2 Catalogue crossmatching

2.3.3 Crossmatch postprocessing

```
class ska_sdc.sdc2.utils.xmatch_postprocessing.XMatchPostprocessing(step_names=[])
    Postprocess crossmatched catalogue.

    __init__(step_names=[])
        Parameters step_names (list) – Name of the steps to be imported from ska_sdc.sdc2.utils.xmatch_postprocessing_steps

    postprocess(*args, **kwargs)
        A wrapper function used to sequentially call all other prerequisite crossmatching postprocessing functions.

        Returns Postprocessed catalogue.
        Return type pandas.DataFrame
```

Crossmatch postprocessing steps

```
class ska_sdc.sdc2.utils.xmatch_postprocessing_steps.XMatchPostprocessingStepStub(*args,
                                                                              **kwargs)
    Stub class for a postprocessing step.

    __init__(*args, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

    execute()
        Execute the step.

        Returns Processed catalogue.
        Return type pandas.DataFrame
```

2.3.4 Score computation

```
ska_sdc.sdc2.utils.create_score.create_sdc_score(config, sieved_sub_df, n_det, train,
                                                detail)
    Complete the scoring pipeline using the data generated by the previous steps. This requires the prepared truth and submission catalogues, and the candidate match catalogues created from the crossmatch step.
```

Parameters

- **sieved_sub_df** (`pandas.DataFrame`) – The processed and sieved candidate match catalogue between submission and truth.
- **n_det** (`int`) – Total number of detected sources.
- **train** (`bool`) – Whether the score is determined based on training area only
- **detail** (`bool`) – If True, will include the detailed score and match data with the returned `Sdc2Score` object.

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

S

- `ska_sdc.sdc1.models.sdc1_score`, 4
- `ska_sdc.sdc1.sdc1_scorer`, 4
- `ska_sdc.sdc2.models.sdc2_score`, 8
- `ska_sdc.sdc2.sdc2_scorer`, 8
- `ska_sdc.sdc2.utils.create_score`, 10
- `ska_sdc.sdc2.utils.xmatch_postprocessing`,
10
- `ska_sdc.sdc2.utils.xmatch_postprocessing_steps`,
10
- `ska_sdc.sdc2.utils.xmatch_preprocessing`,
9
- `ska_sdc.sdc2.utils.xmatch_preprocessing_steps`,
10

Symbols

`__init__()` (*ska_sdc.sdc2.utils.xmatch_postprocessing.XMatchPostprocessingStepStub* *method*), 10

`__init__()` (*ska_sdc.sdc2.utils.xmatch_postprocessing_steps.XMatchPostprocessingStepsStub* *method*), 10

`__init__()` (*ska_sdc.sdc2.utils.xmatch_preprocessing.XMatchPreprocessingStepStub* *method*), 9

`__init__()` (*ska_sdc.sdc2.utils.xmatch_preprocessing_steps.XMatchPreprocessingStepsStub* *method*), 10

A

`acc_pc` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

`acc_pc` (*ska_sdc.sdc2.models.sdc2_score.Sdc2Score* *attribute*), 8

C

`create_sdc_score()` (in *module* *ska_sdc.sdc2.utils.create_score*), 10

D

`detail` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

`detail` (*ska_sdc.sdc2.models.sdc2_score.Sdc2Score* *attribute*), 8

E

`execute()` (*ska_sdc.sdc2.utils.xmatch_postprocessing_steps.XMatchPostprocessingStepStub* *method*), 10

`execute()` (*ska_sdc.sdc2.utils.xmatch_preprocessing_steps.XMatchPreprocessingStepStub* *method*), 10

F

`from_txt()` (*ska_sdc.sdc1.sdc1_scorer.Sdc1Scorer* *class method*), 4

`from_txt()` (*ska_sdc.sdc2.sdc2_scorer.Sdc2Scorer* *class method*), 8

M

`match_postprocess()` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

`match_postprocess()` (*ska_sdc.sdc2.models.sdc2_score.Sdc2Score* *attribute*), 8

`match_preprocess()` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

N

`n_bad` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

`n_bad` (*ska_sdc.sdc2.models.sdc2_score.Sdc2Score* *attribute*), 9

`n_det` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

`n_det` (*ska_sdc.sdc2.models.sdc2_score.Sdc2Score* *attribute*), 9

`n_false` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

`n_false` (*ska_sdc.sdc2.models.sdc2_score.Sdc2Score* *attribute*), 9

`n_match` (*ska_sdc.sdc1.models.sdc1_score.Sdc1Score* *attribute*), 5

`n_match` (*ska_sdc.sdc2.models.sdc2_score.Sdc2Score* *attribute*), 9

P

`postprocess()` (*ska_sdc.sdc2.utils.xmatch_postprocessing.XMatchPostprocessingStepStub* *method*), 10

`preprocess()` (*ska_sdc.sdc2.utils.xmatch_preprocessing.XMatchPreprocessingStepStub* *method*), 9

R

`run()` (*ska_sdc.sdc1.sdc1_scorer.Sdc1Scorer* *method*), 4

`run()` (*ska_sdc.sdc2.sdc2_scorer.Sdc2Scorer* *method*), 8

S

`score (ska_sdc.sdc1.sdc1_scorer.Sdc1Scorer attribute),`
4
`score (ska_sdc.sdc2.sdc2_scorer.Sdc2Scorer attribute),`
8
`score_det (ska_sdc.sdc1.models.sdc1_score.Sdc1Score`
`attribute),` 5
`score_det (ska_sdc.sdc2.models.sdc2_score.Sdc2Score`
`attribute),` 9
`scores_df (ska_sdc.sdc1.models.sdc1_score.Sdc1Score`
`attribute),` 5
`scores_df (ska_sdc.sdc2.models.sdc2_score.Sdc2Score`
`attribute),` 9
`Sdc1Score (class in ska_sdc.sdc1.models.sdc1_score),`
4
`Sdc1Scorer (class in ska_sdc.sdc1.sdc1_scorer),` 4
`Sdc2Score (class in ska_sdc.sdc2.models.sdc2_score),`
8
`Sdc2Scorer (class in ska_sdc.sdc2.sdc2_scorer),` 8
`ska_sdc.sdc1.models.sdc1_score (module),` 4
`ska_sdc.sdc1.sdc1_scorer (module),` 4
`ska_sdc.sdc2.models.sdc2_score (module),` 8
`ska_sdc.sdc2.sdc2_scorer (module),` 8
`ska_sdc.sdc2.utils.create_score (module),`
10
`ska_sdc.sdc2.utils.xmatch_postprocessing`
`(module),` 10
`ska_sdc.sdc2.utils.xmatch_postprocessing_steps`
`(module),` 10
`ska_sdc.sdc2.utils.xmatch_preprocessing`
`(module),` 9
`ska_sdc.sdc2.utils.xmatch_preprocessing_steps`
`(module),` 10

T

`train (ska_sdc.sdc1.models.sdc1_score.Sdc1Score at-`
`tribute),` 5
`train (ska_sdc.sdc2.models.sdc2_score.Sdc2Score at-`
`tribute),` 9

V

`value (ska_sdc.sdc1.models.sdc1_score.Sdc1Score at-`
`tribute),` 5
`value (ska_sdc.sdc2.models.sdc2_score.Sdc2Score at-`
`tribute),` 9

X

`XMatchPostprocessing (class in`
`ska_sdc.sdc2.utils.xmatch_postprocessing),`
10
`XMatchPostprocessingStepStub (class in`
`ska_sdc.sdc2.utils.xmatch_postprocessing_steps),`
10