STAC

Statistical Tests for Algorithms Comparison (STAC) is a new platform for statistical analysis to verify the results obtained from computational intelligence algorithms.

STAC consists of three different layers for performing statistical tests: a Python library, a set of web services and a web client. Through this platform you can verify the results obtained from the learning algorithms applying the statistic tests to the experiments, which, among other uses, support the decision making process (the election of the most suitable algorithm, for example).

Python library

The first layer of STAC is an implementation in Python of several statistical tests. Only those tests not currently implemented in the scipy.stats module are included in this library. Moreover, statistical tests for multiple comparison (ANOVA, Friedman, etc.) are reimplemented in STAC. This is due to the lack of information returned by the corresponding implementation in scipy.stats, which is needed in order to do the post-hoc analysis.

More info about the library can be found in thesphinx documentation.

Web Services API

In order to make accessible the use of the statistical tests to several programming languages, a list of web services are provided within STAC. These services are both for the tests of the Python library, and those implemented in scipy.stats.

The web services are implemented following the REST architectural style: * Each service is fully described by its URL. Only the data sample values are sent through the body of the request. This is because it is impractical to send variable * The POST HTTP operation is used for the request message. This is due to the need of sending data through the body. * The data of the request and the response are codified using an Internet media type, in this case JSON.

More info about the API can be found in theswagger documentation.

Web Client

The last layer is a web-based front-end of the STAC web services API. This web client is designed to facilitate the full process of statistical analysis for those users without any knowledge of the web services or the Python library. Moreover, this platform contains an assistant to decide the best suitable test for the data provided by the user, and a continuous help in each step of the process.

You can access the web client in the CITIUS software tool page.

Project Structure

The following directory tree shows the most important files with a short description of what they do:

```
STAC
|-- README.md
|-- License
|-- stac/                    # Python library sources
|   |-- parametric.py
|   |-- nonparametric.py
|   |-- unit_tests.py
|   `-- doc/                 # Sphinx documentation
|-- api/                     # Rest services sources
|   |-- app.wsgi             # To load from apache
|   |-- services.py          # Bottle implementation
|   |-- utils.py
|   `-- apidoc/              # Swagger documentation
|-- web/                     # Rest services sources
|   |-- header.html          # Header of all pages
|   |-- topbar.html          # Top menu
|   |-- index.html           # Index page
|   |-- modals.html          # Modal elements
|   |-- help.html            # Help content
|   |-- data.html            # File content table
|   |-- \*.html              # The other html files are the test forms
|   |-- css/                 # Style files
|   |-- fonts/               # Font files
|   |-- img/                 # Image files
|   `-- js/                  # Javascript sources
|       |-- lib/             # Javascript external libraries
|       |-- loader.js        # Loads scripts and styles dynamically
|       |-- config.js        # Configurations of the website
|       |-- layout.js        # Dynamic layout loader
|       |-- behavior.js      # Event handlers
|       |-- file_manager.js  # File related event handlers
|       |-- export.js        # Functions to export tables to CSV and Latex
|       `-- tests.js         # Functions to call the STAC API
`-- docker/                  # Docker image resources
      |-- Dockerfile         # Instructions to build the stac image
      `-- stac.conf          # Apache site configuration
```

Build and deployment

The project can be built into adocker image using only the docker folder of the project. To install docker, please follow the instructions here.

To build a docker image, simple execute the following command:

```
docker build -t stac docker
```

Then to deploy the docker image:

```
docker run -d -p 80:80 stac
```

Citation

This library was presented in the "IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2015". If you use this code in your research projects, we encourage you to please cite our work:

> I. Rodríguez-Fdez, A. Canosa, M. Mucientes, A. Bugarín, STAC: a web platform for the comparison of algorithms using statistical tests, in:Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2015.

```
@InProceedings{rodriguez-fdez2015stac,
   author={Ismael Rodr\'{i}guez-Fdez and Adri\'{a}n Canosa and Manuel Mucientes and Alberto Bugar\'{i}n},
   title={{STAC}: a web platform for the comparison of algorithms using statistical tests},
   booktitle = {Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)},
   year={2015}
}
```

INFORMACIÓN

Investigadores
Manuel Mucientes Molina
Alberto Bugarín Diz
Ismael Rodríguez Fernández
Adrián Canosa Mouzo

Licenza

DESCARGAR

Repositorio Gitlab

Descargar de Gitlab

PUBLICACIÓNS

STAC: a web platform for the comparison of algorithms using statistical tests
2015 IEEE International Conference on Fuzzy Systems, 2015

DEMOSTRADORES

Statistical Tests for Algorithms Comparison