

---

**aisquared**

**The AI Squared Team**

**Nov 28, 2023**



# DOCUMENTATION

|          |                             |           |
|----------|-----------------------------|-----------|
| <b>1</b> | <b>Installation</b>         | <b>3</b>  |
| 1.1      | aisquared . . . . .         | 3         |
| 1.1.1    | aisquared package . . . . . | 3         |
| 1.1.1.1  | Subpackages . . . . .       | 3         |
| 1.1.1.2  | Module contents . . . . .   | 70        |
| <b>2</b> | <b>Changelog</b>            | <b>71</b> |
|          | <b>Python Module Index</b>  | <b>75</b> |
|          | <b>Index</b>                | <b>77</b> |





This package contains utilities to interact with the AI Squared technology stack, particularly with developing and deploying models to the AI Squared Platform or other applications developed through the AI Squared JavaScript SDK.

**Current Production Version: 0.3.9**

[View this Documentation in PDF Format.](#)



## INSTALLATION

This package is available through [Pypi](#) and can be installed by running the following command:

```
pip install aisquared
```

Alternatively, the latest version of the software can be installed directly from GitHub using the following command:

```
pip install git+https://github.com/AISquaredInc/aisquared
```

### 1.1 aisquared

#### 1.1.1 aisquared package

##### 1.1.1.1 Subpackages

**aisquared.base package**

**Submodules**

**aisquared.base.BaseObject module**

**class** aisquared.base.BaseObject.**BaseObject**

Bases: object

Base class used for all other classes within the aisquared package. This class is not meant to be used by any end user of this package, but is rather used throughout this package as a parent class.

**to\_dict()** → dict

Get the object as a dictionary

**to\_json()** → str

Return the object as a json string

### aisquared.base.css module

### aisquared.base.endpoints module

NOT MEANT TO BE CALLED BY THE END USER - configuration parameters for the different endpoints in the platform

### aisquared.base.harvesting module

Some allowed configuration parameters - not designed to be directly called by the user

### aisquared.base.platform module

### aisquared.base.preprocessing module

Some allowed configuration parameters - not designed to be directly called by the user

### aisquared.base.rendering module

Some allowed configuration parameters - not meant to be directly called by the end user

### aisquared.base.stages module

Some allowed configuration parameters - not designed to be directly called by the user

## Module contents

The aisquared.base package contains both some basic objects that are used across the aisquared package backend and some objects which are designed to facilitate simple use cases of the technology.

## aisquared.config package

### Subpackages

### aisquared.config.analytic package

### Submodules

### aisquared.config.analytic.DeployedAnalytic module

```
class aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic(url: str, input_type: str,  
                                                                    secret: str = 'request', header:  
                                                                    dict | None = None,  
                                                                    api_key_header_name=None,  
                                                                    api_key_prefix=None)
```



Bases: *BaseObject*

Interaction with a remote analytic

Example usage:

```
>>> import aisquared
>>> analytic = aisquared.config.analytic.DeployedAnalytic(
    'analytic_url',
    'text'
)
>>> analytic.to_dict()
{'className': 'DeployedAnalytic',
 'params': {'url': 'analytic_url',
 'inputType': 'text',
 'secret': 'request',
 'header': None,
 'apiKeyHeaderName': None,
 'apiKeyPrefix': None}}
```

**property** api\_key\_header\_name

**property** api\_key\_prefix

**property** header

**property** input\_type

**property** secret

**to\_dict()** → dict

Get the object as a dictionary

**property** url

## aisquared.config.analytic.DeployedModel module

**class** aisquared.config.analytic.DeployedModel.**DeployedModel**(url: str, input\_type: str, secret: str = 'request', header: dict | None = None)

Bases: *BaseObject*

Interaction with a remote model

Example usage:

```
>>> import aisquared
>>> analytic = aisquared.config.analytic.DeployedModel(
    'model_url',
    'text'
)
>>> analytic.to_dict()
{'className': 'DeployedModel',
 'params': {'url': 'model_url',
 'inputType': 'text',
```

(continues on next page)

(continued from previous page)

```
'secret': 'request',  
'header': None}}
```

**property header**

**property input\_type**

**property secret**

**to\_dict()** → dict

Get the config object as a dictionary

**property url**

### aisquared.config.analytic.LocalAnalytic module

**class** aisquared.config.analytic.LocalAnalytic.**LocalAnalytic**(*path: str, input\_type: str, all: bool = False*)

Bases: *BaseObject*

Interaction with an analytic (lookup table) saved to the local file system

Example usage:

```
>>> import aisquared  
>>> analytic = aisquared.config.analytic.LocalAnalytic(  
    'analytic_path',  
    'text'  
)  
>>> analytic.to_dict()  
{'className': 'LocalAnalytic',  
 'params': {'path': 'analytic_path',  
            'inputType': 'text',  
            'all': False}}
```

**property all**

**property input\_type**

**property path**

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.analytic.LocalModel module

**class** aisquared.config.analytic.LocalModel.**LocalModel**(*path: str, input\_type: str*)

Bases: *BaseObject*

Interaction with a model currently saved to the local file system

Example usage:

```
>>> import aisquared
>>> analytic = aisquared.config.analytic.LocalModel(
    'model_path',
    'text'
)
>>> analytic.to_dict()
{'className': 'LocalModel',
 'params': {'path': 'model_path',
 'inputType': 'text'}}
```

**property** input\_type

**property** path

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.analytic.ReverseMLWorkflow module

```
class aisquared.config.analytic.ReverseMLWorkflow(bucket: str, filenames: list,
    column: str, input_type:
    str, filter_type: str,
    filter_by_columns: list |
    None = None, period: int |
    None = None, secret: str =
    "")
```

Bases: *BaseObject*

Interaction with a ReverseML CSV stored in S3

Example usage:

```
>>> import aisquared
>>> analytic = aisquared.config.analytic.ReverseMLWorkflow(
    'bucket_name',
    'file_name',
    'column_name',
    'text'
)
>>> analytic.to_dict()
{'className': 'ReverseMLWorkflow',
 'params': {'bucket': 'bucket_name',
 'fileNames': ['file_name'],
 'inputType': 'text',
 'column': 'column_name',
 'period': None,
 'secret': ''}}
```

**property** bucket

**property** column

**property** filenames

**property filter\_by\_columns**  
**property filter\_type**  
**property input\_type**  
**property period**  
**property secret**  
**to\_dict()** → dict  
Get the configuration object as a dictionary

## Module contents

The `aisquared.config.analytic` subpackage contains objects for packaging individual analytics.

### aisquared.config.feedback package

#### Submodules

#### aisquared.config.feedback.BinaryFeedback module

**class** `aisquared.config.feedback.BinaryFeedback.BinaryFeedback`(*label\_map: list*)

Bases: *BaseObject*

Feedback for binary classification

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.feedback.BinaryFeedback(['class1', 'class2'])
>>> my_obj.to_dict()
{'className': 'BinaryFeedback', 'params': {'labelMap': ['class1', 'class2']}}
```

**property label\_map**

**to\_dict()** → dict  
Return the object as a dictionary

#### aisquared.config.feedback.ModelFeedback module

**class** `aisquared.config.feedback.ModelFeedback.ModelFeedback`

Bases: *BaseObject*

Feedback object for questions and answers for an individual model.

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.feedback.ModelFeedback()
>>> my_obj.add_question(
```

(continues on next page)

(continued from previous page)

```

    'How is the model performing?',
    choices = ['very poorly', 'poorly', 'neutral', 'well', 'very well']
)
>>> my_obj.add_question(
    'Any additional feedback?',
    'text'
)
>>> my_obj.to_dict()
{'className': 'ModelFeedback',
 'params': {'questions': [{'question': 'How is the model performing?',
 'answerType': 'singleChoice',
 'choices': ['very poorly', 'poorly', 'neutral', 'well', 'very well']},
 {'question': 'Any additional feedback?', 'answerType': 'text'}]}}
```

**add\_question**(*question: str, answer\_type: str = 'singleChoice', choices: list = []*)

Add a question to be asked.

#### Parameters

- **question** (*str*) – The question to be asked.
- **answer\_type** (*str (default 'singleChoice')*) – One of either 'singleChoice', 'multiChoice', or 'text'
- **choices** (*list (default [])*) – The choices to be provided, if *answer\_type* is 'singleChoice' or 'multiChoice'

**to\_dict**() → dict

Return the object as a dictionary

### aisquared.config.feedback.MulticlassFeedback module

**class** aisquared.config.feedback.MulticlassFeedback.MulticlassFeedback(*label\_map: list*)

Bases: *BaseObject*

Feedback for multiclass classification

Example Usage:

```

>>> import aisquared
>>> my_obj = aisquared.config.feedback.MulticlassFeedback(['class1', 'class2',
↪ 'class3'])
>>> my_obj.to_dict()
{'className': 'MulticlassFeedback',
 'params': {'labelMap': ['class1', 'class2', 'class3']}}
```

**property** label\_map

**to\_dict**() → dict

Return the object as a dictionary

**aisquared.config.feedback.QualitativeFeedback module****class** aisquared.config.feedback.QualitativeFeedback.**QualitativeFeedback**Bases: *BaseObject*

Feedback object for questions and answers for individual predictions.

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.feedback.QualitativeFeedback()
>>> my_obj.add_question('Any additional feedback?', 'text')
>>> my_obj.to_dict()
{'className': 'QualitativeFeedback',
 'params': {'questions': [{'question': 'Any additional feedback?',
 'answerType': 'text'}]}}
```

**add\_question**(*question: str, answer\_type: str = 'singleChoice', choices: list = []*)

Add a question to be asked.

**Parameters**

- **question** (*str*) – The question to be asked.
- **answer\_type** (*str* (default 'singleChoice')) – One of either 'singleChoice', 'multiChoice', or 'text'
- **choices** (*list* (default [])) – The choices to be provided, if *answer\_type* is 'singleChoice' or 'multiChoice'

**to\_dict**() → dict

Return the object as a dictionary

**aisquared.config.feedback.RegressionFeedback module****class** aisquared.config.feedback.RegressionFeedback.**RegressionFeedback**Bases: *BaseObject*

Feedback for regression

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.feedback.RegressionFeedback()
>>> my_obj.to_dict()
{'className': 'RegressionFeedback', 'params': {}}
```

**to\_dict**() → dict

Return the object as a dictionary

## aisquared.config.feedback.SimpleFeedback module

**class** aisquared.config.feedback.SimpleFeedback.**SimpleFeedback**

Bases: *BaseObject*

Simple thumbs-up/thumbs-down feedback for predictions

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.feedback.SimpleFeedback()
>>> my_obj.to_dict()
{'className': 'SimpleFeedback', 'params': {}}
```

**to\_dict()** → dict

Return the object as a dictionary

## Module contents

The aisquared.config.feedback subpackage contains objects for configuring feedback in aisquared models.

## aisquared.config.harvesting package

### Submodules

## aisquared.config.harvesting.ChatbotHarvester module

**class** aisquared.config.harvesting.ChatbotHarvester.**ChatbotHarvester**(*max\_length: int | None = None*)

Bases: *BaseObject*

Object to harvest for a chatbot

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.harvesting.ChatbotHarvester(
    max_length = 128
)
>>> my_obj.to_dict()
{'className': 'ChatbotHarvester',
 'params': {'maxLength': '128',
 'inputType': 'text'}}
```

**property** max\_length

**to\_dict()**

Get the object as a dictionary

### aisquared.config.harvesting.ImageHarvester module

**class** aisquared.config.harvesting.ImageHarvester.**ImageHarvester**(*how: str = 'all'*)

Bases: *BaseObject*

Object to harvest images

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.harvesting.ImageHarvester()
>>> my_obj.to_dict()
{'className': 'ImageHarvester', 'params': {'how' : 'all'}}
```

**property** how

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.harvesting.InputHarvester module

**class** aisquared.config.harvesting.InputHarvester.**InputHarvester**(*input\_type: str = 'text',  
max\_length: int | None = None,  
features: list | None = None*)

Bases: *BaseObject*

Object to harvest user-input text

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.harvesting.InputHarvester()
>>> my_obj.to_dict()
{'className': 'InputHarvester',
 'params': {'inputType': 'text', 'maxLength': None, 'features': None}}
```

**property** features

**property** input\_type

**property** max\_length

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.harvesting.QueryParameterHarvester module

**class** aisquared.config.harvesting.QueryParameterHarvester.**QueryParameterHarvester**(*query\_keys:  
str | list,  
url\_locations:  
str | list,  
at-  
tributes:  
str | list*)



Bases: *BaseObject*

Harvester for Query Parameters

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.harvesting.QueryParameterHarvester(
    'test_key',
    'test_url',
    'test_attribute'
)
>>> my_obj.to_dict()
{'className': 'QueryParameterHarvester',
 'params': {'queryKeys': ['test_key'],
 'urlLocations': ['test_url'],
 'attributes': ['test_attribute']}
```

**property attributes**

**property query\_keys**

**to\_dict()** → dict

Get the configuration object as a dictionary

**property url\_locations**

## aisquared.config.harvesting.TextHarvester module

**class** aisquared.config.harvesting.TextHarvester.**TextHarvester**(*how: str = 'all', regex: str | None = None, flags: str = 'gu', body\_only: bool = False, keywords: str | list | None = None, limit: int | None = None*)

Bases: *BaseObject*

Object to harvest text

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.harvesting.TextHarvester(
    how = 'all',
    body_only = True
)
>>> my_obj.to_dict()
{'className': 'TextHarvester',
 'params': {'how': 'all',
 'regex': None,
 'flags': 'gu',
 'bodyOnly': True,
 'limit': None}}
```

**property body\_only**

**property flags**

**property how**

**property limit**

**property regex**

**to\_dict()** → dict

Get the configuration object as a dictionary

## Module contents

The `aisquared.config.harvesting` subpackage contains objects for configuring harvesting of data.

## aisquared.config.postprocessing package

### Submodules

## aisquared.config.postprocessing.BinaryClassification module

**class** `aisquared.config.postprocessing.BinaryClassification`.**BinaryClassification**(*label\_map:*  
*list,*  
*threshold:*  
*float = 0.5*)

Bases: *BaseObject*

Postprocessing configuration object for binary classification

Example usage

```
>>> import aisquared
>>> my_obj = aisquared.config.postprocessing.BinaryClassification(
    ['class1', 'class2']
)
>>> my_obj.to_dict()
{'className': 'BinaryClassification',
 'params': {'labelMap': ['class1', 'class2'], 'threshold': 0.5}}
```

**property label\_map**

**property threshold**

**to\_dict()** → dict

Get the configuration object as a dictionary

**aisquared.config.postprocessing.MulticlassClassification module**

**class** aisquared.config.postprocessing.MulticlassClassification.**MulticlassClassification**(*label\_map: list*)

Bases: *BaseObject*

Postprocessing configuration object for multiclass classification

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.postprocessing.MulticlassClassification(
    ['class1', 'class2', 'class3']
)
>>> my_obj.to_dict()
{'className': 'MulticlassClassification',
 'params': {'labelMap': ['class1', 'class2', 'class3']}}
```

**property** label\_map

**to\_dict()** → dict

Get the configuration object as a dictionary

**aisquared.config.postprocessing.ObjectDetection module**

**class** aisquared.config.postprocessing.ObjectDetection.**ObjectDetection**(*label\_map: list, threshold: float = 0.5*)

Bases: *BaseObject*

Postprocessing configuration object for object detection

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.postprocessing.ObjectDetection(
    ['class1', 'class2', 'class3']
)
>>> my_obj.to_dict()
{'className': 'ObjectDetection',
 'params': {'labelMap': ['class1', 'class2', 'class3'], 'threshold': 0.5}}
```

**property** label\_map

**property** threshold

**to\_dict()** → dict

Get the configuration object as a dictionary

**aisquared.config.postprocessing.Regression module**

```
class aisquared.config.postprocessing.Regression.Regression(min: int | float | None = None, max:
                                                         int | float | None = None, round: bool
                                                         = False)
```

Bases: *BaseObject*

Postprocessing configuration object for Regression

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.postprocessing.Regression(
    10,
    100
)
>>> my_obj.to_dict()
{'className': 'Regression', 'params': {'min': 10, 'max': 100, 'round': False}}
```

**property** max

**property** min

**property** round

**to\_dict()** → dict

Get the configuration object as a dictionary

**Module contents**

The aisquared.config.postprocessing subpackage contains objects for configuring how predictions are postprocessed.

**aisquared.config.preprocessing package****Subpackages****aisquared.config.preprocessing.image package****Submodules****aisquared.config.preprocessing.image.ImagePreprocessing module**

```
class aisquared.config.preprocessing.image.ImagePreprocessing.ImagePreprocessor(steps: list |
                                                                               None =
                                                                               None)
```

Bases: *BaseObject*

Preprocessor object for image data

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.image.ImagePreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.image.AddValue(255.0)
)
```

**add\_step(step)**

Add a step to the preprocessor object

**property step\_dict**

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.preprocessing.image.Steps module

**class** aisquared.config.preprocessing.image.Steps.**AddValue**(value: int | float)

Bases: *BaseObject*

Preprocessing step to add a value to all pixels in an image

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.image.ImagePreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.image.AddValue(255.0)
)
```

**to\_dict()** → dict

Get the configuration object as a dictionary

**property value**

**class** aisquared.config.preprocessing.image.Steps.**ConvertToColor**(color: str)

Bases: *BaseObject*

Preprocessing step to convert images to a color scheme

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.image.ImagePreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.image.ConvertToColor('RGB')
)
```

**property color**

**to\_dict()** → dict

Get the configuration object as a dictionary

**class** aisquared.config.preprocessing.image.Steps.**DivideValue**(value: int | float)

Bases: *BaseObject*

Preprocessing step to divide all pixels in an image by a value

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.image.ImagePreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.image.DivideValue(255.0)
)
```

**to\_dict()** → dict

Get the configuration object as a dictionary

**property value**

**class** aisquared.config.preprocessing.image.Steps.**MultiplyValue**(value: int | float)

Bases: *BaseObject*

Preprocessing step to multiply all pixels in an image by a value

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.image.ImagePreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.image.MultiplyValue(2.0)
)
```

**to\_dict()** → dict

Get the configuration object as a dictionary

**property value**

**class** aisquared.config.preprocessing.image.Steps.**Resize**(size: list, method: str = 'bilinear',  
preserve\_aspect\_ratio: bool = False)

Bases: *BaseObject*

Preprocessing step to resize an image

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.image.ImagePreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.image.Resize([100, 100])
)
```

**property method**

**property preserve\_aspect\_ratio**

**property size**

**to\_dict()** → dict

Get the configuration object as a dictionary

**class** aisquared.config.preprocessing.image.Steps.**SubtractValue**(value: int | float)

Bases: *BaseObject*

Preprocessing step to subtract a value from all pixels in an image

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.image.ImagePreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.image.SubtractValue(255.0)
)
```

**to\_dict()** → dict

Get the configuration object as a dictionary

**property value**

## Module contents

The aisquared.config.preprocessing.image subpackage contains objects for configuring image preprocessing.

## aisquared.config.preprocessing.tabular package

### Submodules

#### aisquared.config.preprocessing.tabular.Steps module

**class** aisquared.config.preprocessing.tabular.Steps.**DropColumn**(column: int)

Bases: *BaseObject*

Drop a column from tabular data

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.tabular.TabularPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.tabular.DropColumn(
        3
    )
)
```

**property column**

**to\_dict()** → dict

Get the configuration object as a dictionary

**class** aisquared.config.preprocessing.tabular.Steps.**MinMax**(mins: list, maxs: list, columns: list | None = None)

Bases: *BaseObject*

Min-Max Scaling preprocessing step

Min-Max Scaling takes all associated columns and maps values relative to the minimum and maximum values of the training data.

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.tabular.TabularPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.tabular.MinMax(
        [0, 1.1, 2],
        [0.2, 14, 18.3]
    )
)
```

**property columns**

**property maxs**

**property mins**

**to\_dict()** → dict

Get the configuration object as a dictionary

**class** aisquared.config.preprocessing.tabular.Steps.**OneHot**(*column: int, values: list*)

Bases: [BaseObject](#)

One Hot encoding preprocessing step

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.tabular.TabularPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.tabular.OneHot(
        6,
        ['one', 'two', 'three']
    )
)
```

**property column**

**to\_dict()** → dict

Get the configuration object as a dictionary

**property values**

**class** aisquared.config.preprocessing.tabular.Steps.**ZScore**(*means: list, stds: list, columns: int | list | None = None*)

Bases: [BaseObject](#)

Z-Score normalization preprocessing step

Z-Score normalization takes each supplied column value, subtracts that column's provided mean, and divides by the provided standard deviation.

Example usage:



```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.tabular.TabularPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.tabular.ZScore(
        [0, 1, 2],
        [0.2, 0.4, 0.6]
    )
)
```

**property columns**

**property means**

**property stds**

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.preprocessing.tabular.TabularPreprocessing module

**class** aisquared.config.preprocessing.tabular.TabularPreprocessing.**TabularPreprocessor**(*steps:*  
*list |*  
*None*  
*=*  
*None*)

Bases: *BaseObject*

Preprocessor object for tabular data

Example usage:

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.tabular.TabularPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.tabular.ZScore(
        [0, 1, 2],
        [0.2, 0.4, 0.6]
    )
)
```

**add\_step**(*step*)

Add a step to the preprocessor object

**to\_dict()**

Get the configuration object as a dictionary

## Module contents

The `aisquared.config.preprocessing.tabular` subpackage contains objects for preprocessing tabular data.

## aisquared.config.preprocessing.text package

### Submodules

#### aisquared.config.preprocessing.text.Steps module

**class** `aisquared.config.preprocessing.text.Steps.ConvertToCase`(*lowercase: bool = True*)

Bases: *BaseObject*

Text preprocessing object to convert inputs to all lowercase or all uppercase

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.text.TextPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.text.ConvertToCase()
)
```

**property** `lowercase`

**to\_dict()** → dict

Get the configuration object as a dictionary

**class** `aisquared.config.preprocessing.text.Steps.ConvertToVocabulary`(*vocabulary: dict*,  
*start\_character: int = 1*,  
*oov\_character: int = 2*,  
*max\_vocab: int | None = None*)

Bases: *BaseObject*

Text preprocessing object to convert tokens to integer vocabularies

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.text.TextPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.text.ConvertToVocabulary(
        {
            'test' : 3,
            'vocabulary' : 4
        }
    )
)
```

**property** `max_vocab`

**property** `oov_character`

**property start\_character**

**to\_dict()** → dict

Get the configuration object as a dictionary

**property vocabulary**

```
class aisquared.config.preprocessing.text.Steps.PadSequences(pad_character: int = 0, length: int = 128, pad_location: str = 'post', truncate_location: str = 'post')
```

Bases: *BaseObject*

Text preprocessing object to pad sequences

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.text.TextPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.text.PadSequences()
)
```

**property length**

**property pad\_character**

**property pad\_location**

**to\_dict()** → dict

Get the configuration object as a dictionary

**property truncate\_location**

```
class aisquared.config.preprocessing.text.Steps.RemoveCharacters(remove_digits: bool = True, remove_punctuation: bool = True)
```

Bases: *BaseObject*

Preprocessing step to remove characters from text

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.text.TextPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.text.RemoveCharacters()
)
```

**property remove\_digits**

**property remove\_punctuation**

**to\_dict()** → dict

Get the configuration object as a dictionary

```
class aisquared.config.preprocessing.text.Steps.Tokenize(split_sentences: bool = False, split_words: bool = True, token_pattern: str = '\\x08\\w\\w+\\x08')
```

Bases: *BaseObject*

Preprocessing Step to tokenize text

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.text.TextPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.text.Tokenize()
)
```

**property** `split_sentences`

**property** `split_words`

**to\_dict()** → dict

Get the configuration object as a dictionary

**property** `token_pattern`

**class** `aisquared.config.preprocessing.text.Steps.Trim`

Bases: *BaseObject*

Text preprocessing class to trim whitespace from text

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.text.TextPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.text.Trim()
)
```

**to\_dict()** → dict

Get the configuration object as a dictionary

## **aisquared.config.preprocessing.text.TextPreprocessing module**

**class** `aisquared.config.preprocessing.text.TextPreprocessing.TextPreprocessor`(*steps: list | None = None*)

Bases: *BaseObject*

Preprocessor object for natural language

Example usage:

```
>>> import aisquared
>>> preprocessor = aisquared.config.preprocessing.text.TextPreprocessor()
>>> preprocessor.add_step(
    aisquared.config.preprocessing.text.Tokenize()
)
```

**add\_step**(*step*)

Add a step to the preprocessor object

**property** `step_dict`

**to\_dict()** → dict

Get the configuration object as a dictionary

## Module contents

The `aisquared.config.preprocessing.text` subpackage contains objects for preprocessing text data.

## Module contents

**The `aisquared.config.preprocessing` subpackage contains utilities to configure the preprocessing of data in the data pipeline. It contains**

three separate subpackages, `aisquared.config.preprocessing.text`, `aisquared.config.preprocessing.image`, and `aisquared.config.preprocessing.tabular`, which configure the preprocessing of different types of data.

## aisquared.config.rendering package

### Submodules

#### aisquared.config.rendering.BarChartRendering module

```
class aisquared.config.rendering.BarChartRendering.BarChartRendering(label: str, id: str,
                                                                    chart_name: str,
                                                                    container_id: str,
                                                                    prediction_name_key: str,
                                                                    prediction_value_key: str,
                                                                    prediction_name_value:
                                                                    str, display_legend: bool,
                                                                    legend_icon: str,
                                                                    labels_key: str | None =
                                                                    None, width: str = 'auto',
                                                                    height: str = 'auto',
                                                                    xOffset: str = '0', yOffset:
                                                                    str = '0', labels: list | None
                                                                    = None,
                                                                    consolidate_rows: bool =
                                                                    True, css_params: dict |
                                                                    None = None)
```

Bases: *BaseObject*

Rendering class for rendering a Bar Chart

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.BarChartRendering(
    'my_label',
    'my_id',
    'my_bar_chart',
    'my_container_id',
```

(continues on next page)

(continued from previous page)

```

        'name',
        'value',
        'name_value',
        True,
        'circle'
    )
    >>> my_obj.to_dict()
    {'className': 'BarChartRendering',
     'label': 'my_label',
     'params': {'id': 'my_id',
                'chartName': 'my_bar_chart',
                'containerId': 'my_container_id',
                'displayLegend': True,
                'legendIcon': 'circle',
                'width': 'auto',
                'height': 'auto',
                'xOffset': '0',
                'yOffset': '0',
                'datasource': [{'labels': None,
                                'labelsKey': None,
                                'consolidateRows': True,
                                'predictionNameKey': 'name',
                                'predictionValueKey': 'value',
                                'predictionNameValue': 'name_value'}]}}
```

**to\_dict()** → dict

Get the configuration object as a dictionary

**aisquared.config.rendering.ContainerRendering module**

```

class aisquared.config.rendering.ContainerRendering.ContainerRendering(label: str, id: str,
                                                                       query_selector: str,
                                                                       position: str =
                                                                       'absolute',
                                                                       static_position: str |
                                                                       None = None, width:
                                                                       str = 'auto', height: str
                                                                       = 'auto', display: str =
                                                                       'flex', xOffset: str = '0',
                                                                       yOffset: str = '0',
                                                                       orientation: str =
                                                                       'column', css_params:
                                                                       dict | None = None)
```

Bases: *BaseObject*

Rendering for a container

Example usage:

```

>>> import aisquared
>>> my_obj = aisquared.config.rendering.ContainerRendering(
    'my container',
```

(continues on next page)

(continued from previous page)

```

    'myContainerID',
    "[data-id='tabpanel-general']"
)
>>> my_obj.to_dict()
{'className': 'ContainerRendering',
 'label': 'my container',
 'params': {'id': 'myContainerID',
 'width': 'auto',
 'height': 'auto',
 'display': 'flex',
 'xOffset': '0',
 'yOffset': '0',
 'position': 'absolute',
 'orientation': 'column',
 'querySelector': "[data-id='tabpanel-general']",
 'staticPosition': None}}
```

**property display**

**property height**

**property id**

**property label**

**property orientation**

**property position**

**property query\_selector**

**property static\_position**

**to\_dict()** → dict

Get the configuration object as a dictionary

**property width**

**property xOffset**

**property yOffset**

## aisquared.config.rendering.CustomRendering module

```

class aisquared.config.rendering.CustomRendering.CustomRendering(id: str, content_html: str,
                                                                    content_script: str,
                                                                    content_style: str,
                                                                    query_selector: str | None =
                                                                    None)
```

Bases: *BaseObject*

**property content\_html**

**property content\_script**

**property content\_style**

**property id**

**property query\_selector**

**to\_dict()**

Get the object as a dictionary

## aisquared.config.rendering.DashboardRendering module

THIS MODULE IS IN DEVELOPMENT AND NOT STABLE. PLEASE USE WITH CAUTION AND DO NOT USE FOR ANY PRODUCTION WORKLOADS

**class** aisquared.config.rendering.DashboardRendering.DashboardRendering

Bases: *BaseObject*

THIS CLASS IS IN DEVELOPMENT AND IS NOT STABLE. PLEASE USE WITH CAUTION AND DO NOT USE FOR ANY PRODUCTION WORKLOADS

**add\_bar\_chart**(*container\_id: str, prediction\_name\_key: str, prediction\_value\_key: str, prediction\_name\_value: str, chart\_colors: list, chart\_labels: list, width: str = 'auto', height: str = 'auto', xOffset: str = '0', yOffset: str = '0', id: str | None = None, label: str | None = None, chart\_name: str | None = None*)

**add\_container**(*query\_selector: str, width: str = 'auto', height: str = 'auto', display: str = 'flex', xOffset: str = '0', yOffset: str = '0', position: str = '', orientation: str = 'column', id: str | None = None, label: str | None = None*)

**add\_doughnut\_chart**(*container\_id: str, prediction\_name\_key: str, prediction\_value\_key: str, prediction\_name\_value: str, chart\_colors: list, chart\_labels: list, display\_legend: bool = True, legend\_icon: str = 'circle', width: str = 'auto', height: str = 'auto', xOffset: str = '0', yOffset: str = '0', id: str | None = None, label: str | None = None, chart\_name: str | None = None*)

**add\_html\_tag**(*container\_id: str, html\_content: str, prediction\_name\_key: str = '', prediction\_value\_key: str = '', prediction\_name\_value: str = '', extra\_content\_tag: str = 'strong', injection\_action: str = 'prepend', id: str | None = None, content: str = '', label: str | None = None*)

**add\_line\_chart**(*container\_id: str, prediction\_name\_key: str, prediction\_value\_key: str, prediction\_name\_value: str, chart\_colors: list, chart\_labels: list, width: str = 'auto', height: str = 'auto', xOffset: str = '0', yOffset: str = '0', id: str | None = None, label: str | None = None, chart\_name: str | None = None*)

**add\_pie\_chart**(*container\_id: str, prediction\_name\_key: str, prediction\_value\_key: str, prediction\_name\_value: str, chart\_colors: list, chart\_labels: list, display\_legend: bool = True, legend\_icon: str = 'circle', width: str = 'auto', height: str = 'auto', xOffset: str = '0', yOffset: str = '0', id=None, label: str | None = None, chart\_name: str | None = None*)

**add\_table**(*container\_id: str, prediction\_name\_key: str, prediction\_value\_key: str, prediction\_name\_values: str, table\_name: str = '', id: str | None = None, label: str | None = None*)

**property steps**

**to\_dict()**

Get the object as a dictionary



**aisquared.config.rendering.DashboardReplacementRendering module**

```
class aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering(anchor_selector: str,
where_replace: str
=
",
la-
bel:
str
=
")
```

Bases: *BaseObject*

Rendering for dashboard replacement

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.DashboardReplacementRendering(
    'test_anchor_selector'
)
>>> my_obj.to_dict()
{'className': 'DashboardReplacementRendering',
'label': '',
'params': {'anchorSelector': 'test_anchor_selector', 'whereReplace': ''}}
```

**property** anchor\_selector

**property** label

**to\_dict()** → dict

Get the configuration object as a dictionary

**property** where\_replace

**aisquared.config.rendering.DocumentRendering module**

```
class aisquared.config.rendering.DocumentRendering.DocumentRendering(prediction_key: str =
'className', words: list |
dict | str | None = None,
documents: list | dict | str |
None = None,
include_probability: bool
= False, probability_key:
str = 'probability',
underline_color: str =
'blue', classes: list | None
= None, threshold_key: str
| None = None,
threshold_value: int | float
| None = None)
```

Bases: *BaseObject*

Object which dictates how to render predictions on entire documents

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.DocumentRendering()
>>> my_obj.to_dict()
{'className': 'DocumentRendering',
 'params': {'predictionKey': 'className',
 'words': None,
 'documents': None,
 'includeProbability': False,
 'probabilityKey': 'probability',
 'underlineColor': 'blue',
 'classes': None,
 'thresholdKey': None,
 'thresholdValue': None}}
```

**property classes**

**property documents**

**property include\_probability**

**property prediction\_key**

**property probability\_key**

**property threshold\_key**

**property threshold\_value**

**to\_dict()** → dict

Get the configuration object as a dictionary

**property underline\_color**

**property words**

**aisquared.config.rendering.DoughnutChartRendering module**

```

class aisquared.config.rendering.DoughnutChartRendering.DoughnutChartRendering(label: str, id:
    str,
    chart_name:
    str,
    container_id:
    str, prediction_name_key:
    str, prediction_value_key:
    str, prediction_name_value:
    str, display_legend:
    bool,
    legend_icon:
    str,
    labels_key:
    str | None =
    None, width:
    str = 'auto',
    height: str =
    'auto',
    xOffset: str =
    '0', yOffset:
    str = '0',
    labels: list |
    None =
    None,
    consolidate_rows:
    bool = True,
    css_params:
    dict | None =
    None)

```

Bases: *BaseObject*

Rendering class for rendering a Doughnut Chart

Example usage:

```

>>> import aisquared
>>> my_obj = aisquared.config.rendering.DoughnutChartRendering(
    'my_label',
    'my_id',
    'my_doughnut_chart',
    'my_container_id',
    'name',
    'value',
    'name_value',
    True,
    'circle'
)
>>> my_obj.to_dict()
{'className': 'DoughnutChartRendering',

```

(continues on next page)

(continued from previous page)

```
'label': 'my_label',
'params': {'id': 'my_id',
'chartName': 'my_doughnut_chart',
'containerId': 'my_container_id',
'displayLegend': True,
'legendIcon': 'circle',
'width': 'auto',
'height': 'auto',
'xOffset': '0',
'yOffset': '0',
'datasource': [{'labels': None,
'labelsKey': None,
'consolidateRows': True,
'predictionNameKey': 'name',
'predictionValueKey': 'value',
'predictionNameValue': 'name_value'}]}
```

**to\_dict()** → dict

Get the configuration object as a dictionary

**aisquared.config.rendering.FilterRendering module**

**class** aisquared.config.rendering.FilterRendering.**FilterRendering**(*source: str, key: str, qualifier: str, value: list | str | int | float*)

Bases: *BaseObject*

Object which dictates how predictions are to be passed to downstream analytics

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.FilterRendering(
    'inputs',
    'key',
    'gt',
    0.2
)
>>> my_obj.to_dict()
{'className': 'FilterRendering',
'params': {'source': 'inputs', 'key': 'key', 'qualifier': 'gt', 'value': 0.2}}
```

**property** key**property** qualifier**property** source**to\_dict()** → dict

Get the configuration object as a dictionary

**property** value

**aisquared.config.rendering.HTMLTagRendering module**

```
class aisquared.config.rendering.HTMLTagRendering.HTMLTagRendering(label: str, id: str,
                                                                    container_id: str,
                                                                    html_content: str,
                                                                    extra_content_tag: str,
                                                                    injection_action: str,
                                                                    prediction_name_key: str,
                                                                    prediction_value_key: str,
                                                                    prediction_name_value: str,
                                                                    content: str = "", css_params:
                                                                    dict | None = None)
```

Bases: *BaseObject*

Rendering for HTML tags

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.HTMLTagRendering(
    'my HTML tag',
    'MyHTMLTagRenderingID',
    'MyContainerID',
    '<p>Example Text</p>',
    'extra_tag',
    'append',
    'name_key',
    'value_key',
    'name_value'
)
>>> my_obj.to_dict()
{'className': 'HTMLTagRendering',
 'label': 'my HTML tag',
 'params': {'id': 'MyHTMLTagRenderingID',
            'containerId': 'MyContainerID',
            'htmlContent': '<p>Example Text</p>',
            'extraContentTag': 'extra_tag',
            'injectionAction': 'append',
            'predictionNameKey': 'name_key',
            'predictionValueKey': 'value_key',
            'predictionNameValue': 'name_value',
            'content': ''}}
```

**to\_dict()** → dict

Return the configuration object as a dictionary

**aisquared.config.rendering.ImageRendering module**

```
class aisquared.config.rendering.ImageRendering.ImageRendering(color: str = 'blue', thickness: str =  
    '5', placement: str = 'bottomleft',  
    include_probability: bool = False,  
    badge_color: str = 'white',  
    font_color: str = 'black', font_size:  
    str = '5', classes: list | None =  
    None, threshold_key: str | None =  
    None, threshold_value: int | float |  
    None = None)
```

Bases: *BaseObject*

Object which dictates how to render images

Example usage:

```
>>> import aisquared  
>>> my_obj = aisquared.config.rendering.ImageRendering()  
>>> my_obj.to_dict()  
{'className': 'ImageRendering',  
 'params': {'color': 'blue',  
            'thickness': '5',  
            'placement': 'bottomleft',  
            'includeProbability': False,  
            'badgeColor': 'white',  
            'fontColor': 'black',  
            'fontSize': '5',  
            'classes': None,  
            'thresholdKey': None,  
            'thresholdValue': None}}
```

**property** badge\_color

**property** classes

**property** color

**property** font\_color

**property** font\_size

**property** include\_probability

**property** placement

**property** thickness

**property** threshold\_key

**property** threshold\_value

**to\_dict()** → dict

Get the configuration object as a dictionary

**aisquared.config.rendering.LineChartRendering module**

```
class aisquared.config.rendering.LineChartRendering(label: str, id: str,
                                                    chart_name: str,
                                                    container_id: str,
                                                    prediction_name_key:
                                                    str,
                                                    prediction_value_key:
                                                    str, predic-
                                                    tion_name_value: str,
                                                    display_legend: bool,
                                                    legend_icon: str,
                                                    labels_key: str, width:
                                                    str = 'auto', height: str =
                                                    'auto', xOffset: str =
                                                    '0', yOffset: str = '0',
                                                    labels: list | None =
                                                    None,
                                                    consolidate_rows: bool
                                                    = True, css_params:
                                                    dict | None = None)
```

Bases: *BaseObject*

Rendering class for rendering a Line Chart

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.LineChartRendering(
    'my_label',
    'my_id',
    'my_line_chart',
    'my_container_id',
    'name',
    'value',
    'name_value',
    True,
    'circle',
    'labels'
)
>>> my_obj.to_dict()
{'className': 'LineChartRendering',
 'label': 'my_label',
 'params': {'id': 'my_id',
 'chartName': 'my_line_chart',
 'containerId': 'my_container_id',
 'displayLegend': True,
 'legendIcon': 'circle',
 'width': 'auto',
 'height': 'auto',
 'xOffset': '0',
 'yOffset': '0',
 'datasource': [{'labels': None,
 'labelsKey': 'labels',
```

(continues on next page)

(continued from previous page)

```
'consolidateRows': True,
'predictionNameKey': 'name',
'predictionValueKey': 'value',
'predictionNameValue': 'name_value']}]}}
```

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.rendering.ObjectRendering module

```
class aisquared.config.rendering.ObjectRendering.ObjectRendering(color: str = 'blue', thickness:
    str = '5', placement: str =
    'bottomleft',
    include_probability: bool =
    False, badge_color: str =
    'white', font_color: str = 'black',
    font_size: str = '5')
```

Bases: *BaseObject*

Object which dictates how to render object detection in images

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.ObjectRendering()
>>> my_obj.to_dict()
{'className': 'ObjectRendering',
'params': {'color': 'blue',
'thickness': '5',
'placement': 'bottomleft',
'includeProbability': False,
'badgeColor': 'white',
'fontColor': 'black',
'fontSize': '5'}}
```

**property badge\_color****property color****property font\_color****property font\_size****property include\_probability****property placement****property thickness****to\_dict()** → dict

Get the configuration object as a dictionary



**aisquared.config.rendering.PieChartRendering module**

```
class aisquared.config.rendering.PieChartRendering(label: str, id: str,  

chart_name: str,  

container_id: str,  

prediction_name_key: str,  

prediction_value_key: str,  

prediction_name_value:  

str, display_legend: bool,  

legend_icon: str,  

labels_key: str | None =  

None, width: str = 'auto',  

height: str = 'auto',  

xOffset: str = '0', yOffset:  

str = '0', labels: list | None  

= None,  

consolidate_rows: bool =  

True, css_params: dict |  

None = None)
```

Bases: *BaseObject*

Rendering class for rendering a Pie Chart

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.PieChartRendering(
    'my_label',
    'my_id',
    'my_doughnut_chart',
    'my_container_id',
    'name',
    'value',
    'name_value',
    True,
    'circle'
)
>>> my_obj.to_dict()
{'className': 'PieChartRendering',
 'label': 'my_label',
 'params': {'id': 'my_id',
 'chartName': 'my_doughnut_chart',
 'containerId': 'my_container_id',
 'displayLegend': True,
 'legendIcon': 'circle',
 'width': 'auto',
 'height': 'auto',
 'xOffset': '0',
 'yOffset': '0',
 'datasource': [{'labels': None,
 'labelsKey': None,
 'consolidateRows': True,
 'predictionNameKey': 'name',
 'predictionValueKey': 'value',
```

(continues on next page)

(continued from previous page)

```
'predictionNameValue': 'name_value']}]}
```

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.rendering.SOSRendering module

**class** aisquared.config.rendering.SOSRendering.SOSRendering(*can\_toggle: bool, label: str = ""*)

Bases: *BaseObject*

Rendering of an SOS dashboard

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.SOSRendering(True)
>>> my_obj.to_dict()
{'className': 'SOSRendering', 'label': '', 'params': {'canToggle': True}}
```

**property can\_toggle**

**property label**

**to\_dict()** → dict

Get the configuration object as a dictionary

### aisquared.config.rendering.TableRendering module

**class** aisquared.config.rendering.TableRendering.TableRendering(*label: str, id: str, container\_id: str, prediction\_name\_key: str, prediction\_value\_key: str, prediction\_name\_values: str, table\_name: str = "", css\_params: dict | None = None*)

Bases: *BaseObject*

Class for rendering tables

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.TableRendering(
    'my table',
    'MyTableID',
    'MyContainerID',
    'name_key',
    'value_key',
    'name_values'
)
>>> my_obj.to_dict()
{'className': 'TableRendering',
 'label': 'my table',
```

(continues on next page)

(continued from previous page)

```
'params': {'id': 'MyTableID',
'containerId': 'MyContainerID',
'predictionNameKey': 'name_key',
'predictionValueKey': 'value_key',
'predictionNameValues': 'name_values',
'tableName': ''}}
```

**to\_dict()** → dict

Get the configuration object as a dictionary

## aisquared.config.rendering.TextRendering module

**class** aisquared.config.rendering.TextRendering.**TextRendering**(prediction\_value\_key: str | None = None)

Bases: *BaseObject*

Class for rendering text

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.TextRendering(
    prediction_value_key = 'my_key'
)
>>> my_obj.to_dict()
{'className': 'TextRendering',
'params': {'predictionValueKey': 'my_key'}}
```

**property** prediction\_value\_key

**to\_dict()**

Get the object as a dictionary

## aisquared.config.rendering.WordRendering module

**class** aisquared.config.rendering.WordRendering.**WordRendering**(word\_list: str = 'input', result\_key: str | None = None, content\_key: str | None = None, badge\_shape: str = 'star', badge\_color: str = 'blue', classes: list | None = None, threshold\_key: str | None = None, threshold\_value: int | float | None = None, position: str = 'after')

Bases: *BaseObject*

Object for rendering badges on individual words

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.WordRendering()
>>> my_obj.to_dict()
```

(continues on next page)

(continued from previous page)

```
{'className': 'WordRendering',  
'params': {'wordList': 'input',  
'resultKey': None,  
'contentKey': None,  
'badgeShape': 'star',  
'badgeColor': 'blue',  
'classes': None,  
'thresholdKey': None,  
'thresholdValue': None  
'position': 'after'}}
```

**property** badge\_color

**property** badge\_shape

**property** classes

**property** content\_key

**property** position

**property** result\_key

**property** threshold\_key

**property** threshold\_value

**to\_dict()** → dict

Get the configuration object as a dictionary

**property** word\_list

## aisquared.config.rendering.utils module

aisquared.config.rendering.utils.**save\_default\_css()**

Save default CSS so that default CSS can be edited and automatically utilized with changes

### Notes

- Saves all CSS files to the `~/.aisquared/` directory

## Module contents

The aisquared.config.rendering subpackage contains objects for configuring how rendering of predictions is to occur.

## Submodules

### aisquared.config.CustomObject module

**class** aisquared.config.CustomObject.**CustomObject**(*class\_name: str, top\_level\_kwargs: dict | None = None, \*\*kwargs*)

Bases: [BaseObject](#)

Custom class that allows the user to define custom classes for configuration

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.base.CustomObject(
    'MyClass',
    key1 = 'foo',
    key2 = 'bar'
)
>>> my_obj.to_dict()
{'className': 'MyClass', 'params': {'key1': 'foo', 'key2': 'bar'}}
```

**to\_dict()** → dict

Get the object as a dictionary

### aisquared.config.GraphConfiguration module

**class** aisquared.config.GraphConfiguration.**GraphConfiguration**(*name: str, stage: str = 'experimental', version: int | None = None, description: str = "", mlflow\_uri: str | None = None, mlflow\_user: str | None = None, mlflow\_token: str | None = None, owner: str | None = None, url: str = '\*', auto\_run: bool = False, documentation\_link: str = ""*)

Bases: [BaseObject](#)

Configuration object for deploying a set of processing steps and/or analytics as a dependency graph

**add\_node**(*step: BaseObject, dependencies: int | list | None = None*) → int

Add a node to the configuration graph

#### Parameters

- **step** (*aisquared configuration step*) – The step to add
- **dependencies** (*int, list of int, or None*) – The ids of nodes which must be run before the added node

#### Returns

**node\_id** – The integer id of the node that is added

#### Return type

int

**property auto\_run**

**compile**(filename: str | None = None, dtype: str | None = None) → None

Compile the object into a '.air' file, which can then be dragged and dropped into applications using the AI Squared JavaScript SDK

**Parameters**

- **filename** (*path-like or None (default None)*) – Filename to compile to. If None, defaults to '{NAME}.air', where {NAME} is the name of the analytic
- **dtype** (*str or None (default None)*) – The datatype to use for the model weights when using a Keras model. If None, defaults to 'float32'

**property description**

**property documentation\_link**

**get\_filenames()** → list

Get filenames for all models in the configuration

**property mlflow\_token**

**property mlflow\_uri**

**property mlflow\_user**

**property name**

**property owner**

**property stage**

**to\_dict()** → dict

Get the object as a dictionary

**property url**

**property version**

**aisquared.config.ModelConfiguration module**

```
class aisquared.config.ModelConfiguration.ModelConfiguration(name: str, harvesting_steps:
    BaseObject | list | None = None,
    preprocessing_steps: BaseObject |
    list | None = None, analytic:
    BaseObject | list | None = None,
    postprocessing_steps: BaseObject |
    list | None = None, rendering_steps:
    BaseObject | list | None = None,
    feedback_steps: BaseObject | list |
    None = None, stage: str =
    'experimental', version: int | None =
    None, description: str = "",
    mlflow_uri: str | None = None,
    mlflow_user: str | None = None,
    mlflow_token: str | None = None,
    owner: str | None = None, url: str =
    '*', auto_run: bool = False,
    documentation_link: str = "")
```

Bases: *BaseObject*

Configuration object for deploying a model or analytic

**property analytic**

**property analytic\_dict**

**property auto\_run**

**compile**(filename: str | None = None, dtype: str | None = None) → None

Compile the object into a '.air' file, which can then be dragged and dropped into applications using the AI Squared JavaScript SDK

#### Parameters

- **filename** (*path-like or None (default None)*) – Filename to compile to. If None, defaults to '{NAME}.air', where {NAME} is the name of the analytic
- **dtype** (*str or None (default None)*) – The datatype to use for the model weights. If None, defaults to 'float32'

**property description**

**property documentation\_link**

**property feedback\_dict**

**property feedback\_steps**

**get\_model\_filenames**() → list

Get filenames for all models in the configuration

**property harvester\_dict**

**property harvesting\_steps**

**property mlflow\_token**

**property mlflow\_uri**

```
property mlflow_user
property name
property owner
property postprocessor_dict
property postprocessing_steps
property preprocessor_dict
property preprocessing_steps
property render_dict
property rendering_steps
property stage
to_dict() → dict
    Get the object as a dictionary
property url
property version
```

## Module contents

The `aisquared.config` subpackage contains utilities and objects for packaging `aisquared` configuration steps and models. For in-depth examples of how to build out `.air` files using the utilities and classes in this library, please visit our GitHub repository at <https://github.com/AISquaredInc/airFiles>

## aisquared.logging package

### Module contents

The `aisquared.logging` subpackage contains utilities for performing experiments within `aisquared`. This functionality is inherited from MLFlow. Please see the MLFlow documentation at <https://mlflow.org>.

## aisquared.platform package

### Submodules

#### aisquared.platform.AISquaredAPIException module

```
exception aisquared.platform.AISquaredAPIException.AISquaredAPIException
    Bases: Exception
```



**aisquared.platform.AISquaredPlatformClient module**

**class** aisquared.platform.AISquaredPlatformClient.**AISquaredPlatformClient**(*use\_port: bool = False*)

Bases: object

Client for interacting with the AI Squared platform programmatically

When using the client for the first time, it is important to run the *client.login()* method. When doing so, the client will ask for any required information interactively.

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> # If you have never logged in before, run the following code:
>>> client.login()
>>> # Test connection
>>> client.test_connection()
True
```

**add\_users\_to\_group**(*group\_id: str, user\_ids: list, port: int = 8086, use\_port: bool | None = None*) → bool

Add users to a group

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.add_users_to_group('group_id', ['user_id_1', 'user_id_2'])
True
```

**Parameters**

- **group\_id** (*str*) – The group to add the users to
- **user\_ids** (*list of str*) – The IDs of the users to add
- **port** (*int (default 8086)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**success** – Returns True if operation was successful

**Return type**

bool

**property base\_url: str**

The base URL associated with the client

**create\_group**(*display\_name: str, role\_id: str, port: int = 8086, use\_port: bool | None = None*) → dict

Create a group in the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.create_group(
    'group display name',
    'role_id'
```

(continues on next page)

(continued from previous page)

```
)
*dictionary containing group information*
```

**Parameters**

- **display\_name** (*str*) – The display name of the group
- **role\_id** (*str*) – The role ID for the group
- **port** (*int (default 8086)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**group\_info** – Metadata about the created group

**Return type**

dict

```
create_user(user_name: str, given_name: str, family_name: str, email: str, role_id: str, active: bool =
    True, middle_name: str | None = None, company_id: str | None = None, password: str | None
    = None, port: int = 8085, use_port: bool | None = None) → dict
```

Create a user within the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.create_user(
    'user name',
    'given_name',
    'family_name',
    'user_email',
    'role_id'
)
*Dictionary with user information*
```

**Parameters**

- **user\_name** (*str*) – The display name of the user
- **given\_name** (*str*) – The user's first name
- **family\_name** (*str*) – The user's last name
- **email** (*str*) – The user's email
- **role\_id** (*str*) – The ID of the role to be given to the user
- **active** (*bool (default True)*) – Whether the user is active
- **middle\_name** (*str or None (default None)*) – The user's middle name
- **company\_id** (*str or None (default None)*) – The user's company ID
- **password** (*str or None (default None)*) – The user's password
- **port** (*int (default 8085)*) – The API port for the call. This can be handled automatically by the platform ALB

- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**user\_data** – Metadata about the user

**Return type**

dict

**delete\_group**(*group\_id, port=8086, use\_port: bool | None = None*) → bool

Delete a group from the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.delete_group('group_id')
True
```

**Parameters**

- **group\_id** (*str*) – The ID of the group to delete
- **port** (*int (default 8086)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**result** – Returns True if successful

**Return type**

bool

**delete\_model**(*id: str, port: int = 8080, use\_port: bool | None = None*) → bool

Delete a model

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.delete_model('model_id')
True
```

**Parameters**

- **id** (*str*) – The ID for the model
- **port** (*int (default 8080)*) – The API port for the model. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**success** – Whether the action was successful

**Return type**

bool

**delete\_user**(*user\_id: str, port: int = 8085, use\_port: bool | None = None*) → bool

Delete a user from the system

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.delete_user('user_id')
True
```

#### Parameters

- **user\_id** (*str*) – The user's ID
- **port** (*int (default 8085)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**result** – Returns True if the call is successful

#### Return type

bool

**get\_group**(*group\_id: str, port: int = 8086, use\_port: bool | None = None*) → dict

Retrieve information about a group

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_group('group_id')
*dictionary containing group data*
```

#### Parameters

- **group\_id** (*str*) – The ID of the group requested
- **port** (*int (default 8086)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**group\_info** – The information about the group

#### Return type

dict

**get\_group\_id\_by\_name**(*group\_name: str, port: int = 8083, use\_port: bool | None = None*) → str

Get the ID of a group by searching for its display name

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_group_id_by_name('Group Name')
*group_id*
```

#### Parameters

- **group\_name** (*str*) – The display name of the group
- **port** (*int* (*default* 8083)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (*default* *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**group\_id** – The ID of the group

**Return type**

*str*

**get\_model** (*id: str, port: int = 8080, use\_port: bool | None = None*) → *dict*

Retrieve a model configuration

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_model('model_id')
*JSON Response including model data and metadata*
```

**Parameters**

- **id** (*str*) – The ID for the model
- **port** (*int* (*default* 8080)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (*default* *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**model** – Metadata about the model coupled with the model's configuration information

**Return type**

*dictionary*

**get\_model\_id\_by\_name** (*model\_name: str, port: int = 8080, use\_port: bool | None = None*) → *str*

Retrieve a model's ID using the name of the model

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_model_id_by_name('my_awesome_model')
*model_id*
```

**Parameters**

- **model\_name** (*str*) – The name of the model
- **port** (*int* (*default* 8080)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (*default* *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**model\_id** – The model's ID

**Return type**

str

**get\_role\_id\_by\_role\_name**(*role\_name: str, port: int = 8086, use\_port: bool | None = None*) → str

Get the ID of a role by searching for its display name

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_role_id_by_role_name('Role Name')
*role_id*
```

**Parameters**

- **role\_name** (*str*) – The name of the role
- **port** (*int (default 8086)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns****role\_id** – The ID of the role**Return type**

str

**get\_user**(*user\_id: str, port: int = 8085, use\_port: bool | None = None*) → dict

Retrieve a user's information from the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_user('user_id')
*dictionary with results*
```

**Parameters**

- **user\_id** (*str*) – The ID of the user
- **port** (*int (default 8085)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns****user\_info** – The information about the user**Return type**

dict

**get\_user\_id\_by\_name**(*name: str, port: int = 8080, use\_port: bool | None = None*) → str

Get a user's ID from their display name

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_user_id_by_name('User Name')
*user_id*
```

**Parameters**

- **name** (*str*) – The display name of the user
- **port** (*int (default 8080)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**id** – The ID of the user

**Return type**

*str*

**property headers**

Headers used for authentication with the AI Squared Platform

**list\_group\_users**(*group\_id: str, as\_df: bool = True, port: int = 8083, use\_port: bool | None = None*) → *DataFrame | dict*

List users in a group

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_group_users('group_id')
*DataFrame with results*
```

**Parameters**

- **group\_id** (*str*) – The ID for the group
- **as\_df** (*bool (default True)*) – Whether to return the response as a pandas DataFrame
- **port** (*int (default 8083)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**users** – The response from the API

**Return type**

*pandas DataFrame or dictionary*

**list\_groups**(*max\_count: int = 100, as\_df: bool = True, port: int = 8083, use\_port: bool | None = None*) → *DataFrame | dict*

List all groups

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_groups()
*DataFrame with results*
```

**Parameters**

- **max\_count** (*int (default 100)*) – The maximum number of groups to return

- **as\_df** (*bool (default True)*) – Whether to return the result as a pandas DataFrame
- **port** (*int (default 8083)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**groups** – The response from the API

**Return type**

pandas DataFrame or dictionary

**list\_model\_feedback**(*model\_id: str, limit: int = 10, as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → dict | DataFrame

List feedback on a model

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_model_feedback('model_id')
*DataFrame with Results*
```

**Parameters**

- **model\_id** (*str*) – The ID of the model
- **limit** (*int (default 10)*) – The maximum number of feedback items to return
- **port** (*int (default 8080)*) – The API port to use. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**feedback** – The feedback

**Return type**

dict or pandas DataFrame

**list\_model\_prediction\_feedback**(*model\_id: str, as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → dict | DataFrame

List all feedback for a model

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_model_prediction_feedback('model_id')
*DataFrame with Results*
```

**Parameters**

- **model\_id** (*str*) – The ID of the model requested
- **as\_df** (*bool (default True)*) – Whether to return the results as a pandas DataFrame
- **port** (*int (default 8080)*) – The API port to use. This can be handled automatically by the platform ALB



- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**results** – The results from the platform

#### Return type

dict or pandas DataFrame

**list\_model\_usage\_metrics** (*model\_id: str, period: str = 'hourly', as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → dict | DataFrame

Get usage metrics for a model

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_model_usage_metrics('model_id')
*DataFrame with results*
```

#### Parameters

- **model\_id** (*str*) – The ID of the model
- **period** (*str (default 'hourly')*) – The period to group metrics into
- **as\_df** (*bool (default True)*) – Whether to return results as a pandas DataFrame
- **port** (*int (default 8080)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**results** – The results from the platform

#### Return type

pandas DataFrame or dict

**list\_model\_users** (*id: str, as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → DataFrame | dict

List users for a model

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_model_users('model_id')
*DataFrame with results*
```

#### Parameters

- **id** (*str*) – The ID for the model
- **as\_df** (*bool (default True)*) – Whether to return the response as a Pandas DataFrame
- **port** (*int (default 8080)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**model\_users** – The users for the model

**Return type**

pandas DataFrame or dictionary

**list\_models**(*as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → DataFrame | dict

List models within the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_models()
*DataFrame with results*
```

**Parameters**

- **as\_df** (*bool (default True)*) – Whether to return the response as a pandas DataFrame
- **port** (*default None*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**models** – The models

**Return type**

pandas DataFrame or dictionary

**list\_prediction\_feedback**(*prediction\_id: str, as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → DataFrame | dict

List prediction feedback given a prediction ID

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_prediction_feedback('prediction_id')
*DataFrame with results*
```

**Parameters**

- **prediction\_id** (*str*) – The prediction ID
- **as\_df** (*bool (default True)*) – Whether to return the results as a pandas DataFrame
- **port** (*int (default 8080)*) – The API port to use. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**Returns**

**results** – The results from the platform

**Return type**

pandas DataFrame or dict

**list\_roles**(*as\_df: bool = True, port: int = 8086, use\_port: bool | None = None*) → DataFrame | dict

List the roles available in the platform

Example usage:

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_roles()
*DataFrame with results*
```

#### Parameters

- **as\_df** (*bool (default True)*) – Whether to return the results as a pandas DataFrame
- **port** (*int (default 8086)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**roles** – The roles

#### Return type

pandas DataFrame or dict

**list\_user\_usage\_metrics**(*user\_id: str, period: str = 'hourly', as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → dict | DataFrame

Get usage metrics for a user

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.get_user_usage_metrics('user_id')
*DataFrame with results*
```

#### Parameters

- **user\_id** (*str*) – The ID of the user
- **period** (*str (default 'hourly')*) – The period to group metrics into
- **as\_df** (*bool (default True)*) – Whether to return results as a pandas DataFrame
- **port** (*int (default 8080)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**results** – The results from the platform

#### Return type

pandas DataFrame or dict

**list\_users**(*max\_count: int = 100, as\_df: bool = True, port: int = 8080, use\_port: bool | None = None*) → DataFrame | dict

List all users

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.list_users()
*DataFrame with results*
```

#### Parameters

- **max\_count** (*int (default 100)*) – The maximum number of users to return
- **as\_df** (*bool (default True)*) – Whether to return the data as a Pandas DataFrame
- **port** (*int (default 8080)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**users** – The response from the API

#### Return type

pandas DataFrame or dictionary

**login**(*url: str | None = None, port: int = 8080, username: str | None = None, password: str | None = None, use\_port: bool | None = None*) → None

Log in to the platform programmatically. If no url, username, or password are provided, logs in interactively

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.login()
Enter URL: https://platform.squared.ai
Enter Username: your.email@your_domain.com
Enter Password: <hidden>
```

#### Parameters

- **url** (*str or None (default None)*) – The URL for the platform API
- **port** (*int or None (default 8080)*) – The API port for the call. This can be handled automatically by the platform ALB
- **username** (*str or None (default None)*) – The username
- **password** (*str or None (default None)*) – The password
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

**property password:** str

The password associated with the client

**remove\_users\_from\_group**(*group\_id: str, user\_ids: list, port: int = 8086, use\_port: bool | None = None*) → bool

Remove users from a group

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.remove_users_from_group('group_id', ['user_id_1', 'user_id_2'])
True
```

#### Parameters

- **group\_id** (*str*) – The ID of the group
- **user\_ids** (*list of str*) – The IDs of the users to remove
- **port** (*int (default = 8086)*) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**success** – Returns True if successful

#### Return type

bool

**share\_model\_with\_group**(*model\_id: str, group\_id: str, port: int = 8080, use\_port: bool | None = None*)  
→ bool

Share a model with a group

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.share_model_with_group('model_id', 'group_id')
True
```

#### Parameters

- **model\_id** (*str*) – The ID for the model to be shared
- **group\_id** (*str*) – The ID for the group to be shared with. This can be handled automatically by the platform ALB
- **port** (*int (default 8080)*) – The API port to use. This can be handled automatically by the platform ALB
- **use\_port** (*bool or None (default None)*) – Whether to use port in URL formatting. If None, defaults to class value

#### Returns

**success** – Returns True if successful

#### Return type

bool

**share\_model\_with\_user**(*model\_id: str, user\_id: str, port: int = 8080, use\_port: bool | None = None*) →  
bool

Share a model with a user

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.share_model_with_user('model_id', 'user_id')
True
```

**Parameters**

- **model\_id** (*str*) – The ID for the model
- **user\_id** (*str*) – The ID for the user
- **port** (*int* (*default* 8080)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (*default* *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**success** – Whether the action was successful

**Return type**

*bool*

**test\_connection**(*port: int = 8080, use\_port: bool | None = None*) → *bool*

Test whether there is a healthy connection to the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.test_connection()
True
```

**Parameters**

- **port** (*int* (*default* 8080)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (*default* *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**success** – True if connection was successful

**Return type**

*bool*

**property token:** *str*

The token associated with the client

**unshare\_model\_with\_group**(*model\_id: str, group\_id: str, port: int = 8080, use\_port: bool | None = None*) → *bool*

Unshare a model with a group

```
>>> import aisquared
>>> client = aisquared.client.AISquaredPlatformClient()
>>> client.unshare_model_with_group('model_id', 'group_id')
True
```

**Parameters**

- **model\_id** (*str*) – The ID of the model
- **group\_id** (*str*) – The ID of the group
- **port** (*int* (default 8080)) – The API port to use. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (default *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**success** – Returns True if successful

**Return type**

bool

**unshare\_model\_with\_user**(*model\_id: str, user\_id: str, port: int = 8080, use\_port: bool | None = None*) → bool

Unshare a model with a user

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.unshare_model_with_user('model_id', 'user_id')
True
```

**Parameters**

- **model\_id** (*str*) – The ID for the model
- **user\_id** (*str*) – The ID for the user
- **port** (*int* (default 8080)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (default *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**success** – Whether the action was successful

**Return type**

bool

**update\_group**(*group\_id: str, display\_name: str, role\_id: str, port: int = 8086, use\_port: bool | None = None*) → bool

Update information about a group

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.update_group(
    'group_id',
    'group display name',
    'role_id'
)
True
```

**Parameters**

- **group\_id** (*str*) – The ID of the group to update
- **display\_name** (*str*) – The display name of the group
- **role\_id** (*str*) – The ID of the role for the group
- **port** (*int* (*default* 8086)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (*default* *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value

**Returns**

**success** – Returns True if successful

**Return type**

bool

**update\_user**(*user\_id: str, user\_name: str, given\_name: str, family\_name: str, email: str, role\_id: str, active: bool = True, middle\_name: str | None = None, company\_id: str | None = None, password: str | None = None, port: int = 8085, use\_port: bool | None = None*) → bool

Update information about a user

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.update_user(
    'user_id',
    'user name',
    'given_name',
    'family_name',
    'user_email',
    'role_id'
)
True
```

**Parameters**

- **user\_id** (*str*) – The ID of the user to update
- **user\_name** (*str*) – The display name of the user
- **given\_name** (*str*) – The first name of the user
- **family\_name** (*str*) – The last name of the user
- **email** (*str*) – The user's email
- **role\_id** (*str*) – The ID of the user's role
- **active** (*bool* (*default* *True*)) – Whether the user is active
- **middle\_name** (*str* or *None* (*default* *None*)) – The user's middle name
- **company\_id** (*str* or *None* (*default* *None*)) – The user's company ID
- **password** (*str* or *None* (*default* *None*)) – The user's password
- **port** (*int* (*default* 8085)) – The API port for the call. This can be handled automatically by the platform ALB
- **use\_port** (*bool* or *None* (*default* *None*)) – Whether to use port in URL formatting. If *None*, defaults to class value



**Returns****success** – Returns True if update is successful**Return type**

bool

**upload\_model**(*model\_file*: str, *port*: int = 8081, *use\_port*: bool | None = None) → str

Upload a model to the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.upload_model('my_model_filename.air')
True
```

**Parameters**

- **model\_file** (*path* or *path-like*) – The path to the model file
- **port** (int (default 8081)) – The API port to use. This can be handled automatically by the platform ALB
- **use\_port** (bool or None (default None)) – Whether to use port in URL formatting. If None, defaults to class value

**Returns****successful** – Whether the action was successful**Return type**

bool

**property use\_port****property username:** str

The username associated with the client

**aisquared.platform.DatabricksAPIException module****exception** aisquared.platform.DatabricksAPIException.DatabricksAPIException

Bases: Exception

**aisquared.platform.DatabricksClient module****class** aisquared.platform.DatabricksClient.DatabricksClient

Bases: object

Client for working with a connected Databricks environment

When using the client for the first time, it is important to authenticate the client using the *client.login()* method. When doing so, the client will ask for any required information interactively.

```
>>> import aisquared
>>> client = aisquared.platform.DatabricksClient()
>>> # If you have never logged in before, run the following code:
>>> client.login()
>>> # Interactive session requesting required information
```

**property** `base_url: str`

The base URL for the workspace

**create\_compute**(*compute\_name: str, spark\_version: str, node\_type\_id: str*) → dict

Create a compute resource

**Parameters**

- **compute\_name** (*str*) – The name of the compute to create
- **spark\_version** (*str*) – The spark version to use for the compute resource
- **node\_type\_id** (*str*) – The node type ID to use

**Returns**

**compute\_info** – The information about the created compute resource

**Return type**

dict

**create\_job**(*job\_name: str, tasks: list, libraries: list, compute\_name: str, spark\_version: str, node\_type\_id: str, cron\_syntax: str | None = None, timezone: str | None = None*) → bool

Create a job using notebooks and/or scripts in the workspace

**Parameters**

- **job\_name** (*str*) – The name for the job
- **tasks** (*list of dict*) – List of {task\_name : task\_script} dictionary pairs to run in the job
- **libraries** (*list of str*) – The dependent libraries to install on all compute
- **compute\_name** (*str*) – The name of the compute to provision specifically for this job
- **spark\_version** (*str*) – The version of Spark to use on the compute instances
- **node\_type\_id** (*str*) – The node type to use
- **cron\_syntax** (*str or None (default None)*) – If the job is to be set to a schedule, the cron syntax for that schedule
- **timezone** (*str or None (default None)*) – The timezone to set the schedule to, if cron syntax is provided

**Returns**

**success** – Whether the create job call was successful

**Return type**

bool

**create\_served\_model**(*model\_name: str, model\_version: str, workload\_size: str, scale\_to\_zero\_enabled: bool = True, workload\_type: str = 'CPU'*) → dict

Create a model serving endpoint

**Parameters**

- **model\_name** (*str*) – The name of the model to serve
- **model\_version** (*str*) – The version of the model to serve
- **workload\_size** (*str*) – The workload size of the serving endpoint
- **scale\_to\_zero\_enabled** (*bool (default True)*) – Whether to allow for scaling the endpoint to zero

- **type** (*workload*) – The workload type - either ‘CPU’ or ‘GPU’

**Returns**

**configuration** – Configuration information about the serving endpoint

**Return type**

dict

**delete\_compute**(*compute\_id: str*) → bool

Delete a compute resource in the workspace

**Parameters**

**compute\_id** (*str*) – The ID for the compute to delete

**Returns**

**success** – Whether the operation was successful

**Return type**

bool

**delete\_from\_workspace**(*filename: str*) → bool

Delete a file from the workspace

**Parameters**

**filename** (*str*) – The name of the file to delete

**Returns**

**success** – Whether the operation is successful

**Return type**

bool

**delete\_job**(*job\_id: str*) → bool

Delete a job from the workspace

**Parameters**

**job\_id** (*str*) – The ID of the job to delete

**Returns**

**success** – Whether the delete operation was successful

**Return type**

bool

**delete\_registered\_model**(*model\_name: str*) → bool

Delete a registered model

**Parameters**

**model\_name** (*str*) – The name of the model to delete

**Returns**

**success** – Whether the delete operation was successful

**Return type**

bool

**delete\_served\_model**(*model\_name: str*) → bool

Delete a served model in the workspace

**Parameters**

**model\_name** (*str*) – The name of the model to delete

**Returns**

**success** – Whether the delete operation was successful

**Return type**

bool

**download\_from\_workspace**(*filename: str*) → str

Download a file from the workspace

**Parameters**

**filename** (*str*) – The filename of the file to download

**Returns**

**contents** – The contents of the file

**Return type**

str

**property headers:** dict

API headers for calls to the API

**list\_compute**(*as\_df: bool = True*) → dict | DataFrame

List compute in the workspace

**Parameters**

**as\_df** (*bool (default True)*) – Whether to return a pandas DataFrame

**Returns**

**compute** – The compute resources in the workspace

**Return type**

dict or pd.DataFrame

**list\_jobs**(*as\_df: bool = True*) → dict | DataFrame

List all jobs in the workspace

**Parameters**

**as\_df** (*bool (default True)*) – Whether to return a pandas DataFrame

**Returns**

**jobs** – The jobs that exist in the workspace

**Return type**

dict or pandas DataFrame

**list\_registered\_models**(*as\_df: bool = True*) → dict | DataFrame

List registered models in the workspace

**Parameters**

**as\_df** (*bool (default True)*) – Whether to return a pandas DataFrame

**Returns**

**models** – The models in the workspace

**Return type**

dict or pandas DataFrame

**list\_served\_models**(*as\_df: bool = True*) → dict | DataFrame

List served models in the workspace

**Parameters**

**as\_df** (*bool (default True)*) – Whether to return results as a pandas DataFrame

**Returns****models** – The models served in the workspace**Return type**

dict or pandas DataFrame

**list\_workspace**(*as\_df: bool = True*) → DataFrame | dict

List files in the connected Databricks workspace

**Parameters****as\_df** (*bool (default True)*) – Whether to return the results as a pandas DataFrame**Returns****results** – The files in the workspace**Return type**

dict or pd.DataFrame

**login**(*url: str | None = None, username: str | None = None, token: str | None = None, persist: bool = True*) → None

Log in to the Databricks environment programmatically

```

>>> import aisquared
>>> client = aisquared.platform.DatabricksClient()
>>> client.login()
Enter URL: {Databricks_workspace_url}
Enter Username: your.email@your_domain.com
Enter Secret Token: <hidden>

```

**Parameters**

- **url** (*str or None (default None)*) – The URL of the Databricks workspace
- **username** (*str or None (default None)*) – The username in the Databricks workspace
- **token** (*str or None (default None)*) – The secret token for the Databricks workspace
- **persist** (*bool (default True)*) – Whether to persist the login information, eliminating the need to run this command again in the future

**run\_job**(*job\_id: str*) → int

Run a job

**Parameters****job\_id** (*str*) – The ID of the job to run**Returns****run\_id** – The ID of the specific run that was created**Return type**

int

**start\_compute**(*compute\_id: str*) → bool

Start a compute resource

**Parameters****compute\_id** (*str*) – The ID of the compute to start

**Returns**

**success** – Whether the start operation was successful

**Return type**

bool

**stop\_compute**(*compute\_id: str*) → bool

Stop a compute resource

**Parameters**

**compute\_id**(*str*) – The ID of the compute to start

**Returns**

**success** – Whether the stop operation was successful

**Return type**

bool

**property token: str**

The token to use for the workspace

**upload\_to\_workspace**(*filename: str, overwrite: bool = False*) → bool

Upload a file to the workspace

**Parameters**

- **filename**(*str*) – The name of the file to upload
- **overwrite**(*bool (default False)*) – Whether to overwrite the file if one of the same name already exists in the workspace

**Returns**

**success** – Whether the upload was successful

**Return type**

bool

**property username: str**

The user's username

**aisquared.platform.NoResultsFoundError module**

**exception** aisquared.platform.NoResultsFoundError.NoResultsFoundError

Bases: Exception

**aisquared.platform.additional\_utils module****aisquared.platform.crudl module****aisquared.platform.feedback module****aisquared.platform.metrics module****aisquared.platform.sharing module**

## aisquared.platform.user\_group module

### Module contents

Utilities for interacting with the AI Squared Platform.

The primary class within this subpackage is the *AISquaredPlatformClient* class, which has the capabilities to interact with much of the functionality in the AI Squared platform. For more information about this class, please see its documentation.

## aisquared.serving package

### Submodules

#### aisquared.serving.deploy\_model module

```
aisquared.serving.deploy_model.deploy_model(saved_model: str, model_type: str, host: str = '127.0.0.1',
                                             port: int = 2244, custom_objects: dict | None = None,
                                             additional_functions_file: str | None = None)
```

Deploy a model to a Flask server on the specified host

##### Parameters

- **saved\_model** (*Path-like*) – The path to the saved model directory or model file
- **model\_type** (*str*) – The type of model
- **host** (*str* (default '127.0.0.1')) – The host to deploy to
- **port** (*int* (default 2244)) – The port to deploy to
- **custom\_objects** (*dict or None* (default None)) – Any custom objects to load when using a BeyondML model
- **additional\_functions\_file** (*file-like or None* (default None)) – File name containing additional functions (which have to be named *preprocess* and *postprocess*, if created) that are used during the prediction process

```
aisquared.serving.deploy_model.load_beyondml_model(model: str, custom_objects: dict)
```

Load a BeyondML model with custom objects

#### aisquared.serving.get\_remote\_prediction module

```
aisquared.serving.get_remote_prediction.get_remote_prediction(data: dict | str | ndarray | list, host:
                                                             str = '127.0.0.1', port: int = 2244)
                                                             → list
```

Send data to use for prediction

##### Parameters

- **data** (*dict, str, np.ndarray, or list*) – The data to be predicted on
- **host** (*str* (default '127.0.0.1')) – The host to use
- **port** (*int* (default '2244')) – The port to use

## Notes

- If data is a dictionary, it is expected to already be correctly formatted
- If data is a string, it is expected to already be correctly formatted

## Returns

**predictions** – The predictions from the deployed model

## Return type

list

## Module contents

The `aisquared.serving` package contains utilities to serve models to a local REST endpoint.

Here is an example of how to serve a simple keras model using these utilities:

```
>>> # Assume model is already trained and stored in memory as model
>>> from aisquared import serving
>>> serving.save_keras_model(model, 'my_model')
>>> serving.deploy_model(
    'my_model',
    'keras',
    additional_functions_file = '<optional file containing `preprocess` and
    ↳ `postprocess` functions, if applicable>'
)
App created successfullly. Serving and awaiting requests
```

And to retrieve predictions from the model:

```
>>> # From a separate terminal, assume data is already loaded
>>> from aisquared import serving
>>> serving.get_remote_predictions(data) # Do not need to change host or port if
    ↳ predicting from the same machine
*predictions*
```

## aisquared.utils package

### Submodules

#### aisquared.utils.utils module

`aisquared.utils.utils.get_model(model_type: str, input_shape: int | tuple, num_outputs: int, output_activation: str, size: str = 'small', vocab_size: None | int = None)`

Get a pre-configured model for different use cases

## Parameters

- **model\_type** (*str*) – Either 'cv', 'nlp\_embedding', or 'fc', defining the model type
- **input\_shape** (*int or tuple of int*) – The input shape to the model
- **num\_outputs** (*int*) – The output shape of the model



- **output\_activation** (*str or keras activation function*) – The activation of the final layer of the model
- **size** (*str (default 'small')*) – One of either 'small', 'medium', or 'large'
- **vocab\_size** (*str or None (default None)*) – Size of the vocab, if model\_type is 'nlp\_embedding'

**Returns**

**model** – The model

**Return type**

TensorFlow Keras model

`aisquared.utils.mimic_model(trained_model: BaseEstimator, nnet: Model, training_data: ndarray, test_data: ndarray, test_labels: ndarray, problem_type: str, loss: str, metrics: str | list, optimizer: str, mimic_proba: bool = False, retention: float = 0.9, batch_size: int = 32, epochs: int = 100, starting_sparsification: int = 0, max_sparsification: int = 99, sparsification_rate: int = 5) → Model`

Train a sparse neural network to mimic a scikit-learn model

**Parameters**

- **trained\_model** (*sklearn model*) – The model that is already trained
- **nnet** (*TensorFlow keras Model*) – The neural network to train to mimic the trained model
- **training\_data** (*array or array-like*) – The input data that was used to train the trained model
- **test\_data** (*array or array-like*) – The input data to be used for testing
- **test\_labels** (*array or array-like*) – The output data used in testing
- **problem\_type** (*str*) – The type of problem, either 'classification' or 'regression'
- **loss** (*str or keras loss function*) – The loss to use
- **metrics** (*str, function or list of str, function*) – Metrics to measure
- **optimizer** (*str or keras optimizer*) – The optimizer to use
- **mimic\_proba** (*bool (default False)*) – For classification, mimic the probability outputs
- **retention** (*float (default 0.9)*) – The retention of performance to allow further pruning
- **batch\_size** (*int (default 32)*) – The batch size to use while training
- **epochs** (*int (default 100)*) – The number of epochs (if early stopping is not met beforehand)
- **starting\_sparsification** (*int (default 0)*) – The starting model sparsification
- **max\_sparsification** (*int (default 99)*) – The maximum sparsification to allow
- **sparsification\_rate** (*int (default 5)*) – The sparsification rate when invoked

**Returns**

**nnet** – The trained model

**Return type**

TensorFlow keras Model

## **Module contents**

Additional utilities to use with the *aisquared* package. These utilities currently consist of two functions, the *mimic\_model* and *get\_model* functions. They utilize functionality that exists in our open source package BeyondML to train teacher-student models

To see in-depth examples of how to use these functions, please visit our GitHub repository at <https://github.com/AISquaredInc/MimicModelExamples>

### **1.1.1.2 Module contents**

This package contains utilities to interact with the AI Squared technology stack, particularly with developing and deploying models to the AI Squared Browser Extension or other applications developed through the AI Squared JavaScript SDK.

## CHANGELOG

- **Version 0.1.3**

- Added *flags* parameter to *TextHarvester* using regular expression harvesting
- Deleted *model\_feedback* parameter in *ModelConfiguration* object and included functionality in *feedback\_steps* parameter
- Changed *format* parameter to *header* for both deployed analytics
- Added feedback and stages to *DocumentPredictor* and *ImagePredictor* objects
- Non-API changes for *ALLOWED\_STAGES*
- Fixed bugs preventing Windows users from importing the package
- Updated *ModelConfiguration* to include *url* parameter
- Changed default tokenization string

- **Version 0.2.0**

- Moved preprocessing steps under subpackages for specific kinds of preprocessing steps
- Cleaned up documentation to render within programmatic access environments
- Added *aisquared.logging* subpackage
- **Created *InputHarvester***
  - \* Allows for harvesting of input text, images, and tabular data
- Created the *aisquared.serving* subpackage, specifically the *deploy\_model* and *get\_remote\_prediction* functions
- Created the *GraphConfiguration* class
- Added *auto-run* parameter to *ModelConfiguration* and *GraphConfiguration* classes
- **Created the *aisquared* CLI with the following commands:**
  - \* *aisquared deploy*, which deploys a model locally
  - \* *aisquared predict*, which predicts using a local JSON file
  - \* *aisquared airfiles*, which contains the subcommands *list*, *delete*, *download*, and *upload*
- Changed all classes within *aisquared.config.analytic* to accept 'tabular' as an *input\_type*
- Removed *aisquared.logging* and *aisquared.remote* from top-level imports
- Added *round* parameter to Regression postprocessor
- Removed *DocumentPredictor* and *ImagePredictor* classes

- Removed *ChainRendering* class
- Created *FilterRendering* class
- Altered *QUALIFIERS*
- Added advanced rendering parameters to rendering objects
- Removed *logging* and *remote* subpackages from top-level *aisquared* import
- **Version 0.2.1**
  - Added the *S3Connector* class to the *analytics* subpackage, which allows download of an analytic directly from S3
  - Updated the documentation and added the *docs* subdirectory for hosting the documentation on GitHub Pages
- **Version 0.2.2**
  - Fixed bug in *to\_dict* method within *ObjectRendering* class
  - Fixed bug in name of *MultiplyValue* step
  - Fixed bug in datatype checking for text harvester
  - Added *body\_only* parameter to *TextHarvester*
  - Added ‘underline’ to possible badges
  - Added *threshold\_key* and *threshold\_values* to relevant rendering classes
  - Added *Trim* text preprocessing class
  - Added *CustomObject* in the base package to allow for creation of custom classes
  - Added keyword harvesting capabilities
  - Added *utils* subpackage with capabilities to mimic a trained sklearn model
  - Small documentation changes
  - Changed the required imports for the package to streamline installation process, and created two installation options *aisquared* and *aisquared[full]*
- **Version 0.2.3**
  - Added functionality to add custom preprocessing and postprocessing functions to the model deployment pipeline
  - Added *all* parameter to *LocalAnalytic* class
  - Changed under-the-hood functionality of *mimic\_model* function in line with updates to *BeyondML*
  - Altered the *ReverseMLWorkflow* analytic
  - Added the *BarChartRendering*, *ContainerRendering*, *DashboardReplacementRendering*, *DoughnutChartRendering*, *HTMLTagRendering*, *LineChartRendering*, *PieChartRendering*, *SOSRendering*, and *TableRendering* rendering classes
  - Added the *QueryParameterHarvester* harvester class
  - Added the *limit* parameter to the *TextHarvester* class
- **Version 0.3.0**
  - Added type hinting to documentation strings
  - Revamped documentation to use Sphinx

- **Version 0.3.1**
  - Changed Python type hints to allow for backwards compatibility with older versions of Python
- **Version 0.3.2**
  - Added functionality to the *AISquaredPlatformClient*
  - Added *top\_level\_kwargs* parameter to the *CustomObject* class
  - Added *DashboardRendering* class
  - Removed ‘px’ from default values in *ImageRendering* and *ObjectRendering* classes
  - Added functionality for creating, updating, and deleting users to *AISquaredPlatformClient*
  - Added functionality for creating, updating, and deleting groups to *AISquaredPlatformClient*
  - Fixed bug related to requiring *auto\_run* parameter to be string (fix involves casting as string)
  - Altered schemas for different “Chart” Rendering classes to conform to JavaScript standards
  - Streamlined the *ModelConfiguration* class to allow a more functional interface to build *.air* files
  - Updated *ContainerRendering* class with parameters for *position* and *static\_position*
  - Updated across-the-board functionality of the *AISquaredPlatformClient*
- **Version 0.3.3**
  - Updated functionality of the *AISquaredPlatformClient* to interact directly with the platform ALB
  - Changed function names in support of change from MANN to BeyondML
  - Added documentation surrounding global configuration objects
  - Removed redundant additional dependencies
- **Version 0.3.4**
  - Added support for custom CSS strings to appropriate rendering classes
  - Refactored *AISquaredPlatformClient* to import functions from support files
  - Fixed documentation errors for the documentation site
  - Checked whether responses returned OK status code rather than 200
  - Moved *CustomObject* to *aisquared.config* from *aisquared.base*
  - Changed endpoint used to list platform users
  - Fixed response behaviors where no data was returned from *AISquaredPlatformClient*
- **Version 0.3.5**
  - Changed *file\_name* parameter in *ReverseMLWorkflow* to *file\_names*
  - Added *documentation\_link* parameter to *ModelConfiguration* class
- **Version 0.3.6**
  - Fixed issue with type checking for *ModelConfiguration* Rendering classes
  - Restricted TensorFlow version to below 2.12.0 to prevent import issues
  - Added *position* parameter to *WordRendering* class
  - Changed default CSS styling for rendering classes
  - Changed name of all *processor* classes to *processer*

- **Version 0.3.7**
  - Changed schema of the *DeployedAnalytic* class to include API key management
  - Changed JSON schema of Preprocessor classes
  - Allowed .keras files to be saved and loaded with the *ModelConfiguration* and *GraphConfiguration* APIs into .air files
  - Relaxed TensorFlow requirements enforced in version 0.3.6
- **Version 0.3.8**
  - Created *ChatbotHarvester* class
  - Created *TextRendering* class
  - Changed location of reference lists of classes to clean up code
  - Updated class schemas to ensure compliance with expectations
  - Updated test cases
- **Version 0.3.9**
  - Created *CustomRendering* class
  - Changed to full import of *CustomObject* in *aisquared.base* subpackage
- **Version 0.3.10**
  - Added *DatabricksClient* to the *aisquared.platform* subpackage

## PYTHON MODULE INDEX

### a

aisquared, 70

aisquared.base, 4

aisquared.base.BaseObject, 3

aisquared.base.css, 4

aisquared.base.endpoints, 4

aisquared.base.harvesting, 4

aisquared.base.platform, 4

aisquared.base.preprocessing, 4

aisquared.base.rendering, 4

aisquared.base.stages, 4

aisquared.config, 44

aisquared.config.analytic, 8

aisquared.config.analytic.DeployedAnalytic, 4

aisquared.config.analytic.DeployedModel, 5

aisquared.config.analytic.LocalAnalytic, 6

aisquared.config.analytic.LocalModel, 6

aisquared.config.analytic.ReverseMLWorkflow,  
7

aisquared.config.CustomObject, 41

aisquared.config.feedback, 11

aisquared.config.feedback.BinaryFeedback, 8

aisquared.config.feedback.ModelFeedback, 8

aisquared.config.feedback.MulticlassFeedback,  
9

aisquared.config.feedback.QualitativeFeedback,  
10

aisquared.config.feedback.RegressionFeedback,  
10

aisquared.config.feedback.SimpleFeedback, 11

aisquared.config.GraphConfiguration, 41

aisquared.config.harvesting, 14

aisquared.config.harvesting.ChatbotHarvester,  
11

aisquared.config.harvesting.ImageHarvester,  
12

aisquared.config.harvesting.InputHarvester,  
12

aisquared.config.harvesting.QueryParameterHarvester,  
12

aisquared.config.harvesting.TextHarvester, 13

aisquared.config.ModelConfiguration, 42

aisquared.config.postprocessing, 16

aisquared.config.postprocessing.BinaryClassification,  
14

aisquared.config.postprocessing.MulticlassClassification,  
15

aisquared.config.postprocessing.ObjectDetection,  
15

aisquared.config.postprocessing.Regression,  
16

aisquared.config.preprocessing, 25

aisquared.config.preprocessing.image, 19

aisquared.config.preprocessing.image.ImagePreprocessing,  
16

aisquared.config.preprocessing.image.Steps,  
17

aisquared.config.preprocessing.tabular, 22

aisquared.config.preprocessing.tabular.Steps,  
19

aisquared.config.preprocessing.tabular.TabularPreprocessing,  
21

aisquared.config.preprocessing.text, 25

aisquared.config.preprocessing.text.Steps, 22

aisquared.config.preprocessing.text.TextPreprocessing,  
24

aisquared.config.rendering, 40

aisquared.config.rendering.BarChartRendering,  
25

aisquared.config.rendering.ContainerRendering,  
26

aisquared.config.rendering.CustomRendering,  
27

aisquared.config.rendering.DashboardRendering,  
28

aisquared.config.rendering.DashboardReplacementRendering,  
29

aisquared.config.rendering.DocumentRendering,  
29

aisquared.config.rendering.DoughnutChartRendering,  
30

aisquared.config.rendering.FilterRendering,  
32

aisquared.config.rendering.HTMLTagRendering,

33  
aisquared.config.rendering.ImageRendering, 34  
aisquared.config.rendering.LineChartRendering,  
35  
aisquared.config.rendering.ObjectRendering,  
36  
aisquared.config.rendering.PieChartRendering,  
37  
aisquared.config.rendering.SOSRendering, 38  
aisquared.config.rendering.TableRendering, 38  
aisquared.config.rendering.TextRendering, 39  
aisquared.config.rendering.utils, 40  
aisquared.config.rendering.WordRendering, 39  
aisquared.logging, 44  
aisquared.platform, 67  
aisquared.platform.additional\_utils, 66  
aisquared.platform.AISquaredAPIException, 44  
aisquared.platform.AISquaredPlatformClient,  
45  
aisquared.platform.crudl, 66  
aisquared.platform.DatabricksAPIException, 61  
aisquared.platform.DatabricksClient, 61  
aisquared.platform.feedback, 66  
aisquared.platform.metrics, 66  
aisquared.platform.NoResultsFoundError, 66  
aisquared.platform.sharing, 66  
aisquared.platform.user\_group, 67  
aisquared.serving, 68  
aisquared.serving.deploy\_model, 67  
aisquared.serving.get\_remote\_prediction, 67  
aisquared.utils, 70  
aisquared.utils.utils, 68



## A

`add_bar_chart()` (*aisquared.config.rendering.DashboardRendering.DashboardRendering*  
*method*), 28  
`add_container()` (*aisquared.config.rendering.DashboardRendering.DashboardRendering*  
*method*), 28  
`add_doughnut_chart()`  
*(aisquared.config.rendering.DashboardRendering.DashboardRendering*  
*method)*, 28  
`add_html_tag()` (*aisquared.config.rendering.DashboardRendering.DashboardRendering*  
*method*), 28  
`add_line_chart()` (*aisquared.config.rendering.DashboardRendering.DashboardRendering*  
*method*), 28  
`add_node()` (*aisquared.config.GraphConfiguration.GraphConfiguration*  
*method*), 41  
`add_pie_chart()` (*aisquared.config.rendering.DashboardRendering.DashboardRendering*  
*method*), 28  
`add_question()` (*aisquared.config.feedback.ModelFeedback.ModelFeedback*  
*method*), 9  
`add_question()` (*aisquared.config.feedback.QualitativeFeedback.QualitativeFeedback*  
*method*), 10  
`add_step()` (*aisquared.config.preprocessing.image.ImagePreprocessing.ImagePreprocessor*  
*method*), 17  
`add_step()` (*aisquared.config.preprocessing.tabular.TabularPreprocessing.TabularPreprocessor*  
*method*), 21  
`add_step()` (*aisquared.config.preprocessing.text.TextPreprocessing.TextPreprocessor*  
*method*), 24  
`add_table()` (*aisquared.config.rendering.DashboardRendering.DashboardRendering*  
*method*), 28  
`add_users_to_group()`  
*(aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient*  
*method)*, 45  
`AddValue` (*class in aisquared.config.preprocessing.image.Steps*),  
17  
`aisquared`  
*module*, 70  
`aisquared.base`  
*module*, 4  
`aisquared.base.BaseObject`  
*module*, 3  
`aisquared.base.css`  
*module*, 4  
`aisquared.base.endpoints`  
*module*, 4  
`aisquared.base.harvesting`  
*module*, 4  
`aisquared.base.platform`  
*module*, 4  
`aisquared.base.preprocessing`  
*module*, 4  
`aisquared.base.rendering`  
*module*, 4  
`aisquared.base.stages`  
*module*, 4  
`aisquared.config`  
*module*, 44  
`aisquared.config.analytic`  
*module*, 8  
`aisquared.config.analytic.DeployedAnalytic`  
*module*, 4  
`aisquared.config.analytic.DeployedModel`  
*module*, 5  
`aisquared.config.analytic.LocalAnalytic`  
*module*, 6  
`aisquared.config.analytic.LocalModel`  
*module*, 6  
`aisquared.config.analytic.ReverseMLWorkflow`  
*module*, 7  
`aisquared.config.CustomObject`  
*module*, 41  
`aisquared.config.feedback`  
*module*, 11  
`aisquared.config.feedback.BinaryFeedback`  
*module*, 8  
`aisquared.config.feedback.ModelFeedback`  
*module*, 8  
`aisquared.config.feedback.MulticlassFeedback`  
*module*, 9  
`aisquared.config.feedback.QualitativeFeedback`  
*module*, 10  
`aisquared.config.feedback.RegressionFeedback`  
*module*, 10  
`aisquared.config.feedback.SimpleFeedback`  
*module*, 11  
`aisquared.config.GraphConfiguration`

|   |  |
|---|--|
| module, 41  | module, 28   |
| aisquared.config.harvesting                                 | aisquared.config.rendering.DashboardReplacementRendering |
| module, 14  | module, 29   |
| aisquared.config.harvesting.ChatbotHarvester                | aisquared.config.rendering.DocumentRendering             |
| module, 11  | module, 29   |
| aisquared.config.harvesting.ImageHarvester                  | aisquared.config.rendering.DoughnutChartRendering        |
| module, 12  | module, 30   |
| aisquared.config.harvesting.InputHarvester                  | aisquared.config.rendering.FilterRendering               |
| module, 12  | module, 32   |
| aisquared.config.harvesting.QueryParameterHarvester         | aisquared.config.rendering.HTMLTagRendering              |
| module, 12  | module, 33   |
| aisquared.config.harvesting.TextHarvester                   | aisquared.config.rendering.ImageRendering                |
| module, 13  | module, 34   |
| aisquared.config.ModelConfiguration                         | aisquared.config.rendering.LineChartRendering            |
| module, 42  | module, 35   |
| aisquared.config.postprocessing                             | aisquared.config.rendering.ObjectRendering               |
| module, 16  | module, 36   |
| aisquared.config.postprocessing.BinaryClassification        | aisquared.config.rendering.PieChartRendering             |
| module, 14  | module, 37   |
| aisquared.config.postprocessing.MulticlassClassification    | aisquared.config.rendering.SOSRendering                  |
| module, 15  | module, 38   |
| aisquared.config.postprocessing.ObjectDetection             | aisquared.config.rendering.TableRendering                |
| module, 15  | module, 38   |
| aisquared.config.postprocessing.Regression                  | aisquared.config.rendering.TextRendering                 |
| module, 16  | module, 39   |
| aisquared.config.preprocessing                              | aisquared.config.rendering.utils                         |
| module, 25  | module, 40   |
| aisquared.config.preprocessing.image                        | aisquared.config.rendering.WordRendering                 |
| module, 19  | module, 39   |
| aisquared.config.preprocessing.image.ImagePreprocessing     | aisquared.config.logging                                 |
| module, 16  | module, 44   |
| aisquared.config.preprocessing.image.Steps                  | aisquared.platform                                       |
| module, 17  | module, 67   |
| aisquared.config.preprocessing.tabular                      | aisquared.platform.additional_utils                      |
| module, 22  | module, 66   |
| aisquared.config.preprocessing.tabular.Steps                | aisquared.platform.AISquaredAPIException                 |
| module, 19  | module, 44   |
| aisquared.config.preprocessing.tabular.TabularPreprocessing | aisquared.platform.AISquaredPlatformClient               |
| module, 21  | module, 45   |
| aisquared.config.preprocessing.text                         | aisquared.platform.crudl                                 |
| module, 25  | module, 66   |
| aisquared.config.preprocessing.text.Steps                   | aisquared.platform.DatabricksAPIException                |
| module, 22  | module, 61   |
| aisquared.config.preprocessing.text.TextPreprocessing       | aisquared.platform.DatabricksClient                      |
| module, 24  | module, 61   |
| aisquared.config.rendering                                  | aisquared.platform.feedback                              |
| module, 40  | module, 66   |
| aisquared.config.rendering.BarChartRendering                | aisquared.platform.metrics                               |
| module, 25  | module, 66   |
| aisquared.config.rendering.ContainerRendering               | aisquared.platform.NoResultsFoundError                   |
| module, 26  | module, 66   |
| aisquared.config.rendering.CustomRendering                  | aisquared.platform.sharing                               |
| module, 27  | module, 66   |
| aisquared.config.rendering.DashboardRendering               | aisquared.platform.user_group                            |

module, 67  
 aisquared.serving  
   module, 68  
 aisquared.serving.deploy\_model  
   module, 67  
 aisquared.serving.get\_remote\_prediction  
   module, 67  
 aisquared.utils  
   module, 70  
 aisquared.utils.utils  
   module, 68  
 AISquaredAPIException, 44  
 AISquaredPlatformClient (class in  
   aisquared.platform.AISquaredPlatformClient),  
   45  
 all (aisquared.config.analytic.LocalAnalytic.LocalAnalytic  
   property), 6  
 analytic (aisquared.config.ModelConfiguration.ModelConfiguration  
   property), 43  
 analytic\_dict (aisquared.config.ModelConfiguration.ModelConfiguration  
   property), 43  
 anchor\_selector (aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering  
   property), 29  
 api\_key\_header\_name  
   (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic  
   property), 5  
 api\_key\_prefix (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic  
   property), 5  
 attributes (aisquared.config.harvesting.QueryParameterHarvester.QueryParameterHarvester  
   property), 13  
 auto\_run (aisquared.config.GraphConfiguration.GraphConfiguration  
   property), 41  
 auto\_run (aisquared.config.ModelConfiguration.ModelConfiguration  
   property), 43  
**B**  
 badge\_color (aisquared.config.rendering.ImageRendering.ImageRendering  
   property), 34  
 badge\_color (aisquared.config.rendering.ObjectRendering.ObjectRendering  
   property), 36  
 badge\_color (aisquared.config.rendering.WordRendering.WordRendering  
   property), 40  
 badge\_shape (aisquared.config.rendering.WordRendering.WordRendering  
   property), 40  
 BarChartRendering (class in  
   aisquared.config.rendering.BarChartRendering),  
   25  
 base\_url (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
   property), 45  
 base\_url (aisquared.platform.DatabricksClient.DatabricksClient  
   property), 61  
 BaseObject (class in aisquared.base.BaseObject), 3  
 BinaryClassification (class in  
   aisquared.config.postprocessing.BinaryClassification),  
   14  
 BinaryFeedback (class in  
   aisquared.config.feedback.BinaryFeedback), 8  
 body\_only (aisquared.config.harvesting.TextHarvester.TextHarvester  
   property), 13  
 bucket (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow  
   property), 7  
**C**  
 can\_toggle (aisquared.config.rendering.SOSRendering.SOSRendering  
   property), 38  
 ChatbotHarvester (class in  
   aisquared.config.harvesting.ChatbotHarvester),  
   11  
 classes (aisquared.config.rendering.DocumentRendering.DocumentRendering  
   property), 30  
 classes (aisquared.config.rendering.ImageRendering.ImageRendering  
   property), 34  
 classes (aisquared.config.rendering.WordRendering.WordRendering  
   property), 40  
 color (aisquared.config.preprocessing.image.Steps.ConvertToColor  
   property), 19  
 color (aisquared.config.rendering.ImageRendering.ImageRendering  
   property), 34  
 color (aisquared.config.rendering.ObjectRendering.ObjectRendering  
   property), 36  
 column (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow  
   property), 7  
 column (aisquared.config.preprocessing.tabular.Steps.DropColumn  
   property), 19  
 column (aisquared.config.preprocessing.tabular.Steps.OneHot  
   property), 20  
 columns (aisquared.config.preprocessing.tabular.Steps.MinMax  
   property), 20  
 columns (aisquared.config.preprocessing.tabular.Steps.ZScore  
   property), 21  
 compare (aisquared.config.GraphConfiguration.GraphConfiguration  
   method), 42  
 compare (aisquared.config.ModelConfiguration.ModelConfiguration  
   method), 43  
 ContainerRendering (class in  
   aisquared.config.rendering.ContainerRendering),  
   26  
 content\_html (aisquared.config.rendering.CustomRendering.CustomRendering  
   property), 27  
 content\_key (aisquared.config.rendering.WordRendering.WordRendering  
   property), 40  
 content\_style (aisquared.config.rendering.CustomRendering.CustomRendering  
   property), 27  
 ConvertToCase (class in  
   aisquared.config.preprocessing.text.Steps),  
   22

ConvertToColor (class in `aisquared.config.preprocessing.image.Steps`), 17  
 ConvertToVocabulary (class in `aisquared.config.preprocessing.text.Steps`), 22  
 create\_compute() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 62  
 create\_group() (`aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient` method), 45  
 create\_job() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 62  
 create\_served\_model() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 62  
 create\_user() (`aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient` method), 46  
 CustomObject (class in `aisquared.config.CustomObject`), 41  
 CustomRendering (class in `aisquared.config.rendering.CustomRendering`), 27  
**D**  
 DashboardRendering (class in `aisquared.config.rendering.DashboardRendering`), 28  
 DashboardReplacementRendering (class in `aisquared.config.rendering.DashboardReplacementRendering`), 29  
 DatabricksAPIException, 61  
 DatabricksClient (class in `aisquared.platform.DatabricksClient`), 61  
 delete\_compute() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 63  
 delete\_from\_workspace() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 63  
 delete\_group() (`aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient` method), 47  
 delete\_job() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 63  
 delete\_model() (`aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient` method), 47  
 delete\_registered\_model() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 63  
 delete\_served\_model() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 63  
 delete\_user() (`aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient` method), 47  
 deploy\_model() (in module `aisquared.serving.deploy_model`), 67  
 DeployedAnalytic (class in `aisquared.config.analytic.DeployedAnalytic`), 4  
 DeployedModel (class in `aisquared.config.analytic.DeployedModel`), 5  
 description (`aisquared.config.GraphConfiguration.GraphConfiguration` property), 42  
 description (`aisquared.config.ModelConfiguration.ModelConfiguration` property), 43  
 display (`aisquared.config.rendering.ContainerRendering.ContainerRendering` property), 27  
 DivideValue (class in `aisquared.config.preprocessing.image.Steps`), 17  
 documentation\_link (`aisquared.config.GraphConfiguration.GraphConfiguration` property), 42  
 documentation\_link (`aisquared.config.ModelConfiguration.ModelConfiguration` property), 43  
 DocumentRendering (class in `aisquared.config.rendering.DocumentRendering`), 29  
 documents (`aisquared.config.rendering.DocumentRendering.DocumentRendering` property), 30  
 DoughnutChartRendering (class in `aisquared.config.rendering.DoughnutChartRendering`), 30  
 download\_from\_workspace() (`aisquared.platform.DatabricksClient.DatabricksClient` method), 64  
 DropColumn (class in `aisquared.config.preprocessing.tabular.Steps`), 19  
**F**  
 FeedbackDict (class in `aisquared.config.harvesting.InputHarvester.InputHarvester` property), 12  
 feedback\_dict (`aisquared.config.ModelConfiguration.ModelConfiguration` property), 43  
 feedback\_steps (`aisquared.config.ModelConfiguration.ModelConfiguration` property), 43  
 filenames (`aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow` property), 7  
 filter\_by\_columns (`aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow` property), 8  
 filter\_type (`aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow` property), 8  
 FilterRendering (class in `aisquared.config.rendering.FilterRendering`), 32  
 Flags (`aisquared.config.harvesting.TextHarvester.TextHarvester` property), 13  
 FontASize (class in `aisquared.config.rendering.ImageRendering.ImageRendering` property), 34  
 font\_color (`aisquared.config.rendering.ObjectRendering.ObjectRendering` property), 36

```
font_size(aisquared.config.rendering.ImageRendering.ImageRendering.config.harvesting.TextHarvester.TextHarvester
property), 34
```

```
font_size(aisquared.config.rendering.ObjectRendering.HTMLTagRendering (class in
property), 36 aisquared.config.rendering.HTMLTagRendering),
33
```

## G

```
get_filenames() (aisquared.config.GraphConfiguration.GraphConfiguration
method), 42
id (aisquared.config.rendering.ContainerRendering.ContainerRendering
```

```
get_group() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient
method), 48
id(aisquared.config.rendering.CustomRendering.CustomRendering
```

```
get_group_id_by_name() (property), 28
(aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient (class in
method), 48
aisquared.config.harvesting.ImageHarvester),
```

`get_model()` (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient*  
method), [49](#) `ImagePreprocessor` (class in [12](#)

`get_model()` (in module `aisquared.utils.utils`), 68

```
get_model_filenames()
    (aisquared.config.ModelConfiguration.ModelConfiguration
    method), 43
    ImageRendering (class in
    aisquared.config.rendering.ImageRendering),
```

```
get_model_id_by_name()
    (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient
    method), 49
    include_probability
    (aisquared.config.rendering.DocumentRendering.DocumentRender
```

```
get_remote_prediction()      (in module property), 30
    aisquared.serving.get_remote_prediction), include_probability
67 (aisquared.config.rendering.ImageRendering.ImageRendering
get_role_id_by_role_name()   property), 34
```

```
(aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient.  
method), 50
```

```
include(aisquared.rendering.ObjectRendering.ObjectRendering  
method), 50
```

```
get_user() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient
method), 50
input_type (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic
```

```
get_user_id_by_name()                                property), 5
(aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient,
method), 50                                         property), 6
```

```

GraphConfiguration (class in aisquared.config.analytic.LocalAnalytic.LocalAnalytic
aisquared.config.GraphConfiguration), 41

```

```
input_type(aisquared.config.analytic.LocalModel.LocalModel
property),7
```

## H

```
harvester_dict(aisquared.config.ModelConfiguration.ModelConfiguration
property), 43
input_type(aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWo
property), 8
```

```

    harvesting_steps(aisquared.config.ModelConfiguration.ModelConfiguration
    input_type(aisquared.config.harvesting.InputHarvester.InputHarvester

```

```
header(aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic,
      InputHarvester,
      aisquared.config.harvesting.InputHarvester),
```

```
header (aisquared.config.analytic.DeployedModel.DeployedModel
property) 6 12
```

property),<sup>6</sup>

|   |   |
|---|---|
| headers (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient | key (aisquared.config.rendering.FilterRendering.FilterRendering |
|---|---|

headers (aisquared.platform.DatabricksClient.DatabricksClient property), 32

property),<sup>64</sup>

```
height (aisquared.config.rendering.ContainerRendering.ContainerRendering
property), 27 label (aisquared.config.rendering.ContainerRendering.ContainerRendering
```

how (aisquared.config.harvesting.ImageHarvester.ImageHarvester property), 27

label (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 12

label (aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering property), 29



label (*aisquared.config.rendering.SOSRendering.SOSRendering* property), 38  
 label\_map (*aisquared.config.feedback.BinaryFeedback.BinaryFeedback* property), 8  
 label\_map (*aisquared.config.feedback.MulticlassFeedback.MulticlassFeedback* property), 9  
 label\_map (*aisquared.config.postprocessing.BinaryClassification.BinaryClassification* property), 14  
 label\_map (*aisquared.config.postprocessing.MulticlassClassification.MulticlassClassification* property), 15  
 label\_map (*aisquared.config.postprocessing.ObjectDetection.ObjectDetection* property), 15  
 length (*aisquared.config.preprocessing.text.Steps.PadSequences* property), 23  
 limit (*aisquared.config.harvesting.TextHarvester.TextHarvester* property), 14  
 LineChartRendering (class in *aisquared.config.rendering.LineChartRendering*), 35  
 list\_compute() (*aisquared.platform.DatabricksClient.DatabricksClient* method), 64  
 list\_group\_users() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 51  
 list\_groups() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 51  
 list\_jobs() (*aisquared.platform.DatabricksClient.DatabricksClient* method), 64  
 list\_model\_feedback() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 52  
 list\_model\_prediction\_feedback() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 52  
 list\_model\_usage\_metrics() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 53  
 list\_model\_users() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 53  
 list\_models() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 54  
 list\_prediction\_feedback() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 54  
 list\_registered\_models() (*aisquared.platform.DatabricksClient.DatabricksClient* method), 64  
 list\_roles() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 54  
 list\_served\_models() (*aisquared.platform.DatabricksClient.DatabricksClient* method), 64  
 list\_user\_usage\_metrics() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 55  
 list\_users() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 55  
 list\_embeddings() (*aisquared.platform.DatabricksClient.DatabricksClient* method), 65  
 load\_key\_from\_model() (in module *aisquared.serving.deploy\_model*), 67  
 LocalAnalytic (class in *aisquared.config.analytic.LocalAnalytic*),  
 LocalModel (class in *aisquared.config.analytic.LocalModel*),  
 login() (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* method), 56  
 login() (*aisquared.platform.DatabricksClient.DatabricksClient* method), 65  
 lowercase (*aisquared.config.preprocessing.text.Steps.ConvertToCase* property), 22  
 M  
 max (*aisquared.config.postprocessing.Regression.Regression* property), 16  
 max\_length (*aisquared.config.harvesting.ChatbotHarvester.ChatbotHarvester* property), 11  
 max\_length (*aisquared.config.harvesting.InputHarvester.InputHarvester* property), 12  
 max\_vocab (*aisquared.config.preprocessing.text.Steps.ConvertToVocabulary* property), 22  
 maxs (*aisquared.config.preprocessing.tabular.Steps.MinMax* property), 19  
 means (*aisquared.config.preprocessing.tabular.Steps.ZScore* property), 21  
 method (*aisquared.config.preprocessing.image.Steps.Resize* property), 18  
 mimic\_model() (in module *aisquared.utils.utils*), 69  
 min (*aisquared.config.postprocessing.Regression.Regression* property), 16  
 MinMax (class in *aisquared.config.preprocessing.tabular.Steps*), 19  
 mins (*aisquared.config.preprocessing.tabular.Steps.MinMax* property), 20  
 mlflow\_token (*aisquared.config.GraphConfiguration.GraphConfiguration* property), 42  
 mlflow\_token (*aisquared.config.ModelConfiguration.ModelConfiguration* property), 43  
 mlflow\_uri (*aisquared.config.GraphConfiguration.GraphConfiguration* property), 42  
 mlflow\_uri (*aisquared.config.ModelConfiguration.ModelConfiguration* property), 43  
 mlflow\_user (*aisquared.config.GraphConfiguration.GraphConfiguration* property), 42  
 mlflow\_user (*aisquared.config.ModelConfiguration.ModelConfiguration* property), 43  
 ModelConfiguration (class in *aisquared.config.ModelConfiguration*), 42

|   |   |    |  |
|---|---|----|--|
| ModelFeedback   | (class  | in | aisquared.config.postprocessing.MulticlassClassification,    |
|   | <i>aisquared.config.feedback.ModelFeedback</i> ), | 8  | 15   |
| module  |   |    | aisquared.config.postprocessing.ObjectDetection,             |
| aisquared,  | 70  |    | 15   |
| aisquared.base,                                       | 4   |    | aisquared.config.postprocessing.Regression,                  |
| aisquared.base.BaseObject,                            | 3   |    | 16   |
| aisquared.base.css,                                   | 4   |    | aisquared.config.preprocessing, 25                           |
| aisquared.base.endpoints,                             | 4   |    | aisquared.config.preprocessing.image, 19                     |
| aisquared.base.harvesting,                            | 4   |    | aisquared.config.preprocessing.image.ImagePreprocessing,     |
| aisquared.base.platform,                              | 4   |    | 16   |
| aisquared.base.preprocessing,                         | 4   |    | aisquared.config.preprocessing.image.Steps,                  |
| aisquared.base.rendering,                             | 4   |    | 17   |
| aisquared.base.stages,                                | 4   |    | aisquared.config.preprocessing.tabular,                      |
| aisquared.config,                                     | 44  |    | 22   |
| aisquared.config.analytic,                            | 8   |    | aisquared.config.preprocessing.tabular.Steps,                |
| aisquared.config.analytic.DeployedAnalytic,           | 4   |    | 19   |
| aisquared.config.analytic.DeployedModel,              | 5   |    | aisquared.config.preprocessing.tabular.TabularPreprocessing, |
| aisquared.config.analytic.LocalAnalytic,              | 6   |    | 21   |
| aisquared.config.analytic.LocalModel,                 | 6   |    | aisquared.config.preprocessing.text, 25                      |
| aisquared.config.analytic.ReverseMLWorkflow,          | 7   |    | aisquared.config.preprocessing.text.Steps,                   |
| aisquared.config.CustomObject,                        | 41  |    | 22   |
| aisquared.config.feedback,                            | 11  |    | aisquared.config.preprocessing.text.TextPreprocessing,       |
| aisquared.config.feedback.BinaryFeedback,             | 8   |    | 24   |
| aisquared.config.feedback.ModelFeedback,              | 8   |    | aisquared.config.rendering, 40                               |
| aisquared.config.feedback.MulticlassFeedback,         | 9   |    | aisquared.config.rendering.BarChartRendering,                |
| aisquared.config.feedback.QualitativeFeedback,        | 10  |    | 25   |
| aisquared.config.feedback.RegressionFeedback,         | 10  |    | aisquared.config.rendering.ContainerRendering,               |
| aisquared.config.feedback.SimpleFeedback,             | 11  |    | 26   |
| aisquared.config.GraphConfiguration,                  | 41  |    | aisquared.config.rendering.CustomRendering,                  |
| aisquared.config.harvesting,                          | 14  |    | 27   |
| aisquared.config.harvesting.ChatbotHarvester,         | 11  |    | aisquared.config.rendering.DashboardRendering,               |
| aisquared.config.harvesting.ImageHarvester,           | 12  |    | 28   |
| aisquared.config.harvesting.InputHarvester,           | 12  |    | aisquared.config.rendering.DashboardReplacementRendering,    |
| aisquared.config.harvesting.QueryParameterHarvester,  | 12  |    | 29   |
| aisquared.config.harvesting.TextHarvester,            | 13  |    | aisquared.config.rendering.DocumentRendering,                |
| aisquared.config.ModelConfiguration,                  | 42  |    | 29   |
| aisquared.config.postprocessing,                      | 16  |    | aisquared.config.rendering.DoughnutChartRendering,           |
| aisquared.config.postprocessing.BinaryClassification, | 14  |    | 30   |
|   |   |    | aisquared.config.rendering.FilterRendering,                  |
|   |   |    | 32   |
|   |   |    | aisquared.config.rendering.HTMLTagRendering,                 |
|   |   |    | 33   |
|   |   |    | aisquared.config.rendering.ImageRendering,                   |
|   |   |    | 34   |
|   |   |    | aisquared.config.rendering.LineChartRendering,               |
|   |   |    | 35   |
|   |   |    | aisquared.config.rendering.ObjectRendering,                  |
|   |   |    | 36   |
|   |   |    | aisquared.config.rendering.PieChartRendering,                |
|   |   |    | 37   |
|   |   |    | aisquared.config.rendering.SOSRendering,                     |
|   |   |    | 38   |
|   |   |    | aisquared.config.rendering.TableRendering,                   |
|   |   |    | 38   |

aisquared.config.rendering.TextRendering, 39  
aisquared.config.rendering.utils, 40  
aisquared.config.rendering.WordRendering, 39  
aisquared.logging, 44  
aisquared.platform, 67  
aisquared.platform.additional\_utils, 66  
aisquared.platform.AISquaredAPIException, 44  
aisquared.platform.AISquaredPlatformClient, 45  
aisquared.platform.crudl, 66  
aisquared.platform.DatabricksAPIException, 61  
aisquared.platform.DatabricksClient, 61  
aisquared.platform.feedback, 66  
aisquared.platform.metrics, 66  
aisquared.platform.NoResultsFoundError, 66  
aisquared.platform.sharing, 66  
aisquared.platform.user\_group, 67  
aisquared.serving, 68  
aisquared.serving.deploy\_model, 67  
aisquared.serving.get\_remote\_prediction, 67  
aisquared.utils, 70  
aisquared.utils.utils, 68  
MulticlassClassification (class in aisquared.config.postprocessing.MulticlassClassification), 15  
MulticlassFeedback (class in aisquared.config.feedback.MulticlassFeedback), 9  
MultiplyValue (class in aisquared.config.preprocessing.image.Steps), 18  
**N**  
name (aisquared.config.GraphConfiguration.GraphConfiguration property), 42  
name (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
NoResultsFoundError, 66  
**O**  
ObjectDetection (class in aisquared.config.postprocessing.ObjectDetection), 15  
ObjectRendering (class in aisquared.config.rendering.ObjectRendering), 36  
OneHot (class in aisquared.config.preprocessing.tabular.Steps), 20  
oov\_character (aisquared.config.preprocessing.text.Steps.ConvertToVocabulary property), 22  
orientation (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 27  
owner (aisquared.config.GraphConfiguration.GraphConfiguration property), 42  
owner (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
**P**  
pad\_character (aisquared.config.preprocessing.text.Steps.PadSequences property), 23  
pad\_location (aisquared.config.preprocessing.text.Steps.PadSequences property), 23  
PadSequences (class in aisquared.config.preprocessing.text.Steps), 23  
password (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient property), 56  
path (aisquared.config.analytic.LocalAnalytic.LocalAnalytic property), 6  
path (aisquared.config.analytic.LocalModel.LocalModel property), 7  
period (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow property), 8  
PieChartRendering (class in aisquared.config.rendering.PieChartRendering), 37  
placement (aisquared.config.rendering.ImageRendering.ImageRendering property), 34  
placement (aisquared.config.rendering.ObjectRendering.ObjectRendering property), 36  
position (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 27  
position (aisquared.config.rendering.WordRendering.WordRendering property), 40  
postprocessor\_dict (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
postprocessing\_steps (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
prediction\_key (aisquared.config.rendering.DocumentRendering.DocumentRendering property), 30  
prediction\_value\_key (aisquared.config.rendering.TextRendering.TextRendering property), 39  
preprocessor\_dict (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
preprocessing\_steps (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
preserve\_aspect\_ratio (aisquared.config.preprocessing.image.Steps.Resize property), 18



**probability\_key** (aisquared.config.rendering.DocumentRendering.DocumentRendering property), 30  
**Q**  
**qualifier** (aisquared.config.rendering.FilterRendering.FilterRendering property), 32  
**QualitativeFeedback** (class in aisquared.config.feedback.QualitativeFeedback), 10  
**query\_keys** (aisquared.config.harvesting.QueryParameterHarvester.QueryParameterHarvester property), 13  
**query\_selector** (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 27  
**query\_selector** (aisquared.config.rendering.CustomRendering.CustomRendering property), 28  
**QueryParameterHarvester** (class in aisquared.config.harvesting.QueryParameterHarvester), 12  
**R**  
**regex** (aisquared.config.harvesting.TextHarvester.TextHarvester property), 14  
**Regression** (class in aisquared.config.postprocessing.Regression.Regression), 16  
**RegressionFeedback** (class in aisquared.config.feedback.RegressionFeedback), 10  
**remove\_digits** (aisquared.config.preprocessing.text.Steps.RemoveDigits.RemoveDigits property), 23  
**remove\_punctuation** (aisquared.config.preprocessing.text.Steps.RemovePunctuation.RemovePunctuation property), 23  
**remove\_users\_from\_group()** (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient method), 56  
**RemoveCharacters** (class in aisquared.config.preprocessing.text.Steps.RemoveCharacters.RemoveCharacters), 23  
**render\_dict** (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
**rendering\_steps** (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
**Resize** (class in aisquared.config.preprocessing.image.Steps.Resize.Resize), 18  
**result\_key** (aisquared.config.rendering.WordRendering.WordRendering property), 40  
**ReverseMLWorkflow** (class in aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow), 7  
**round** (aisquared.config.postprocessing.Regression.Regression property), 16  
**run\_job()** (aisquared.platform.DatabricksClient.DatabricksClient method), 65  
**S**  
**save\_default\_css()** (in aisquared.config.rendering.utils), 40  
**secret** (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic property), 5  
**secret** (aisquared.config.analytic.DeployedModel.DeployedModel property), 6  
**secret** (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow property), 8  
**ShareModelWithGroup()** (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient method), 57  
**share\_model\_with\_user()** (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient method), 57  
**size** (aisquared.config.preprocessing.image.Steps.Resize.Resize property), 18  
**SOSRendering** (class in aisquared.config.rendering.SOSRendering.SOSRendering), 38  
**split\_sentences** (aisquared.config.preprocessing.text.Steps.Tokenize.Tokenize property), 24  
**split\_words** (aisquared.config.preprocessing.text.Steps.Tokenize.Tokenize property), 24  
**stage** (aisquared.config.GraphConfiguration.GraphConfiguration property), 44  
**stage** (aisquared.config.ModelConfiguration.ModelConfiguration property), 44  
**start\_compute()** (aisquared.platform.DatabricksClient.DatabricksClient method), 65  
**static\_position** (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 27  
**stds** (aisquared.config.preprocessing.tabular.Steps.ZScore.ZScore property), 21  
**step\_dict** (aisquared.config.preprocessing.image.ImagePreprocessing.ImagePreprocessing property), 17  
**step\_dict** (aisquared.config.preprocessing.text.TextPreprocessing.TextPreprocessing property), 24  
**steps** (aisquared.config.rendering.DashboardRendering.DashboardRendering property), 28  
**stop\_compute()** (aisquared.platform.DatabricksClient.DatabricksClient method), 66  
**subtract\_value** (class in aisquared.config.preprocessing.image.Steps.SubtractValue.SubtractValue), 18  
**T**  
**TableRendering** (class in

aisquared.config.rendering.TableRendering),  
38  
TabularPreprocessor (class in aisquared.config.preprocessing.tabular.TabularPreprocessor), 10  
21  
test\_connection() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient), 11  
method), 58  
TextHarvester (class in aisquared.config.harvesting.TextHarvester), 13  
aisquared.config.harvesting.TextHarvester), 13  
TextPreprocessor (class in aisquared.config.preprocessing.text.TextPreprocessing), 24  
aisquared.config.preprocessing.text.TextPreprocessing), 24  
TextRendering (class in aisquared.config.rendering.TextRendering), 39  
aisquared.config.rendering.TextRendering), 39  
thickness (aisquared.config.rendering.ImageRendering.ImageRendering), 34  
property), 34  
thickness (aisquared.config.rendering.ObjectRendering.ObjectRendering), 36  
property), 36  
threshold (aisquared.config.postprocessing.BinaryClassification.BinaryClassification), 14  
property), 14  
threshold (aisquared.config.postprocessing.ObjectDetection.ObjectDetection), 15  
property), 15  
threshold\_key (aisquared.config.rendering.DocumentRendering.DocumentRendering), 30  
property), 30  
threshold\_key (aisquared.config.rendering.ImageRendering.ImageRendering), 34  
property), 34  
threshold\_key (aisquared.config.rendering.WordRendering.WordRendering), 40  
property), 40  
threshold\_value (aisquared.config.rendering.DocumentRendering.DocumentRendering), 30  
property), 30  
threshold\_value (aisquared.config.rendering.ImageRendering.ImageRendering), 34  
property), 34  
threshold\_value (aisquared.config.rendering.WordRendering.WordRendering), 40  
property), 40  
to\_dict() (aisquared.base.BaseObject.BaseObject), 3  
method), 3  
to\_dict() (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic), 5  
method), 5  
to\_dict() (aisquared.config.analytic.DeployedModel.DeployedModel), 6  
method), 6  
to\_dict() (aisquared.config.analytic.LocalAnalytic.LocalAnalytic), 6  
method), 6  
to\_dict() (aisquared.config.analytic.LocalModel.LocalModel), 7  
method), 7  
to\_dict() (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow), 8  
method), 8  
to\_dict() (aisquared.config.CustomObject.CustomObject), 41  
method), 41  
to\_dict() (aisquared.config.feedback.BinaryFeedback.BinaryFeedback), 8  
method), 8  
to\_dict() (aisquared.config.feedback.ModelFeedback.ModelFeedback), 9  
method), 9  
to\_dict() (aisquared.config.feedback.MulticlassFeedback.MulticlassFeedback), 9  
method), 9  
to\_dict() (aisquared.config.feedback.QualitativeFeedback.QualitativeFeedback), 10  
method), 10  
to\_dict() (aisquared.config.feedback.RegressionFeedback.RegressionFeedback), 10  
method), 10  
to\_dict() (aisquared.config.feedback.SimpleFeedback.SimpleFeedback), 11  
method), 11  
to\_dict() (aisquared.config.GraphConfiguration.GraphConfiguration), 42  
method), 42  
to\_dict() (aisquared.config.harvesting.ChatbotHarvester.ChatbotHarvester), 11  
method), 11  
to\_dict() (aisquared.config.harvesting.ImageHarvester.ImageHarvester), 12  
method), 12  
to\_dict() (aisquared.config.harvesting.InputHarvester.InputHarvester), 12  
method), 12  
to\_dict() (aisquared.config.harvesting.QueryParameterHarvester.QueryParameterHarvester), 13  
method), 13  
to\_dict() (aisquared.config.harvesting.TextHarvester.TextHarvester), 14  
method), 14  
to\_dict() (aisquared.config.ModelConfiguration.ModelConfiguration), 44  
method), 44  
to\_dict() (aisquared.config.postprocessing.BinaryClassification.BinaryClassification), 14  
method), 14  
to\_dict() (aisquared.config.postprocessing.MulticlassClassification.MulticlassClassification), 15  
method), 15  
to\_dict() (aisquared.config.postprocessing.ObjectDetection.ObjectDetection), 15  
method), 15  
to\_dict() (aisquared.config.postprocessing.Regression.Regression), 16  
method), 16  
to\_dict() (aisquared.config.preprocessing.image.ImagePreprocessing.ImagePreprocessing), 17  
method), 17  
to\_dict() (aisquared.config.preprocessing.image.Steps.AddValue), 17  
method), 17  
to\_dict() (aisquared.config.preprocessing.image.Steps.ConvertToColor), 17  
method), 17  
to\_dict() (aisquared.config.preprocessing.image.Steps.DivideValue), 18  
method), 18  
to\_dict() (aisquared.config.preprocessing.image.Steps.MultiplyValue), 18  
method), 18  
to\_dict() (aisquared.config.preprocessing.image.Steps.Resize), 18  
method), 18  
to\_dict() (aisquared.config.preprocessing.image.Steps.SubtractValue), 19  
method), 19  
to\_dict() (aisquared.config.preprocessing.tabular.Steps.DropColumn), 19  
method), 19  
to\_dict() (aisquared.config.preprocessing.tabular.Steps.MinMax), 20  
method), 20  
to\_dict() (aisquared.config.preprocessing.tabular.Steps.OneHot), 20  
method), 20  
to\_dict() (aisquared.config.preprocessing.tabular.Steps.ZScore), 21  
method), 21  
to\_dict() (aisquared.config.preprocessing.tabular.TabularPreprocessing.TabularPreprocessing), 21  
method), 21  
to\_dict() (aisquared.config.preprocessing.text.Steps.ConvertToCase), 21  
method), 21

method), 22  
 to\_dict() (aisquared.config.preprocessing.text.Steps.ConvertTokenizer in aisquared.config.preprocessing.text.Steps),  
 method), 23  
 to\_dict() (aisquared.config.preprocessing.text.Steps.PadSequence in aisquared.config.preprocessing.text.Steps),  
 method), 23  
 to\_dict() (aisquared.config.preprocessing.text.Steps.RemoveCharacterLocation (aisquared.config.preprocessing.text.Steps.PadSequence  
 method), 23  
 to\_dict() (aisquared.config.preprocessing.text.Steps.Tokenize  
 method), 24  
 to\_dict() (aisquared.config.preprocessing.text.Steps.Trim  
 method), 24  
 to\_dict() (aisquared.config.preprocessing.text.TextPreprocessing.TextPreprocessor  
 method), 25  
 to\_dict() (aisquared.config.rendering.BarChartRendering.BarChartRendering  
 method), 26  
 to\_dict() (aisquared.config.rendering.ContainerRendering.ContainerRendering  
 method), 27  
 to\_dict() (aisquared.config.rendering.CustomRendering.CustomRendering  
 method), 28  
 to\_dict() (aisquared.config.rendering.DashboardRendering.DashboardRendering  
 method), 28  
 to\_dict() (aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering  
 method), 29  
 to\_dict() (aisquared.config.rendering.DocumentRendering.DocumentRendering  
 method), 30  
 to\_dict() (aisquared.config.rendering.DoughnutChartRendering.DoughnutChartRendering  
 method), 32  
 to\_dict() (aisquared.config.rendering.FilterRendering.FilterRendering  
 method), 32  
 to\_dict() (aisquared.config.rendering.HTMLTagRendering.HTMLTagRendering  
 method), 33  
 to\_dict() (aisquared.config.rendering.ImageRendering.ImageRendering  
 method), 34  
 to\_dict() (aisquared.config.rendering.LineChartRendering.LineChartRendering  
 method), 36  
 to\_dict() (aisquared.config.rendering.ObjectRendering.ObjectRendering  
 method), 36  
 to\_dict() (aisquared.config.rendering.PieChartRendering.PieChartRendering  
 method), 38  
 to\_dict() (aisquared.config.rendering.SOSRendering.SOSRendering  
 method), 38  
 to\_dict() (aisquared.config.rendering.TableRendering.TableRendering  
 method), 39  
 to\_dict() (aisquared.config.rendering.TextRendering.TextRendering  
 method), 39  
 to\_dict() (aisquared.config.rendering.WordRendering.WordRendering  
 method), 40  
 to\_json() (aisquared.base.BaseObject.BaseObject  
 method), 3  
 token (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 property), 58  
 token (aisquared.platform.DatabricksClient.DatabricksClient  
 property), 66  
 token\_pattern (aisquared.config.preprocessing.text.Steps.Tokenize  
 property), 32  
 property), 24  
 Tokenizer in aisquared.config.preprocessing.text.Steps),  
 23  
 Tokenizer in aisquared.config.preprocessing.text.Steps),  
 24  
 truncate\_location (aisquared.config.preprocessing.text.Steps.PadSequence  
 property), 23  
 underline\_color (aisquared.config.rendering.DocumentRendering.Document  
 property), 30  
 unshare\_model\_with\_group() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 method), 58  
 unshare\_model\_with\_user() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 method), 59  
 update\_group() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 method), 59  
 update\_user() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 method), 60  
 upload\_model() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 method), 61  
 upload\_to\_workspace() (aisquared.platform.DatabricksClient.DatabricksClient  
 method), 62  
 url (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic  
 property), 5  
 url (aisquared.config.analytic.DeployedModel.DeployedModel  
 property), 5  
 url (aisquared.config.GraphConfiguration.GraphConfiguration  
 property), 42  
 url (aisquared.config.ModelConfiguration.ModelConfiguration  
 property), 42  
 url\_locations (aisquared.config.harvesting.QueryParameterHarvester.QueryParameterHarvester  
 property), 13  
 use\_port (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 property), 61  
 username (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient  
 property), 61  
 username (aisquared.platform.DatabricksClient.DatabricksClient  
 property), 66  
 value (aisquared.config.preprocessing.image.Steps.AddValue  
 property), 17  
 value (aisquared.config.preprocessing.image.Steps.DivideValue  
 property), 18  
 value (aisquared.config.preprocessing.image.Steps.MultiplyValue  
 property), 18  
 value (aisquared.config.preprocessing.image.Steps.SubtractValue  
 property), 19  
 value (aisquared.config.rendering.FilterRendering.FilterRendering  
 property), 32

`values` (*aisquared.config.preprocessing.tabular.Steps.OneHot*  
*property*), [20](#)  
`version` (*aisquared.config.GraphConfiguration.GraphConfiguration*  
*property*), [42](#)  
`version` (*aisquared.config.ModelConfiguration.ModelConfiguration*  
*property*), [44](#)  
`vocabulary` (*aisquared.config.preprocessing.text.Steps.ConvertToVocabulary*  
*property*), [23](#)

## W

`where_replace` (*aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering*  
*property*), [29](#)  
`width` (*aisquared.config.rendering.ContainerRendering.ContainerRendering*  
*property*), [27](#)  
`word_list` (*aisquared.config.rendering.WordRendering.WordRendering*  
*property*), [40](#)  
`WordRendering` (class in  
*aisquared.config.rendering.WordRendering*),  
[39](#)  
`words` (*aisquared.config.rendering.DocumentRendering.DocumentRendering*  
*property*), [30](#)

## X

`xOffset` (*aisquared.config.rendering.ContainerRendering.ContainerRendering*  
*property*), [27](#)

## Y

`yOffset` (*aisquared.config.rendering.ContainerRendering.ContainerRendering*  
*property*), [27](#)

## Z

`ZScore` (class in *aisquared.config.preprocessing.tabular.Steps*),  
[20](#)