anomalydetection Documentation

Release 0.0.0.dev1

Cristòfol Torrens <piffall@gmail.com>

Jul 11, 2018

Content

1 Status **3**

2 Overview **5**

3 Content **7**

3.1 Features . . . 7
3.1.1 Project . . . 8
3.1.1.1 Contributors . . . 8
3.1.1.2 Resources/Links . . . 8
3.1.1.3 Roadmap . . . 8
3.1.2 License . . . 8
3.1.3 Getting started . . . 20
3.1.3.1 Install . . . 20
3.1.3.2 Sandbox . . . 20
3.1.3.3 Devel mode . . . 20
3.1.4 Configuration . . . 21
3.1.4.1 streams . . . 21
3.1.4.2 websocket . . . 23
3.1.4.3 Example configuration file . . . 23
3.1.5 Deploy . . . 24
3.1.5.1 Backend . . . 24
3.1.5.2 Dashboard . . . 25
3.1.5.3 Embedded . . . 25
3.1.6 Plugins . . . 25
3.1.6.1 Interface . . . 25
3.1.6.2 Example . . . 26
3.1.6.3 Use in anomdec.yml . . . 26
3.1.7 API Reference . . . 27
3.1.7.1 anomalydetection . . . 27
3.1.7.2 anomalydetection.common . . . 27
3.1.7.3 anomalydetection.backend . . . 27

4 Indices and tables **47**

Python Module Index **49**

Warning: Anomaly Detection Framework is under an early development phase. API, architecture and implementations could suffer important changes. For this, Bluekiri do not offer any type of warranty. Use at your **own responsibility**.

CHAPTER 1

Status

CHAPTER 2

Overview

This project born from the need of detect anomalies on multiple and completely different signals, and react to it rapidly. To achieve this, Bluekiri decided to implement its own system to manage multiple signals at the same time in a easy and scalable way.

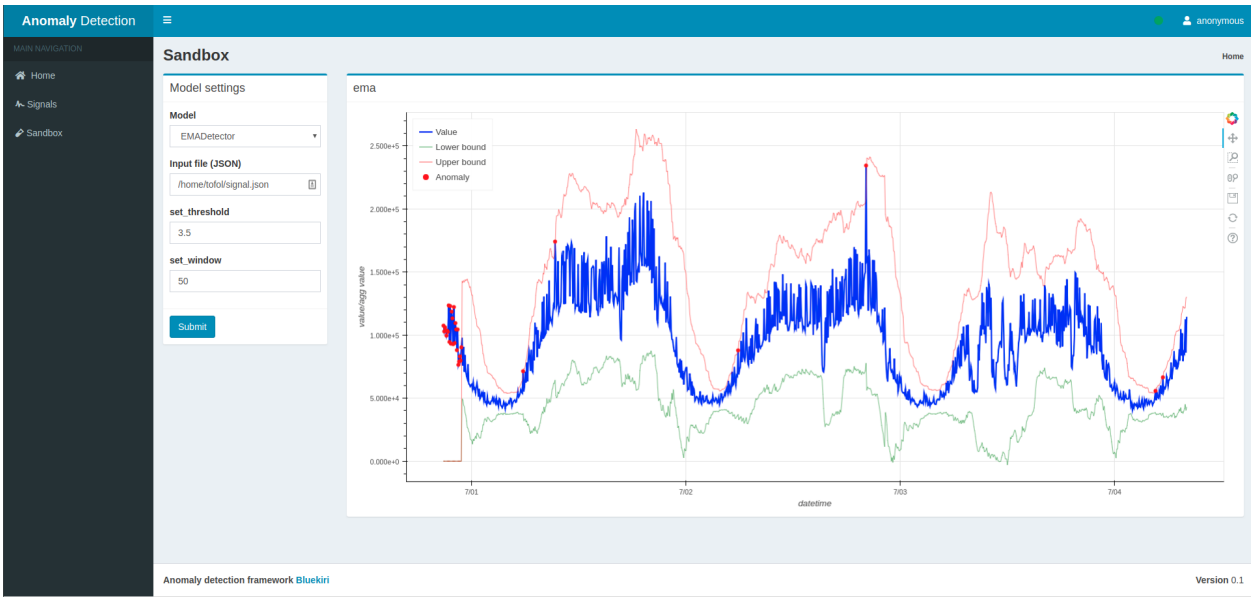
This project is not focused on Machine Learning models, but in an effective Framework to put those models in production.

CHAPTER 3

Content

3.1 Features

- It has an abstraction layer to implement engines, streaming sources and sinks, repository sources and sinks.
- It supports Kafka and PubSub by default.
- It has a default implementation for that supports tumbling window aggregation on Spark, on Kafka and PubSub sources using an specific JSON schema messages.
- Configuration file in YAML format.
- It also includes a dashboard to visualize the signal anomalies and play with signals in a sandbox to try models, tune parameters an see which parameters fits better into your signal.



Note: The Sandbox is limited to use the included message handlers and models only. Custom models will be available to use after we implement the plugin system.

3.1.1 Project

3.1.1.1 Contributors

3.1.1.2 Resources/Links

TODO

3.1.1.3 Roadmap

TODO

3.1.2 License

```
                    GNU AFFERO GENERAL PUBLIC LICENSE
                       Version 3, 19 November 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU Affero General Public License is a free, copyleft license for
software and other kinds of works, specifically designed to ensure
cooperation with the community in the case of network server software.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
our General Public Licenses are intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  Developers that use our General Public Licenses protect your rights
with two steps: (1) assert copyright on the software, and (2) offer
you this License which gives you legal permission to copy, distribute
and/or modify the software.

  A secondary benefit of defending all users' freedom is that
improvements made in alternate versions of the program, if they
receive widespread use, become available for other developers to
```

(continues on next page)

(continued from previous page)

```
incorporate.  Many developers of free software are heartened and
encouraged by the resulting cooperation.  However, in the case of
software used on network servers, this result may fail to come about.
The GNU General Public License permits making a modified version and
letting the public access it on a server without ever releasing its
source code to the public.

  The GNU Affero General Public License is designed specifically to
ensure that, in such cases, the modified source code becomes available
to the community.  It requires the operator of a network server to
provide the source code of the modified version running there to the
users of that server.  Therefore, public use of a modified version, on
a publicly accessible server, gives the public access to the source
code of the modified version.

  An older license, called the Affero General Public License and
published by Affero, was designed to accomplish similar goals.  This is
a different license, not a version of the Affero GPL, but Affero has
released a new version of the Affero GPL which permits relicensing under
this license.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU Affero General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
```

(continues on next page)

(continued from previous page)

```
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
```

(continues on next page)

(continued from previous page)

```
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
```

(continues on next page)

(continued from previous page)

```
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.
```

(continues on next page)

(continued from previous page)

```
    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
```

(continues on next page)

(continued from previous page)

```
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
```

(continues on next page)

(continued from previous page)

```
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.
```

(continues on next page)

(continued from previous page)

```
  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
```

(continues on next page)

(continued from previous page)

```
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Remote Network Interaction; Use with the GNU General Public License.

  Notwithstanding any other provision of this License, if you modify the
Program, your modified version must prominently offer all users
interacting with it remotely through a computer network (if your version
supports such interaction) an opportunity to receive the Corresponding
Source of your version by providing access to the Corresponding Source
from a network server at no charge, through some standard or customary
```

(continues on next page)

(continued from previous page)

```
means of facilitating copying of software.  This Corresponding Source
shall include the Corresponding Source for any work covered by version 3
of the GNU General Public License that is incorporated pursuant to the
following paragraph.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the work with which it is combined will remain governed by version
3 of the GNU General Public License.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU Affero General Public License from time to time.  Such new versions
will be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU Affero General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU Affero General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU Affero General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
```

(continues on next page)

(continued from previous page)

```
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If your software can interact with users remotely through a computer
network, you should also make sure that it provides a way for users to
get its source.  For example, if your program is a web application, its
interface could display a "Source" link that leads users to an archive
of the code.  There are many ways you could offer source, and different
solutions will be better for different programs; see section 13 for the
specific requirements.

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU AGPL, see
<https://www.gnu.org/licenses/>.
```

3.1.3 Getting started

3.1.3.1 Install

The installation is quick and simple using pip.

```
# Anomaly Detection Framework needs a home directory to read settings from.
# In case is not provided, $HOME/anomdec will be used
export ANOMDEC_HOME=~/anomdec

# Install from pypi using pip
pip install anomalydetection
```

3.1.3.2 Sandbox

After installation, you can start a dashboard instance and play with the sandbox, there is no configuration required to do this.

```
# Start a dashboard to play with sandbox
anomdec dashboard
```

You will get an output similar to this one.

```
2018-07-09 12:37:51,930 - anomalydetection.anomdec.Anomdec:35 - INFO - Starting␣
↪anomdec
2018-07-09 12:37:51,930 - anomalydetection.anomdec.Anomdec:38 - INFO - Run dashboard
2018-07-09 12:37:51,933 - anomalydetection.common.config.Config:50 - WARNING - Cannot␣
↪load configuration.
[Errno 2] No such file or directory: '~/anomdec/anomdec.yml'
```

Note: Ignore the Config `WARNING` log if you are following this *Getting started* guide. This is because we have not configured any signal to process yet.

3.1.3.3 Devel mode

You can also run a devel mode. For this, you need to deploy a docker-compose.yml file. This will deploy an Apache Kafka, Apache Zookeeper and a Google PubSub emulator. The *devel mode* processes a random signals that are randomly generated by itself.

```
# Export kafka advertise hostname
export KAFKA_ADVERTISED_HOST_NAME=localhost

# Download docker-compose.yml file
curl -o docker-compose.yml https://raw.githubusercontent.com/bluekiri/
↪anomalydetection/master/docker-compose.yml

# Start docker enviroment
docker-compose up

# In another terminal, start the devel mode
anomdec devel
```

You should see how messages are pushed to Kafka and PubSub backends to `stdout`.

```
- anomalydetection.anomdec.Anomdec:35 - INFO - Starting anomdec
- anomalydetection.anomdec.Anomdec:53 - INFO - Creating configuration for DEVEL MODE
- anomalydetection.anomdec.Anomdec:57 - INFO - Run dashboard, backend and producer
...
- anomalydetection.backend.stream.pubsub.PubSubStreamConsumer:70 - DEBUG - Message␣
↪received: {"application": "devel0", "ts": "2018-07-09 13:00:29.886520", "value": 1}
- anomalydetection.backend.stream.pubsub.PubSubStreamConsumer:70 - DEBUG - Message␣
↪received: {"application": "devel2", "ts": "2018-07-09 13:00:29.905861", "value": 3}
- anomalydetection.backend.stream.pubsub.PubSubStreamConsumer:70 - DEBUG - Message␣
↪received: {"application": "devel1", "ts": "2018-07-09 13:00:29.902405", "value": 2}
```

Open the dashboard on your browser to visualize how data is processed in real time.

3.1.4 Configuration

Configuration is statically defined in a YAML file. The application reads `anomdec.yml` from `$ANOMDEC_HOME` path. The following section describes this file structure.

3.1.4.1 streams

`Streams` is the main section of this file. It's a named list that defines how is a signal processed.

source / engine

It has two required sections, the `source` and the `engine` This is the minimal configuration to start processing signals but, at this point we are not persisting the result.

```
1  version: 1
2
3  streams:
4    - name: my_kafka_source_one
5      source:
6        type: kafka
7        params:
8          broker_servers: localhost:9092
9          input_topic: test1
10     engine:
11       type: robust
12       params:
13         window: 30
14         threshold: 0.9999
```

sink

To persist the result we need to add a `sink` configuration section. This can be a list of *sinks*.

```
1  version: 1
2
3  streams:
4    - name: my_kafka_source_one
5      source:
```

(continues on next page)

(continued from previous page)

```
      type: kafka
      params:
        broker_servers: localhost:9092
        input_topic: test1
    engine:
      type: robust
      params:
        window: 30
        threshold: 0.9999
    sink:
      - name: sqlite
        type: repository
        repository:
          type: sqlite
          params:
            database: /tmp/my_kafka_source_one.sqlite
      - name: kafka
        type: stream
        stream:
          type: kafka
          params:
            broker_servers: localhost:9092
            output_topic: test2
```

warmup

`warmup` section has two roles, the first is to be used to *warm up* the engine before starting making predictions. The second one is to make the data accessible from the dashboard to visualize it. We will define a `warmup` configuration section with one `repository` that is also used in `sink`.

```
version: 1

streams:
  - name: my_kafka_source_one
    source:
      type: kafka
      params:
        broker_servers: localhost:9092
        input_topic: test1
    engine:
      type: robust
      params:
        window: 30
        threshold: 0.9999
    sink:
      - name: sqlite
        type: repository
        repository:
          type: sqlite
          params:
            database: /tmp/my_kafka_source_one.sqlite
      - name: kafka
        type: stream
        stream:
          type: kafka
```

(continues on next page)

(continued from previous page)

```
          params:
            broker_servers: localhost:9092
            output_topic: test2
    warmup:
      - name: sqlite
        repository:
          type: sqlite
          params:
            database: /tmp/my_kafka_source_one.sqlite
```

repository

Repository section could be found in `sink` and in `warmup` sections, it defines an storage backend that is supported by *BaseSink* implementations, *RepositorySink* and *ObservableRepository* respectively.

That `sink` repository could be also used as `warmup` to *warm up* the model in case of a model that require previous data to evaluate new data. *Although it is defined as a list, only the first element will be used to warm up the model.*

```
stream:
  type: kafka
  params:
    broker_servers: localhost:9092
```

3.1.4.2 websocket

There is a `websocket` section that is used to send output `ticks` to the dashboard. This allows to update dashboard plots in realtime.

```
websocket: ws://localhost:5000/ws/
```

3.1.4.3 Example configuration file

anomdec.yml

A full example configuration. This configuration reflects a full message flow reading from a kafka broker, processing with robust detector warmed up with the same repository that persists the output.

```
version: 1

websocket: ws://localhost:5000/ws/

streams:
  - name: my_kafka_source_one
    source:
      type: kafka
      params:
        broker_servers: localhost:9092
        input_topic: test1
    engine:
      type: robust
      params:
```

(continues on next page)

(continued from previous page)

```
        window: 30
        threshold: 0.9999
    sink:
      - name: sqlite
        type: repository
        repository:
          type: sqlite
          params:
            database: /tmp/my_kafka_source_one.sqlite
      - name: kafka
        type: stream
        stream:
          type: kafka
          params:
            broker_servers: localhost:9092
            output_topic: test2
    warmup:
      - name: sqlite
        repository:
          type: sqlite
          params:
            database: /tmp/my_kafka_source_one.sqlite
```

diagram

Here it is a diagram that represents the full configuration file. We can see that the output of the engine could be *sinked* to a repository and to an streaming system to visualize and react for anomalies, and is also used to *warm up* the engine in case of restart or failure.

3.1.5 Deploy

Important: Before continue, please read the *Configuration* section, so you will need to create a configuration file to start distinct components of the framework.

3.1.5.1 Backend

Start only the backend for the given configuration. This allows you to split the configuration in small streams chunks to deploy independently to scale horizontally.

```
export ANOMDEC_HOME=~/anomdec
anomdec backend
```

```
- anomalydetection.anomdec.Anomdec:35 - INFO - Starting anomdec
- anomalydetection.anomdec.Anomdec:44 - INFO - Run backend
...
```

3.1.5.2 Dashboard

Start only the dashboard for the given configuration. Using a unique config file you are able to visualize all streams.

```
export ANOMDEC_HOME=~/anomdec
anomdec dashboard
```

```
- anomalydetection.anomdec.Anomdec:35 - INFO - Starting anomdec
- anomalydetection.anomdec.Anomdec:41 - INFO - Run dashboard
...
```

3.1.5.3 Embedded

You can start it embedded. This will create a subprocess in the dashboard instance to process the streams.

```
export ANOMDEC_HOME=~/anomdec
anomdec
```

```
- anomalydetection.anomdec.Anomdec:35 - INFO - Starting anomdec
- anomalydetection.anomdec.Anomdec:38 - INFO - Run dashboard/backend embedded
...
```

3.1.6 Plugins

Anomaly Detection Framework is extensible by a plugin system. Plugin files must be placed in `$ANOMDEC_HOME/plugins` and are supported by the configuration parser of `anomdec.yml` file.

3.1.6.1 Interface

To create a plugin you will need to implement the class `Plugin`. You can implement any *core-around* component, so you can extend functionality as much as you want.

```
class Plugin(object):

    name = None
    # A class in a list derived from BaseConsumerBuilder
    stream_consumer_builders = []
    # A class in a list derived from BaseStreamConsumer
    stream_consumers = []
    # A class in a list derived from BaseProducerBuilder
    stream_producer_builders = []
    # A class in a list derived from BaseStreamProducer
    stream_producers = []
    # A class in a list derived from BaseEngineBuilder
    engine_builders = []
    # A class in a list derived from BaseEngine
    engines = []
    # A class in a list derived from BaseRepositoryBuilder
    repository_builders = []
    # A class in a list derived from BaseRepository
    repositories = []
    # A class in a list derived from BaseMessageHandler
    message_handlers = []
```

3.1.6.2 Example

Image you want to implement a new source, so you need to implement the *Builder* `BaseConsumerBuilder` and the *Consumer* `BaseStreamConsumer`. This consumer will read messages from a *socket*.

```
from anomalydetection.backend.core.plugins import Plugin

# Needs to be implemented
class SocketConsumer(BaseStreamConsumer):

    def __init__(self, hostname, port):
        pass

# This builder is fully implemented, note that the attrs has its own
# setter in the form of ``set_<attrname>``
class SocketConsumerBuilder(BaseConsumerBuilder):

    def __init__(self, hostname, port):
        self.set_hostname(hostname)
        self.set_port(port)

    def set_hostname(self, hostname):
        self.hostname = str(hostname)
        return self

    def set_port(self, port):
        self.port = int(port)
        return self

    def build():
        return SocketConsumer(**vars(self).copy())

class SocketsPlugin(Plugin):
    name = "socket"
    stream_consumer_builders = [SocketConsumerBuilder]
    stream_consumers = [SocketConsumer]
```

3.1.6.3 Use in anomdec.yml

The plugin system is fully supported by config parser using the plugin name in `type` sections. And params to the builder are also supported by `params` sections. Here is an example `anomdec.yml` file using this plugin.

```
version: 1

streams:

  - name: test
    source:
      type: socket
      params:
        hostname: localhost
        port: 8080
```

3.1.7 API Reference

3.1.7.1 anomalydetection

Anomaly Detection Framework is composed by 3 modules

3.1.7.2 anomalydetection.common

This module contains common components of the framework.

3.1.7.3 anomalydetection.backend

This module contains the core of the anomaly detection framework implementation.

backend.core

backend.core.plugins

backend.core.config

backend.entities

class anomalydetection.backend.entities.**BaseMessageHandler**

Bases: typing.Generic

Base message handler

classmethod extract_extra(*message*)

Extract extra data from the parsed message

Parameters **message** (~*T*) – parsed message

Return type dict

Returns a dict of extra values

classmethod extract_key(*message*)

Extracts the key of the message, this value is used to group messages

Parameters **message** (~*T*) – parsed message

Return type str

Returns word

classmethod extract_ts(*message*)

Extract the datetime of the message

Parameters **message** (~*T*) – parsed message

Return type datetime

Returns a datetime object

classmethod extract_value(*message*)

Extracts the value of the message, this value is that is given to make the prediction

Parameters **message** (~*T*) – parsed message

Return type float

Returns a float value

classmethod parse_message(*message*)
Parse or transform an input message and returns it

Parameters **message** (Any) – message serialized in a string

Return type ~T

Returns parsed message

classmethod validate_message(*message*)
Validates a message

Parameters **message** (*~T*) – validates if a message is valid or not

Return type bool

Returns True if is valid, False if is not

class anomalydetection.backend.entities.input_message.**InputMessage**(*application*, *value*, *ts*)
Bases: object

This is the parser of a json message

Parameters

- **application** (str) – application
- **value** (float) – value
- **ts** (Any) – datetime or current time stamp string in ISO 8601

to_dict()

to_json()

class anomalydetection.backend.entities.input_message.**InputMessageHandler**
Bases: *anomalydetection.backend.entities.BaseMessageHandler*

classmethod extract_extra(*message*)
Extract extra data from the parsed message

Parameters **message** (*~T*) – parsed message

Return type dict

Returns a dict of extra values

classmethod extract_key(*message*)
Extracts the key of the message, this value is used to group messages

Parameters **message** (*InputMessage*) – parsed message

Return type str

Returns word

classmethod extract_ts(*message*)
Extract the datetime of the message

Parameters **message** (*~T*) – parsed message

Return type datetime

Returns a datetime object

classmethod extract_value(*message*)
Extracts the value of the message, this value is that is given to make the prediction

Parameters **message** (*InputMessage*) – parsed message

Return type float

Returns a float value

classmethod parse_message(*message*)
Parse or transform an input message and returns it

Parameters **message** (*InputMessage*) – message serialized in a string

Return type *InputMessage*

Returns parsed message

classmethod validate_message(*message*)
Validates a message

Parameters **message** (*InputMessage*) – validates if a message is valid or not

Return type bool

Returns True if is valid, False if is not

class anomalydetection.backend.entities.output_message.**AnomalyResult**(*value_lower_limit*, *value_upper_limit*, *anomaly_probability*, *is_anomaly*)

Bases: object

AnomalyResult description

Variables

- **value_lower_limit** – lower bound limit
- **value_upper_limit** – upper bound limit
- **anomaly_probability** – probability of being anomalous
- **is_anomaly** – if its anomalous or not

AnomalyResults constructor

Parameters

- **value_lower_limit** (float) – lower bound limit
- **value_upper_limit** (float) – upper bound limit
- **anomaly_probability** (float) – probability of being anomalous
- **is_anomaly** (bool) – if its anomalous or not

to_dict()

class anomalydetection.backend.entities.output_message.**OutputMessage**(*application*, *anomaly_results*, *agg_window_millis=0*, *agg_function=<AggregationFu 'none'>*, *agg_value=0*, *ts=datetime.datetime(2018, 7, 11, 14, 28, 33, 542179)*)

Bases: object

OutputMessage class description

Variables

- **application** – application name
- **anomaly_results** – anomaly results
- **agg_window_millis** – aggregation window in milliseconds
- **agg_function** – aggregation function
- **agg_value** – the value after aggregation
- **ts** – timestamp

OutputMessage class constructor

Parameters

- **application** (*str*) – application name
- **anomaly_results** (*AnomalyResult*) – anomaly results
- **agg_window_millis** (*int*) – aggregation window in milliseconds
- **agg_function** (*AggregationFunction*) – aggregation function
- **agg_value** (*float*) – the value after aggregation
- **ts** (*<module 'datetime' from '/usr/lib/python3.5/datetime.py'>*) – timestamp

to_dict(*ts2str=False*)

to_input()

to_plain_dict()

class anomalydetection.backend.entities.output_message.**OutputMessageHandler**

Bases: *anomalydetection.backend.entities.BaseMessageHandler*

classmethod extract_extra(*message*)

Extract extra data from the parsed message

Parameters **message** (*InputMessage*) – parsed message

Return type dict

Returns a dict of extra values

classmethod extract_key(*message*)

Extracts the key of the message, this value is used to group messages

Parameters **message** (*InputMessage*) – parsed message

Return type str

Returns word

classmethod extract_ts(*message*)
Extract the datetime of the message

Parameters **message** (*~T*) – parsed message

Return type datetime

Returns a datetime object

classmethod extract_value(*message*)
Extracts the value of the message, this value is that is given to make the prediction

Parameters **message** (*InputMessage*) – parsed message

Return type float

Returns a float value

classmethod parse_message(*message*)
Parse or transform an input message and returns it

Parameters **message** (*OutputMessage*) – message serialized in a string

Return type *InputMessage*

Returns parsed message

classmethod validate_message(*message*)
Validates a message

Parameters **message** (*InputMessage*) – validates if a message is valid or not

Return type bool

Returns True if is valid, False if is not

There is a default JSON format message handler implementation ready to use

backend.engine

class anomalydetection.backend.engine.builder.**BaseEngineBuilder**
Bases: object

BaseBuilder, implement this to create Engine Builders.

build()
Build the engine

Return type *BaseEngine*

Returns A BaseEngine implementation instance.

set(*name*, *value*)

class anomalydetection.backend.engine.builder.**CADDetectorBuilder**(*min_value=-9223372036854775807*, *max_value=9223372036854775807*, *threshold=0.95*, *rest_period=30*, *max_left_semi_contexts_length=8*, *max_active_neurons_num=16*, *num_norm_value_bits=3*)

Bases: *anomalydetection.backend.engine.builder.BaseEngineBuilder*

build()

Build the engine

Return type *CADDetector*

Returns A BaseEngine implementation instance.

set(*name*, *value*)

set_max_active_neurons_num(*max_active_neurons_num*)

set_max_left_semi_contexts_length(*max_left_semi_contexts_length*)

set_max_value(*value*)

set_min_value(*value*)

set_num_norm_value_bits(*num_norm_value_bits*)

set_rest_period(*rest_period*)

set_threshold(*threshold*)

type = 'cad'

class anomalydetection.backend.engine.builder.**EMADetectorBuilder**(*window=100*, *threshold=0.9999*)

Bases: *anomalydetection.backend.engine.builder.BaseEngineBuilder*

build()

Build the engine

Return type *EMADetector*

Returns A BaseEngine implementation instance.

set(*name*, *value*)

set_threshold(*threshold*)

set_window(*window*)

type = 'ema'

class anomalydetection.backend.engine.builder.**EngineBuilderFactory**

Bases: object

engines = {'cad': {'key': 'cad', 'name': 'CADDetector'}, 'ema': {'key': 'ema', 'na

static get(*name*)

Return type *BaseEngineBuilder*

static get_cad()

Return type *CADDetectorBuilder*

static get_ema()

Return type *EMADetectorBuilder*

static get_plugin(*name*)

Return type *BaseEngineBuilder*

static get_robust()

Return type *RobustDetectorBuilder*

classmethod register_engine(*key*, *class_name*)

class anomalydetection.backend.engine.builder.**RobustDetectorBuilder**(*window=100*, *threshold=0.9999*)

Bases: *anomalydetection.backend.engine.builder.BaseEngineBuilder*

build()

Build the engine

Return type *RobustDetector*

Returns A BaseEngine implementation instance.

set(*name*, *value*)

set_threshold(*threshold*)

set_window(*window*)

type = 'robust'

class anomalydetection.backend.engine.**BaseEngine**

Bases: object

Base class for any Engine implementation.

predict(*value*, ***kwargs*)

Predict if the given value is anomalous.

Parameters

- **value** (float) – value to predict
- **kwargs** – extra data to make the prediction

Return type *AnomalyResult*

Returns anomaly result

class anomalydetection.backend.engine.cad_engine.**CADDetector**(*min_value=-9223372036854775807*, *max_value=9223372036854775807*, *threshold=0.95*, *rest_period=30*, *max_left_semi_contexts_length=8*, *max_active_neurons_num=16*, *num_norm_value_bits=3*)

Bases: *anomalydetection.backend.engine.BaseEngine*

Contextual Anomaly Detector - Open Source Edition https://github.com/smirmik/CAD

get_anomaly_score(*input_data*)

predict(*value*, ***kwargs*)
Predict if the given value is anomalous.

Parameters

- **value** (float) – value to predict
- **kwargs** – extra data to make the prediction

Return type *AnomalyResult*

Returns anomaly result

step(*inp_facts*)

class anomalydetection.backend.engine.cad_engine.**ContextOperator**(*max_left_semi_cntx_len*)
Bases: object

context_crosser(*left_or_right*, *facts_list*, *new_context_flag=False*, *potential_new_contexts=[]*)

get_context_by_facts(*new_contexts_list*, *zero_level=0*)

update_contexts_and_get_active(*new_context_flag*)

class anomalydetection.backend.engine.ema_engine.**EMADetector**(*window=100*, *threshold=2.0*)
Bases: *anomalydetection.backend.engine.BaseEngine*

EMADetector constructor :param window: window of samples to work with :param threshold: threshold for confidence

predict(*value*, ***kwargs*)
Predict if the given value is anomalous.

Parameters

- **value** (float) – value to predict
- **kwargs** – extra data to make the prediction

Return type *AnomalyResult*

Returns anomaly result

class anomalydetection.backend.engine.robust_z_engine.**RobustDetector**(*window=100*, *threshold=0.9999*)
Bases: *anomalydetection.backend.engine.BaseEngine*, anomalydetection.common.logging.LoggingMixin

Anomaly detection engine based in robust statistics, median and median absolute deviation.

Parameters

- **window** – window of samples to work with
- **threshold** – threshold for confidence

logger
Logger object.

Returns a configured logger object

predict(*value*, ***kwargs*)
Predict if the given value is anomalous.

Parameters

- **value** (*float*) – value to predict
- **kwargs** – extra data to make the prediction

Return type *AnomalyResult*

Returns anomaly result

backend.interactor

class anomalydetection.backend.interactor.**BaseEngineInteractor**(*engine_builder*, *message_handler*)

Bases: object

BaseEngineInteractor is responsible to hold the engine builder and the message handler. It's also responsible for build engines for each application

BaseEngineInteractor constructor

Parameters

- **engine_builder** (*BaseEngineBuilder*) – engine builder
- **message_handler** (*BaseMessageHandler*[~T]) – message handler

get_engine(*application*)

Return the engine for the application in a thread safe way

Parameters **application** (*str*) – application name

Returns its engine

class anomalydetection.backend.interactor.batch_engine.**BatchEngineInteractor**(*batch*, *engine_builder*, *message_handler*)

Bases: *anomalydetection.backend.interactor.BaseEngineInteractor*, anomalydetection.common.logging.LoggingMixin

BatchEngineInteractor is an implementation for batch process an Observable

BatchEngineInteractor constructor

Parameters

- **batch** (*BaseObservable*) – an observable
- **engine_builder** (*BaseEngineBuilder*) – an engine builder
- **message_handler** (*BaseMessageHandler*[~T]) – a message handler

get_engine(*application*)

Return the engine for the application in a thread safe way

Parameters **application** (*str*) – application name

Returns its engine

logger

Logger object.

Returns a configured logger object

map_with_engine(*input_message*)

Return type *OutputMessage*

process()

Return type Observable

backend.repository

class anomalydetection.backend.repository.**BaseObservableRepository**(*repository*)

Bases: *anomalydetection.backend.stream.BaseObservable*

Use a repository as an Observable

BaseObservableRepository constructor

Parameters **repository** (*BaseRepository*) – a repository

get_max()

get_min()

get_observable()

Return type Observable

map(*x*)

Map items in observable.

Parameters **x** (Any) – input item

Return type Any

Returns output item

class anomalydetection.backend.repository.**BaseRepository**(*conn*)

Bases: object

fetch(*application*, *from_ts*, *to_ts*)

Fetch data from repository, should return ordered data

Parameters

- **application** – application name
- **from_ts** – from timestamp
- **to_ts** – to timestamp

Return type Iterable[+T_co]

Returns an iterable

get_applications()

Return a list of distinct applications contained in the repository.

Return type List[str]

Returns a list of application names

initialize()

Initialize the repository

insert(*message*)

Insert an OutputMessage into the repository

Parameters **message** (*OutputMessage*) – an output message

Return type None

map(*item*)
Map function to map elements returned by fetch method to OutputMessage

Parameters **item** (*Any*) – an element in fetch iterable

Return type *OutputMessage*

Returns an OutputMessage

backend.repository.builder

class anomalydetection.backend.repository.builder.**BaseRepositoryBuilder**
Bases: object

BaseBuilder, implement this to create Repository Builders.

build()
Build a repository

Return type *BaseRepository*

Returns A BaseRepository implementation instance.

set(*name*, *value*)

class anomalydetection.backend.repository.builder.**RepositoryBuilderFactory**
Bases: object

static get(*name*)

Return type *BaseRepositoryBuilder*

static get_plugin(*name*)

Return type *BaseRepositoryBuilder*

static get_sqlite()

Return type *SQLiteBuilder*

class anomalydetection.backend.repository.builder.**SQLiteBuilder**(*database=None*)
Bases: *anomalydetection.backend.repository.builder.BaseRepositoryBuilder*

build()
Build a repository

Return type *BaseRepository*

Returns A BaseRepository implementation instance.

set(*name*, *value*)

set_database(*database*)

backend.repository.observable

class anomalydetection.backend.repository.observable.**ObservableRepository**(*repository*, *application=None*, *from_ts=None*, *to_ts=None*)

Bases: *anomalydetection.backend.repository.BaseObservableRepository*

Creates ObservableRepository that is capable to act as an observable

Parameters

- **repository** (*BaseRepository*) – the repository
- **application** – application name
- **from_ts** – from timestamp
- **to_ts** – to timestamp

get_max()

get_min()

get_observable()

Return type Observable

map(*x*)

Map items in observable.

Parameters **x** (Any) – input item

Return type Any

Returns output item

backend.repository.sqlite

class anomalydetection.backend.repository.sqlite.**SQLiteRepository**(*database*)

Bases: *anomalydetection.backend.repository.BaseRepository*

SQLiteRepository constructor

Parameters **database** (str) – database path

fetch(*application*, *from_ts*, *to_ts*)

Fetch data from repository, should return ordered data

Parameters

- **application** – application name
- **from_ts** – from timestamp
- **to_ts** – to timestamp

Return type Iterable[Row]

Returns an iterable

get_applications()
Return a list of distinct applications contained in the repository.

Returns a list of application names

initialize()
Initialize the repository

insert(*message*)
Insert an OutputMessage into the repository

Parameters **message** (*OutputMessage*) – an output message

map(*item*)
Map function to map elements returned by fetch method to OutputMessage

Parameters **item** (Row) – an element in fetch iterable

Return type *OutputMessage*

Returns an OutputMessage

backend.sink

class anomalydetection.backend.sink.**BaseSink**
Bases: rx.core.py3.observer.Observer

BaseSink, implement this to create Sinks

as_observer()
Hides the identity of an observer.

Returns an observer that hides the identity of the specified observer.

checked()
Checks access to the observer for grammar violations. This includes checking for multiple OnError or OnCompleted calls, as well as reentrancy in any of the observer methods. If a violation is detected, an Error is thrown from the offending observer method call.

Returns an observer that checks callbacks invocations against the observer grammar and, if the checks pass, forwards those to the specified observer.

classmethod from_notifier(*handler*)
Creates an observer from a notification callback.

Keyword arguments: :param handler: Action that handles a notification.

Returns The observer object that invokes the specified handler using a

notification corresponding to each message it receives. :rtype: Observer

on_completed()

on_error(*error*)

on_next(*value*)

to_notifier()
Creates a notification callback from an observer.

Returns the action that forwards its input notification to the underlying observer.

backend.sink.repository

class anomalydetection.backend.sink.repository.**RepositorySink**(*repository*)
Bases: *anomalydetection.backend.sink.BaseSink*, anomalydetection.common.logging.LoggingMixin

Creates a RepositorySink that is capable to sink OutputMessages into the given repository

Parameters **repository** (*BaseRepository*) – a repository

as_observer()
Hides the identity of an observer.

Returns an observer that hides the identity of the specified observer.

checked()
Checks access to the observer for grammar violations. This includes checking for multiple OnError or OnCompleted calls, as well as reentrancy in any of the observer methods. If a violation is detected, an Error is thrown from the offending observer method call.

Returns an observer that checks callbacks invocations against the observer grammar and, if the checks pass, forwards those to the specified observer.

classmethod from_notifier(*handler*)
Creates an observer from a notification callback.

Keyword arguments: :param handler: Action that handles a notification.

Returns The observer object that invokes the specified handler using a

notification corresponding to each message it receives. :rtype: Observer

logger
Logger object.

Returns a configured logger object

on_completed()

on_error(*error*)

on_next(*value*)

to_notifier()
Creates a notification callback from an observer.

Returns the action that forwards its input notification to the underlying observer.

backend.sink.stream

class anomalydetection.backend.sink.stream.**StreamSink**(*stream_producer*)
Bases: *anomalydetection.backend.sink.BaseSink*, anomalydetection.common.logging.LoggingMixin

Creates a StreamSink that is capable to sink OutputMessages into the stream producer

Parameters **stream_producer** (*BaseStreamProducer*) – an stream producer

as_observer()
Hides the identity of an observer.

Returns an observer that hides the identity of the specified observer.

checked()
Checks access to the observer for grammar violations. This includes checking for multiple OnError or OnCompleted calls, as well as reentrancy in any of the observer methods. If a violation is detected, an Error is thrown from the offending observer method call.

Returns an observer that checks callbacks invocations against the observer grammar and, if the checks pass, forwards those to the specified observer.

classmethod from_notifier(*handler*)
Creates an observer from a notification callback.

Keyword arguments: :param handler: Action that handles a notification.

Returns The observer object that invokes the specified handler using a

notification corresponding to each message it receives. :rtype: Observer

logger
Logger object.

Returns a configured logger object

on_completed()

on_error(*error*)

on_next(*value*)

to_notifier()
Creates a notification callback from an observer.

Returns the action that forwards its input notification to the underlying observer.

backend.sink.websocket

class anomalydetection.backend.sink.websocket.**WebSocketSink**(*name*, *url*)
Bases: *anomalydetection.backend.sink.BaseSink*, anomalydetection.common.logging.LoggingMixin

Implementation to Sink OutputMessage stream to a WebSocket

Parameters

- **name** (str) – name
- **url** (str) – websocket url

as_observer()
Hides the identity of an observer.

Returns an observer that hides the identity of the specified observer.

checked()
Checks access to the observer for grammar violations. This includes checking for multiple OnError or OnCompleted calls, as well as reentrancy in any of the observer methods. If a violation is detected, an Error is thrown from the offending observer method call.

Returns an observer that checks callbacks invocations against the observer grammar and, if the checks pass, forwards those to the specified observer.

classmethod from_notifier(*handler*)
Creates an observer from a notification callback.

Keyword arguments: :param handler: Action that handles a notification.

Returns The observer object that invokes the specified handler using a

notification corresponding to each message it receives. :rtype: Observer

logger
Logger object.

Returns a configured logger object

on_completed()

on_error(*error*)

on_next(*value*)

to_notifier()
Creates a notification callback from an observer.

Returns the action that forwards its input notification to the underlying observer.

backend.stream

class anomalydetection.backend.stream.**BaseObservable**
Bases: object

get_observable()

Return type Observable

map(*x*)
Map items in observable.

Parameters **x** (Any) – input item

Return type Any

Returns output item

class anomalydetection.backend.stream.**BaseStreamAggregation**(*agg_function=<AggregationFunction.NONE: 'none'>*, *agg_window_millis=0*)
Bases: object

BaseStreamAggregation class

Parameters

- **agg_function** (*AggregationFunction*) – aggregation function
- **agg_window_millis** (int) – aggregation window in milliseconds

class anomalydetection.backend.stream.**BaseStreamConsumer**
Bases: *anomalydetection.backend.stream.BaseObservable*

get_observable()

Return type Observable

map(*x*)
Map items in observable.

Parameters **x** (Any) – input item

Return type Any

Returns output item

poll()

Return type Generator

class anomalydetection.backend.stream.**BaseStreamProducer**
Bases: object

push(*message*)

Return type None

class anomalydetection.backend.stream.**FileObservable**(*file*)
Bases: *anomalydetection.backend.stream.BaseObservable*

FileObservable to transform a file lines to an Observable

Parameters **file** (str) – a path to local file

get_observable()

Return type Observable

map(*x*)
Map items in observable.

Parameters **x** (Any) – input item

Return type Any

Returns output item

backend.stream.agg

class anomalydetection.backend.stream.agg.functions.**AggregationFunction**
Bases: enum.Enum

An enumeration.

AVG = 'avg'

COUNT = 'count'

NONE = 'none'

P50 = 'percentile50'

P75 = 'percentile75'

P95 = 'percentile95'

P99 = 'percentile99'

SUM = 'sum'

backend.stream.builder

backend.stream.kafka

class anomalydetection.backend.stream.kafka.**KafkaStreamConsumer**(*broker_servers*, *input_topic*, *group_id*)
Bases: *anomalydetection.backend.stream.BaseStreamConsumer*, anomalydetection.common.logging.LoggingMixin

KafkaStreamConsumer constructor

Parameters

- **broker_servers** (str) – broker servers
- **input_topic** (str) – input topic
- **group_id** (str) – consumer group id

get_observable()

Return type Observable

logger
Logger object.

Returns a configured logger object

map(*x*)
Map items in observable.

Parameters **x** (Any) – input item

Return type Any

Returns output item

poll()

Return type Generator[+T_co, -T_contra, +V_co]

unsubscribe()

class anomalydetection.backend.stream.kafka.**KafkaStreamProducer**(*broker_servers*, *output_topic*)

Bases: *anomalydetection.backend.stream.BaseStreamProducer*, anomalydetection.common.logging.LoggingMixin

KafkaStreamProducer constructor

Parameters

- **broker_servers** (str) – broker servers
- **output_topic** (str) – topic to write to

logger
Logger object.

Returns a configured logger object

push(*message*)

Return type None

class anomalydetection.backend.stream.kafka.**SparkKafkaStreamConsumer**(*broker_servers*, *input_topic*, *group_id*, *agg_function*, *agg_window_millis*, *spark_opts={}*, *multiprocessing=True*)

Bases: *anomalydetection.backend.stream.BaseStreamConsumer*, *anomalydetection.backend.stream.BaseStreamAggregation*, anomalydetection.common.logging.LoggingMixin

SparkKafkaStreamConsumer constructor

Parameters

- **broker_servers** (str) – broker servers
- **input_topic** (str) – input topic
- **group_id** (str) – consumer group id
- **agg_function** (*AggregationFunction*) – aggregation function to apply
- **agg_window_millis** (int) – aggregation window in milliseconds
- **spark_opts** (dict) – spark options dict
- **multiprocessing** – use multiprocessing instead of threading

get_observable()

Return type Observable

logger

Logger object.

Returns a configured logger object

map(*x*)

Map items in observable.

Parameters **x** (Any) – input item

Return type Any

Returns output item

poll()

Return type Generator[+T_co, -T_contra, +V_co]

unsubscribe()

backend.stream.pubsub

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

anomalydetection, 27
anomalydetection.backend, 27
anomalydetection.backend.core, 27
anomalydetection.backend.engine, 33
anomalydetection.backend.engine.builder, 31
anomalydetection.backend.engine.cad_engine, 33
anomalydetection.backend.engine.ema_engine, 34
anomalydetection.backend.engine.robust_z_engine, 34
anomalydetection.backend.entities, 27
anomalydetection.backend.entities.input_message, 28
anomalydetection.backend.entities.output_message, 29
anomalydetection.backend.interactor, 35
anomalydetection.backend.interactor.batch_engine, 35
anomalydetection.backend.repository, 36
anomalydetection.backend.repository.builder, 37
anomalydetection.backend.repository.observable, 38
anomalydetection.backend.repository.sqlite, 38
anomalydetection.backend.sink, 39
anomalydetection.backend.sink.repository, 40
anomalydetection.backend.sink.stream, 40
anomalydetection.backend.sink.websocket, 41
anomalydetection.backend.stream, 42
anomalydetection.backend.stream.agg.functions, 43
anomalydetection.backend.stream.kafka, 43
anomalydetection.common, 27

Index

A

AggregationFunction (class in anomalydetection.backend.stream.agg.functions), 43
anomalydetection (module), 27
anomalydetection.backend (module), 27
anomalydetection.backend.core (module), 27
anomalydetection.backend.engine (module), 33
anomalydetection.backend.engine.builder (module), 31
anomalydetection.backend.engine.cad_engine (module), 33
anomalydetection.backend.engine.ema_engine (module), 34
anomalydetection.backend.engine.robust_z_engine (module), 34
anomalydetection.backend.entities (module), 27
anomalydetection.backend.entities.input_message (module), 28
anomalydetection.backend.entities.output_message (module), 29
anomalydetection.backend.interactor (module), 35
anomalydetection.backend.interactor.batch_engine (module), 35
anomalydetection.backend.repository (module), 36
anomalydetection.backend.repository.builder (module), 37
anomalydetection.backend.repository.observable (module), 38
anomalydetection.backend.repository.sqlite (module), 38
anomalydetection.backend.sink (module), 39
anomalydetection.backend.sink.repository (module), 40
anomalydetection.backend.sink.stream (module), 40
anomalydetection.backend.sink.websocket (module), 41
anomalydetection.backend.stream (module), 42
anomalydetection.backend.stream.agg.functions (module), 43
anomalydetection.backend.stream.kafka (module), 43
anomalydetection.common (module), 27
AnomalyResult (class in anomalydetection.backend.entities.output_message), 29
as_observer() (anomalydetection.backend.sink.BaseSink method), 39
as_observer() (anomalydetection.backend.sink.repository.RepositorySink method), 40
as_observer() (anomalydetection.backend.sink.stream.StreamSink method), 40
as_observer() (anomalydetection.backend.sink.websocket.WebSocketSink method), 41
AVG (anomalydetection.backend.stream.agg.functions.AggregationFunctio attribute), 43

B

BaseEngine (class in anomalydetection.backend.engine), 33
BaseEngineBuilder (class in anomalydetection.backend.engine.builder), 31
BaseEngineInteractor (class in anomalydetection.backend.interactor), 35
BaseMessageHandler (class in anomalydetection.backend.entities), 27
BaseObservable (class in anomalydetection.backend.stream), 42
BaseObservableRepository (class in anomalydetection.backend.repository), 36
BaseRepository (class in anomalydetection.backend.repository), 36
BaseRepositoryBuilder (class in anomalydetection.backend.repository.builder), 37
BaseSink (class in anomalydetection.backend.sink), 39
BaseStreamAggregation (class in anomalydetection.backend.stream), 42
BaseStreamConsumer (class in anomalydetection.backend.stream), 42
BaseStreamProducer (class in anomalydetection.backend.stream), 43
BatchEngineInteractor (class in anomalydetection.backend.interactor.batch_engine), 35

build() (anomalydetection.backend.engine.builder.BaseEngineBuilder method), 31
build() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
build() (anomalydetection.backend.engine.builder.EMADetectorBuilder method), 32
build() (anomalydetection.backend.engine.builder.RobustDetectorBuilder method), 33
build() (anomalydetection.backend.repository.builder.BaseRepositoryBuilder method), 37
build() (anomalydetection.backend.repository.builder.SQLiteBuilder method), 37

C

CADDetector (class in anomalydetection.backend.engine.cad_engine), 33
CADDetectorBuilder (class in anomalydetection.backend.engine.builder), 31
checked() (anomalydetection.backend.sink.BaseSink method), 39
checked() (anomalydetection.backend.sink.repository.RepositorySink method), 40
checked() (anomalydetection.backend.sink.stream.StreamSink method), 40
checked() (anomalydetection.backend.sink.websocket.WebSocketSink method), 41
context_crosser() (anomalydetection.backend.engine.cad_engine.ContextOperator method), 34
ContextOperator (class in anomalydetection.backend.engine.cad_engine), 34
COUNT (anomalydetection.backend.stream.agg.functions.AggregationFunction attribute), 43

E

EMADetector (class in anomalydetection.backend.engine.ema_engine), 34
EMADetectorBuilder (class in anomalydetection.backend.engine.builder), 32
EngineBuilderFactory (class in anomalydetection.backend.engine.builder), 32
engines (anomalydetection.backend.engine.builder.EngineBuilderFactory attribute), 32
extract_extra() (anomalydetection.backend.entities.BaseMessageHandler class method), 27
extract_extra() (anomalydetection.backend.entities.input_message.InputMessageHandler class method), 28
extract_extra() (anomalydetection.backend.entities.output_message.OutputMessageHandler class method), 30
extract_key() (anomalydetection.backend.entities.BaseMessageHandler class method), 27
extract_key() (anomalydetection.backend.entities.input_message.InputMessageHandler class method), 28
extract_key() (anomalydetection.backend.entities.output_message.OutputMessageHandler class method), 30
extract_ts() (anomalydetection.backend.entities.BaseMessageHandler class method), 27
extract_ts() (anomalydetection.backend.entities.input_message.InputMessageHandler class method), 28
extract_ts() (anomalydetection.backend.entities.output_message.OutputMessageHandler class method), 31
extract_value() (anomalydetection.backend.entities.BaseMessageHandler class method), 27
extract_value() (anomalydetection.backend.entities.input_message.InputMessageHandler class method), 28
extract_value() (anomalydetection.backend.entities.output_message.OutputMessageHandler class method), 31

F

fetch() (anomalydetection.backend.repository.BaseRepository method), 36
fetch() (anomalydetection.backend.repository.sqlite.SQLiteRepository method), 38
FileObservable (class in anomalydetection.backend.stream), 43
from_notifier() (anomalydetection.backend.sink.BaseSink class method), 39
from_notifier() (anomalydetection.backend.sink.repository.RepositorySink class method), 40
from_notifier() (anomalydetection.backend.sink.stream.StreamSink class method), 41
from_notifier() (anomalydetection.backend.sink.websocket.WebSocketSink class method), 41

G

get() (anomalydetection.backend.engine.builder.EngineBuilderFactory static method), 32

get() (anomalydetection.backend.repository.builder.RepositoryBuilderFactory static method), 37
get_anomaly_score() (anomalydetection.backend.engine.cad_engine.CADDetector method), 33
get_applications() (anomalydetection.backend.repository.BaseRepository method), 36
get_applications() (anomalydetection.backend.repository.sqlite.SQLiteRepository method), 38
get_cad() (anomalydetection.backend.engine.builder.EngineBuilderFactory static method), 32
get_context_by_facts() (anomalydetection.backend.engine.cad_engine.ContextOperator method), 34
get_ema() (anomalydetection.backend.engine.builder.EngineBuilderFactory static method), 33
get_engine() (anomalydetection.backend.interactor.BaseEngineInteractor method), 35
get_engine() (anomalydetection.backend.interactor.batch_engine.BatchEngineInteractor method), 35
get_max() (anomalydetection.backend.repository.BaseObservableRepository method), 36
get_max() (anomalydetection.backend.repository.observable.ObservableRepository method), 38
get_min() (anomalydetection.backend.repository.BaseObservableRepository method), 36
get_min() (anomalydetection.backend.repository.observable.ObservableRepository method), 38
get_observable() (anomalydetection.backend.repository.BaseObservableRepository method), 36
get_observable() (anomalydetection.backend.repository.observable.ObservableRepository method), 38
get_observable() (anomalydetection.backend.stream.BaseObservable method), 42
get_observable() (anomalydetection.backend.stream.BaseStreamConsumer method), 42
get_observable() (anomalydetection.backend.stream.FileObservable method), 43
get_observable() (anomalydetection.backend.stream.kafka.KafkaStreamConsumer method), 44
get_observable() (anomalydetection.backend.stream.kafka.SparkKafkaStreamConsumer method), 45
get_plugin() (anomalydetection.backend.engine.builder.EngineBuilderFactory static method), 33
get_plugin() (anomalydetection.backend.repository.builder.RepositoryBuilderFactory static method), 37
get_robust() (anomalydetection.backend.engine.builder.EngineBuilderFactory static method), 33
get_sqlite() (anomalydetection.backend.repository.builder.RepositoryBuilderFactory static method), 37

I

initialize() (anomalydetection.backend.repository.BaseRepository method), 36
initialize() (anomalydetection.backend.repository.sqlite.SQLiteRepository method), 39
InputMessage (class in anomalydetection.backend.entities.input_message), 28
InputMessageHandler (class in anomalydetection.backend.entities.input_message), 28
insert() (anomalydetection.backend.repository.BaseRepository method), 36
insert() (anomalydetection.backend.repository.sqlite.SQLiteRepository method), 39

K

KafkaStreamConsumer (class in anomalydetection.backend.stream.kafka), 43
KafkaStreamProducer (class in anomalydetection.backend.stream.kafka), 44

L

logger (anomalydetection.backend.engine.robust_z_engine.RobustDetector attribute), 34
logger (anomalydetection.backend.interactor.batch_engine.BatchEngineInte attribute), 35
logger (anomalydetection.backend.sink.repository.RepositorySink attribute), 40
logger (anomalydetection.backend.sink.stream.StreamSink attribute), 41
logger (anomalydetection.backend.sink.websocket.WebSocketSink attribute), 42

logger (anomalydetection.backend.stream.kafka.KafkaStreamConsumer attribute), 44
logger (anomalydetection.backend.stream.kafka.KafkaStreamProducer attribute), 44
logger (anomalydetection.backend.stream.kafka.SparkKafkaStreamConsumer attribute), 45

M

map() (anomalydetection.backend.repository.BaseObservableRepository method), 36
map() (anomalydetection.backend.repository.BaseRepository method), 37
map() (anomalydetection.backend.repository.observable.ObservableRepository method), 38
map() (anomalydetection.backend.repository.sqlite.SQLiteRepository method), 39
map() (anomalydetection.backend.stream.BaseObservable method), 42
map() (anomalydetection.backend.stream.BaseStreamConsumer method), 42
map() (anomalydetection.backend.stream.FileObservable method), 43
map() (anomalydetection.backend.stream.kafka.KafkaStreamConsumer method), 44
map() (anomalydetection.backend.stream.kafka.SparkKafkaStreamConsumer method), 45
map_with_engine() (anomalydetection.backend.interactor.batch_engine.BatchEngineInteractor method), 35

N

NONE (anomalydetection.backend.stream.agg.functions.AggregationFunction attribute), 43

O

ObservableRepository (class in anomalydetection.backend.repository.observable), 38
on_completed() (anomalydetection.backend.sink.BaseSink method), 39
on_completed() (anomalydetection.backend.sink.repository.RepositorySink method), 40
on_completed() (anomalydetection.backend.sink.stream.StreamSink method), 41
on_completed() (anomalydetection.backend.sink.websocket.WebSocketSink method), 42
on_error() (anomalydetection.backend.sink.BaseSink method), 39
on_error() (anomalydetection.backend.sink.repository.RepositorySink method), 40
on_error() (anomalydetection.backend.sink.stream.StreamSink method), 41
on_error() (anomalydetection.backend.sink.websocket.WebSocketSink method), 42
on_next() (anomalydetection.backend.sink.BaseSink method), 39
on_next() (anomalydetection.backend.sink.repository.RepositorySink method), 40
on_next() (anomalydetection.backend.sink.stream.StreamSink method), 41
on_next() (anomalydetection.backend.sink.websocket.WebSocketSink method), 42
OutputMessage (class in anomalydetection.backend.entities.output_message), 29
OutputMessageHandler (class in anomalydetection.backend.entities.output_message), 30

P

P50 (anomalydetection.backend.stream.agg.functions.AggregationFunction attribute), 43
P75 (anomalydetection.backend.stream.agg.functions.AggregationFunction attribute), 43
P95 (anomalydetection.backend.stream.agg.functions.AggregationFunction attribute), 43
P99 (anomalydetection.backend.stream.agg.functions.AggregationFunction attribute), 43
parse_message() (anomalydetection.backend.entities.BaseMessageHandler class method), 28
parse_message() (anomalydetection.backend.entities.input_message.InputMessageHandler class method), 29
parse_message() (anomalydetection.backend.entities.output_message.OutputMessageHandler class method), 31
poll() (anomalydetection.backend.stream.BaseStreamConsumer method), 42
poll() (anomalydetection.backend.stream.kafka.KafkaStreamConsumer method), 44
poll() (anomalydetection.backend.stream.kafka.SparkKafkaStreamConsumer method), 45
predict() (anomalydetection.backend.engine.BaseEngine method), 33
predict() (anomalydetection.backend.engine.cad_engine.CADDetector method), 33
predict() (anomalydetection.backend.engine.ema_engine.EMADetector method), 34

predict() (anomalydetection.backend.engine.robust_z_engine.RobustDetector method), 34
process() (anomalydetection.backend.interactor.batch_engine.BatchEngineInteractor method), 36
push() (anomalydetection.backend.stream.BaseStreamProducer method), 43
push() (anomalydetection.backend.stream.kafka.KafkaStreamProducer method), 44

R

register_engine() (anomalydetection.backend.engine.builder.EngineBuilderFactory class method), 33
RepositoryBuilderFactory (class in anomalydetection.backend.repository.builder), 37
RepositorySink (class in anomalydetection.backend.sink.repository), 40
RobustDetector (class in anomalydetection.backend.engine.robust_z_engine), 34
RobustDetectorBuilder (class in anomalydetection.backend.engine.builder), 33

S

set() (anomalydetection.backend.engine.builder.BaseEngineBuilder method), 31
set() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set() (anomalydetection.backend.engine.builder.EMADetectorBuilder method), 32
set() (anomalydetection.backend.engine.builder.RobustDetectorBuilder method), 33
set() (anomalydetection.backend.repository.builder.BaseRepositoryBuilder method), 37
set() (anomalydetection.backend.repository.builder.SQLiteBuilder method), 37
set_database() (anomalydetection.backend.repository.builder.SQLiteBuilder method), 37
set_max_active_neurons_num() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set_max_left_semi_contexts_length() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set_max_value() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set_min_value() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set_num_norm_value_bits() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set_rest_period() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set_threshold() (anomalydetection.backend.engine.builder.CADDetectorBuilder method), 32
set_threshold() (anomalydetection.backend.engine.builder.EMADetectorBuilder method), 32
set_threshold() (anomalydetection.backend.engine.builder.RobustDetectorBuilder method), 33
set_window() (anomalydetection.backend.engine.builder.EMADetectorBuilder method), 32
set_window() (anomalydetection.backend.engine.builder.RobustDetectorBuilder method), 33
SparkKafkaStreamConsumer (class in anomalydetection.backend.stream.kafka), 44
SQLiteBuilder (class in anomalydetection.backend.repository.builder), 37
SQLiteRepository (class in anomalydetection.backend.repository.sqlite), 38
step() (anomalydetection.backend.engine.cad_engine.CADDetector method), 34
StreamSink (class in anomalydetection.backend.sink.stream), 40
SUM (anomalydetection.backend.stream.agg.functions.AggregationFunctio attribute), 43

T

to_dict() (anomalydetection.backend.entities.input_message.InputMessage method), 28
to_dict() (anomalydetection.backend.entities.output_message.AnomalyResult method), 29
to_dict() (anomalydetection.backend.entities.output_message.OutputMessage method), 30
to_input() (anomalydetection.backend.entities.output_message.OutputMessage method), 30
to_json() (anomalydetection.backend.entities.input_message.InputMessage method), 28
to_notifier() (anomalydetection.backend.sink.BaseSink method), 39
to_notifier() (anomalydetection.backend.sink.repository.RepositorySink method), 40

to_notifier() (anomalydetection.backend.sink.stream.StreamSink method), 41
to_notifier() (anomalydetection.backend.sink.websocket.WebSocketSink method), 42
to_plain_dict() (anomalydetection.backend.entities.output_message.OutputMessage method), 30
type (anomalydetection.backend.engine.builder.CADDetectorBuilder attribute), 32
type (anomalydetection.backend.engine.builder.EMADetectorBuilder attribute), 32
type (anomalydetection.backend.engine.builder.RobustDetectorBuilder attribute), 33

U

unsubscribe() (anomalydetection.backend.stream.kafka.KafkaStreamConsumer method), 44
unsubscribe() (anomalydetection.backend.stream.kafka.SparkKafkaStreamConsumer method), 45
update_contexts_and_get_active() (anomalydetection.backend.engine.cad_engine.ContextOperator method), 34

V

validate_message() (anomalydetection.backend.entities.BaseMessageHandler class method), 28
validate_message() (anomalydetection.backend.entities.input_message.InputMessageHandler class method), 29
validate_message() (anomalydetection.backend.entities.output_message.OutputMessageHandler class method), 31

W

WebSocketSink (class in anomalydetection.backend.sink.websocket), 41