

---

**aisquared**

**The AI Squared Team**

**Jan 06, 2023**



**CONTENTS:**

<b>1</b>	<b>aisquared</b>	<b>1</b>
1.1	aisquared package . . . . .	1
<b>2</b>	<b>Indices and tables</b>	<b>49</b>
	<b>Python Module Index</b>	<b>51</b>
	<b>Index</b>	<b>53</b>



## AISQUARED

## 1.1 aisquared package

### 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.CustomObject module**

**class** aisquared.base.CustomObject.**CustomObject**(*class\_name: str, \*\*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()
```

(continues on next page)

(continued from previous page)

```
{'className': 'MyClass', 'params': {'key1': 'foo', 'key2': 'bar'}}
)
```

**to\_dict()** → dict

Get the object as a dictionary

### 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:
                                                                Union[None, dict] = 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',
```

(continues on next page)

(continued from previous page)

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

**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: Union[None, dict] =
    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',
 '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.**ReverseMLWorkflow**(*bucket: str, filename: str, column: str, input\_type: str, period: Union[None, int] = 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',
 'fileName': 'file_name',
 'inputType': 'text',
 'column': 'column_name',
 'period': None,
 'secret': ''}}
```

**property** bucket

**property** column

**property** filename

**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[str]*)

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(
    '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[str]*)

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 ‘single-Choice’ 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.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: Union[None, int] =  
None, features: Union[None,  
list] = 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:  
    Union[str,  
    list[str]],  
    url_locations:  
    Union[str,  
    list[str]],  
    at-  
    tributes:  
    Union[str,  
    list[str]])
```

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: Union[None,  
    str] = None, flags: str = 'gu',  
    body_only: bool = False, keywords:  
    Union[None, str, list[str]] = None,  
    limit: Union[None, int] = None)
```

Bases: *BaseObject*

Object to harvest text

Example usage:

```
>>> import aisquared  
>>> my_obj = aisquared.config.harvesting.TextHarvester(  

```

(continues on next page)

(continued from previous page)

```

    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[str],
                                                                    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[str])

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[str], 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: Union[None, int, float] = None,
                                                         max: Union[None, int, float] = 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: Optional[list]
                                                                               = 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: Union[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: Union[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: Union[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[int], 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: Union[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[Union[int, float]], maxs: list[Union[int, float]], columns: Union[None, list[int]] = 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[Union[int, float]], stds: list[Union[int, float]], columns: Union[None, int, list[int]] = 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:*  
*Union[None,*  
*list]*  
 =  
*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: Union[None, int] = 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:*  
*Optional[list] =*  
*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,
                                                                    chart_colors: list[str],
                                                                    chart_labels: list[str],
                                                                    container_id: str,
                                                                    prediction_name_key: str,
                                                                    prediction_value_key: str,
                                                                    prediction_name_value:
                                                                    str, width: str = 'auto',
                                                                    height: str = 'auto',
                                                                    xOffset: str = '0', yOffset:
                                                                    str = '0')
```

Bases: *BaseObject*

Rendering class for rendering a Bar Chart

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.BarChartRendering(
    'my bar chart',
    'barChart1Label',
    'MyBarChart',
    ['red', 'blue'],
    ['label1', 'label2'],
    'my_container',
```

(continues on next page)

(continued from previous page)

```

        'key',
        'value',
        'name'
    )
    >>> my_obj.to_dict()
    {'className': 'BarChartRendering',
     'label': 'my bar chart',
     'params': {'id': 'barChart1Label',
                 'chartName': 'MyBarChart',
                 'chartColors': ['red', 'blue'],
                 'chartLabels': ['label1', 'label2'],
                 'containerId': 'my_container',
                 'predictionNameKey': 'key',
                 'predictionValueKey': 'value',
                 'predictionNameValue': 'name',
                 'width': 'auto',
                 'height': 'auto',
                 'xOffset': '0',
                 'yOffset': '0'}}

```

**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,
                                                                    width: str = 'auto',
                                                                    height: str = 'auto',
                                                                    display: str = 'flex',
                                                                    xOffset: str = '0',
                                                                    yOffset: str = '0',
                                                                    position: str = "",
                                                                    orientation: str =
                                                                    'column')

```

Bases: *BaseObject*

Rendering for a container

Example usage:

```

>>> import aisquared
>>> my_obj = aisquared.config.rendering.ContainerRendering(
    'my container',
    'myContainerID',
    "[data-id='tabpanel-general']"
)
>>> my_obj.to_dict()
{'className': 'ContainerRendering',
 'label': 'my container',
 'params': {'id': 'myContainerID',
             'width': 'auto',

```

(continues on next page)

(continued from previous page)

```
'height': 'auto',
'display': 'flex',
'xOffset': '0',
'yOffset': '0',
'position': '',
'orientation': 'column',
'querySelector': "[data-id='tabpanel-general']"}}
```

**property display**

**property height**

**property id**

**property label**

**property orientation**

**property position**

**property query\_selector**

**to\_dict()** → dict

Get the configuration object as a dictionary

**property width**

**property xOffset**

**property yOffset**

## aisquared.config.rendering.DashboardReplacementRendering module

```
class aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering(anchor_selector: str,
where_replacement: str,
label: str)

Bases: BaseObject
```

Bases: *BaseObject*

Rendering for dashboard replacement

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.DashboardReplacementRendering(
    'test_anchor_selector'
)
```

(continues on next page)

(continued from previous page)

```
>>> 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:
    Union[None, list[str], dict,
    str] = None, documents:
    Union[None, list[str], dict,
    str] = None,
    include_probability: bool
    = False, probability_key:
    str = 'probability',
    underline_color: str =
    'blue', classes:
    Union[None, list[str]] =
    None, threshold_key:
    Union[None, str] = None,
    threshold_value:
    Union[None, int, float] =
    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,
chart_colors: list[str],
chart_labels: list[str],
container_id: str, prediction_name_key: str, prediction_value_key: str, prediction_name_value: str, display_legend: bool,
legend_icon: str, width: str = 'auto', height: str = 'auto', xOffset: str = '0', yOffset: str = '0')
```

Bases: *BaseObject*

Rendering class for rendering a Doughnut Chart

Example usage:

```

>>> import aisquared
>>> my_obj = aisquared.config.rendering.DoughnutChartRendering(
    'my doughnut chart',
    'MyDoughnutChartID',
    'MyDoughnutChart',
    ['red', 'blue'],
    ['label1', 'label2'],
    'my_container_id',
    'key',
    'value',
    'name',
    True,
    'circle'
)
>>> my_obj.to_dict()
{'className': 'DoughnutChartRendering',
 'label': 'my doughnut chart',
 'params': {'id': 'MyDoughnutChartID',
 'chartName': 'MyDoughnutChart',
 'chartColors': ['red', 'blue'],
 'chartLabels': ['label1', 'label2'],
 'containerId': 'my_container_id',
 'predictionNameKey': 'key',
 'predictionValueKey': 'value',
 'predictionNameValue': 'name',
 'displayLegend': True,
 'legendIcon': 'circle',
 'width': 'auto',
 'height': 'auto',
 'xOffset': '0',
 'yOffset': '0'}}

```

**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: Union[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
)

```

(continues on next page)

(continued from previous page)

```

)
>>> 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 = "")

```

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',

```

(continues on next page)



(continued from previous page)

```
'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(color: str = 'blue', thickness: str = '5px', placement: str = 'bottomleft', include_probability: bool = False, badge_color: str = 'white', font_color: str = 'black', font_size: str = '5px', classes: Union[None, list] = None, threshold_key: Union[None, str] = None, threshold_value: Union[None, int, float] = 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': '5px',
'placement': 'bottomleft',
'includeProbability': False,
'badgeColor': 'white',
'fontColor': 'black',
'fontSize': '5px',
'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,
                                                    chart_colors: list[str],
                                                    chart_labels: list[str],
                                                    container_id: str,
                                                    prediction_name_key:
                                                    str,
                                                    prediction_value_key:
                                                    str, predic-
                                                    tion_name_value: str,
                                                    width: str = 'auto',
                                                    height: str = 'auto',
                                                    xOffset: str = '0',
                                                    yOffset: str = '0')
```

Bases: *BaseObject*

Rendering class for rendering a Line Chart

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.LineChartRendering(
    'my line chart',
    'MyLineChartID',
    'MyLineChart',
    ['red', 'blue'],
    ['label1', 'label2'],
    'MyContainerID',
    'name_key',
    'value_key',
    'name_value'
)
>>> my_obj.to_dict()
{'className': 'LineChartRendering',
 'label': 'my line chart',
 'params': {'id': 'MyLineChartID',
            'chartName': 'MyLineChart',
            'chartColors': ['red', 'blue'],
            'chartLabels': ['label1', 'label2'],
            'containerId': 'MyContainerID',
            'predictionNameKey': 'name_key',
```

(continues on next page)

(continued from previous page)

```
'predictionValueKey': 'value_key',
'predictionNameValue': 'name_value',
'width': 'auto',
'height': 'auto',
'xOffset': '0',
'yOffset': '0'}}
```

**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 = '5px', placement: str =
    'bottomleft',
    include_probability: bool =
    False, badge_color: str =
    'white', font_color: str = 'black',
    font_size: str = '5px')
```

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': '5px',
'placement': 'bottomleft',
'includeProbability': False,
'badgeColor': 'white',
'fontColor': 'black',
'fontSize': '5px'}}
```

**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,  
                                                chart_colors: list[str],  
                                                chart_labels: list[str],  
                                                container_id: str,  
                                                prediction_name_key: str,  
                                                prediction_value_key: str,  
                                                prediction_name_value:  
                                                str, display_legend: bool,  
                                                legend_icon: str, width:  
                                                str = 'auto', height: str =  
                                                'auto', xOffset: str = '0',  
                                                yOffset: str = '0')
```

Bases: *BaseObject*

Rendering class for rendering a Pie Chart

Example usage:

```
>>> import aisquared  
>>> my_obj = aisquared.config.rendering.PieChartRendering(  
    'my pie chart',  
    'MyPieChartID',  
    'MyPieChart',  
    ['red', 'blue'],  
    ['label1', 'label2'],  
    'my_container_id',  
    'key',  
    'value',  
    'name',  
    True,  
    'circle'  
)  
>>> my_obj.to_dict()  
{'className': 'PieChartRendering',  
 'label': 'my pie chart',  
 'params': {'id': 'MyPieChartID',  
            'chartName': 'MyPieChart',  
            'chartColors': ['red', 'blue'],  
            'chartLabels': ['label1', 'label2'],  
            'containerId': 'my_container_id',  
            'predictionNameKey': 'key',  
            'predictionValueKey': 'value',  
            'predictionNameValue': 'name',  
            'displayLegend': True,  
            'legendIcon': 'circle',  
            'width': 'auto',  
            'height': 'auto',  
            'xOffset': '0',  
            'yOffset': '0'}}
```

**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 = ""*)

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',
 '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.WordRendering module**

```
class aisquared.config.rendering.WordRendering.WordRendering(word_list: str = 'input', result_key: Union[None, str] = None, content_key: Union[None, str] = None, badge_shape: str = 'star', badge_color: str = 'blue', classes: Union[None, list[str]] = None, threshold_key: Union[None, str] = None, threshold_value: Union[None, int, float] = None)
```

Bases: *BaseObject*

Object for rendering badges on individual words

Example usage:

```
>>> import aisquared
>>> my_obj = aisquared.config.rendering.WordRendering()
>>> my_obj.to_dict()
{'className': 'WordRendering',
 'params': {'wordList': 'input',
 'resultKey': None,
 'contentKey': None,
 'badgeShape': 'star',
 'badgeColor': 'blue',
 'classes': None,
 'thresholdKey': None,
 'thresholdValue': None}}
```

**property** badge\_color

**property** badge\_shape

**property** classes

**property** content\_key

**property** result\_key

**property** threshold\_key

**property** threshold\_value

**to\_dict()** → dict

Get the configuration object as a dictionary

**property** word\_list

## Module contents

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

## Submodules

### aisquared.config.GraphConfiguration module

```
class aisquared.config.GraphConfiguration.GraphConfiguration(name: str, stage: str =
    'experimental', version: Union[None,
    int] = None, description: str = "",
    mlflow_uri: Union[None, str] =
    None, mlflow_user: Union[None, str]
    = None, mlflow_token: Union[None,
    str] = None, owner: Union[None,
    str] = None, url: str = '*', auto_run:
    bool = False)
```

Bases: [\*BaseObject\*](#)

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

**add\_node**(step: [\*BaseObject\*](#), dependencies: Union[None, int, list[int]] = 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: Union[None, str] = None, dtype: Union[None, str] = 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

**get\_filenames**() → list[str]

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:
    Union[None, BaseObject,
    list[aisquared.base.BaseObject.BaseObject]],
    preprocessing_steps: Union[None,
    BaseObject,
    list[aisquared.base.BaseObject.BaseObject]],
    analytic: Union[BaseObject,
    list[aisquared.base.BaseObject.BaseObject]],
    postprocessing_steps: Union[None,
    BaseObject,
    list[aisquared.base.BaseObject.BaseObject]],
    rendering_steps: Union[None,
    BaseObject,
    list[aisquared.base.BaseObject.BaseObject]],
    feedback_steps: Union[None,
    BaseObject,
    list[aisquared.base.BaseObject.BaseObject]]
    = None, stage: str = 'experimental',
    version: Optional[int] = None,
    description: str = "", mlflow_uri:
    Union[None, str] = None,
    mlflow_user: Union[None, str] =
    None, mlflow_token: Union[None,
    str] = None, owner: Union[None,
    str] = None, url: str = '*', auto_run:
    bool = False)
```

Bases: *BaseObject*

Configuration object for deploying a model or analytic

```
property analytic
property analytic_dict
property auto_run
```



**compile**(filename: Union[None, str] = None, dtype: Union[None, str] = 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 feedback\_dict**

**property feedback\_steps**

**get\_model\_filenames()** → list[str]

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.AISquaredPlatformClient module

**exception** `aisquared.platform.AISquaredPlatformClient.AISquaredAPIException`

Bases: `Exception`

**class** `aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient`

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()
Connection successful
200
```

**add\_user\_to\_group**(*group\_id*, *user\_id*)

Not yet implemented

**property** `base_url`: `str`

The base URL associated with the client

**delete\_model**(*id*: `str`, *port*: `int` = 8080) → `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

**Returns**

**success** – Whether the action was successful

**Return type**

bool

**get\_model**(*id: str, port: int = 8080*) → 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

**Returns**

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

**Return type**

dictionary

**get\_model\_id\_by\_name**(*model\_name: str*) → 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

**Returns**

**model\_id** – The model's ID

**Return type**

str

**get\_user\_id\_by\_name**(*name: str*) → 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

**Returns**

**id** – The ID of the user

**Return type**

*str*

**get\_user\_usage\_metrics**(*user\_id*, *port*=8080)

Not yet implemented

**property headers**

Headers used for authentication with the AI Squared Platform

**list\_group\_users**(*group\_id*: *str*, *as\_df*: *bool* = *True*, *port*: *int* = 8083) → Union[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 to use

**Returns**

**users** – The response from the API

**Return type**

pandas DataFrame or dictionary

**list\_groups**(*as\_df*: *bool* = *True*, *port*: *int* = 8083) → Union[DataFrame, dict]

List all groups

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

**Parameters**

- **as\_df** (*bool* (default *True*)) – Whether to return the result as a pandas DataFrame
- **port** (*int* (default 8083)) – The API port for the call

**Returns**

**groups** – The response from the API

**Return type**

pandas DataFrame or dictionary

**list\_model\_feedback**(*model\_id*, *port*=8080)

Not yet implemented

**list\_model\_prediction\_feedback**(*model\_id*)

Not yet implemented

**list\_model\_predictions**(*model\_id*, *port*=8080)

Not yet implemented

**list\_model\_users**(*id*: str, *as\_df*: bool = True, *port*: int = 8080) → Union[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

#### Returns

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

#### Return type

pandas DataFrame or dictionary

**list\_models**(*as\_df*: bool = True, *port*: int = 8080) → Union[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** (int (default 8080)) – The API port for the call

#### Returns

**models** – The models

#### Return type

pandas DataFrame or dictionary

**list\_prediction\_feedback**(*prediction\_id*, *port*=8080)

Not yet implemented

**list\_users**(*as\_df*: bool = True, *port*: int = 8080) → Union[DataFrame, dict]

List all users

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

**Parameters**

- **as\_df** (*bool (default True)*) – Whether to return the data as a Pandas DataFrame
- **port** (*int (default 8080)*) – The API port for the call

**Returns**

**users** – The response from the API

**Return type**

pandas DataFrame or dictionary

**login**(*url: Union[None, str] = None, port: int = 8080, username: Union[None, str] = None, password: Union[None, str] = 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 (default 8080)*) – The API port for the call
- **username** (*str or None (default None)*) – The username
- **password** (*str or None (default None)*) – The password

**property password:** str

The password associated with the client

**remove\_user\_from\_group**(*group\_id, user\_id*)

Not yet implemented

**share\_model\_with\_group**(*model\_id, group\_id, port=8083*)

Not yet implemented

**share\_model\_with\_user**(*model\_id: str, user\_id: str, port: int = 8080*) → 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

**Returns**

**success** – Whether the action was successful

**Return type**

bool

**test\_connection**(*port: int = 8080*) → int

Test whether there is a healthy connection to the platform

```
>>> import aisquared
>>> client = aisquared.platform.AISquaredPlatformClient()
>>> client.test_connection()
Connection successful
200
```

**Parameters**

**port** (*int* (*default* 8080)) – The API port for the call

**Returns**

**status\_code** – The status code when checking the health API

**Return type**

int

**property token:** str

The token associated with the client

**unshare\_model\_with\_user**(*model\_id: str, user\_id: str, port: int = 8080*) → 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

**Returns**

**success** – Whether the action was successful

**Return type**

bool

**upload\_model**(*model\_file: str, port: int = 8081*) → 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

**Returns**

**successful** – Whether the action was successful

**Return type**

bool

**property username:** str

The username associated with the client

**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: Union[None, dict] =
                                             None, additional_functions_file: Union[None, str] = 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 MANN 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_mann_model(model: str, custom_objects: dict)
```

Load a MANN model with custom objects

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

```
aisquared.serving.get_remote_prediction.get_remote_prediction(data: Union[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: Union[int, tuple[int]], num_outputs: int, output_activation: str, size: str = 'small', vocab_size: Union[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.utils.mimic_model(trained_model: BaseEstimator, nnet: Model, training_data: ndarray, test_data: ndarray, test_labels: ndarray, problem_type: str, loss: str, metrics: Union[str, list[str]], 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>

**1.1.2 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.



## INDICES AND TABLES

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



## PYTHON MODULE INDEX

### a

aisquared, 47

aisquared.base, 2

aisquared.base.BaseObject, 1

aisquared.base.CustomObject, 1

aisquared.base.rendering, 2

aisquared.base.stages, 2

aisquared.config, 38

aisquared.config.analytic, 5

aisquared.config.analytic.DeployedAnalytic, 2

aisquared.config.analytic.DeployedModel, 3

aisquared.config.analytic.LocalAnalytic, 4

aisquared.config.analytic.LocalModel, 4

aisquared.config.analytic.ReverseMLWorkflow,  
5

aisquared.config.feedback, 8

aisquared.config.feedback.BinaryFeedback, 6

aisquared.config.feedback.ModelFeedback, 6

aisquared.config.feedback.MulticlassFeedback,  
7

aisquared.config.feedback.QualitativeFeedback,  
7

aisquared.config.feedback.RegressionFeedback,  
8

aisquared.config.feedback.SimpleFeedback, 8

aisquared.config.GraphConfiguration, 35

aisquared.config.harvesting, 11

aisquared.config.harvesting.ImageHarvester, 9

aisquared.config.harvesting.InputHarvester, 9

aisquared.config.harvesting.QueryParameterHarvester,  
10

aisquared.config.harvesting.TextHarvester, 10

aisquared.config.ModelConfiguration, 36

aisquared.config.postprocessing, 13

aisquared.config.postprocessing.BinaryClassification,  
11

aisquared.config.postprocessing.MulticlassClassification,  
12

aisquared.config.postprocessing.ObjectDetection,  
12

aisquared.config.postprocessing.Regression,  
13

aisquared.config.preprocessing, 22

aisquared.config.preprocessing.image, 16

aisquared.config.preprocessing.image.ImagePreprocessing,  
13

aisquared.config.preprocessing.image.Steps,  
14

aisquared.config.preprocessing.tabular, 19

aisquared.config.preprocessing.tabular.Steps,  
16

aisquared.config.preprocessing.tabular.TabularPreprocessing,  
18

aisquared.config.preprocessing.text, 22

aisquared.config.preprocessing.text.Steps, 19

aisquared.config.preprocessing.text.TextPreprocessing,  
21

aisquared.config.rendering, 35

aisquared.config.rendering.BarChartRendering,  
22

aisquared.config.rendering.ContainerRendering,  
23

aisquared.config.rendering.DashboardReplacementRendering,  
24

aisquared.config.rendering.DocumentRendering,  
25

aisquared.config.rendering.DoughnutChartRendering,  
26

aisquared.config.rendering.FilterRendering,  
27

aisquared.config.rendering.HTMLTagRendering,  
28

aisquared.config.rendering.ImageRendering, 29

aisquared.config.rendering.LineChartRendering,  
30

aisquared.config.rendering.ObjectRendering,

aisquared.config.rendering.PieChartRendering,  
32

aisquared.config.rendering.SOSRendering, 33

aisquared.config.rendering.TableRendering, 33

aisquared.config.rendering.WordRendering, 34

aisquared.logging, 38

aisquared.platform, 44

`aisquared.platform.AISquaredPlatformClient,`  
    38  
`aisquared.serving,` 45  
`aisquared.serving.deploy_model,` 44  
`aisquared.serving.get_remote_prediction,` 45  
`aisquared.utils,` 47  
`aisquared.utils.utils,` 46



## A

- `add_node()` (*aisquared.config.GraphConfiguration.GraphConfiguration* module), 35
- `add_question()` (*aisquared.config.feedback.ModelFeedback.ModelFeedback* module), 6
- `add_question()` (*aisquared.config.feedback.QualitativeFeedback.QualitativeFeedback* module), 7
- `add_step()` (*aisquared.config.preprocessing.image.ImagePreprocessing.ImagePreprocessor* module), 14
- `add_step()` (*aisquared.config.preprocessing.tabular.TabularPreprocessing.TabularPreprocessor* module), 18
- `add_step()` (*aisquared.config.preprocessing.text.TextPreprocessing.TextPreprocessor* module), 21
- `add_user_to_group()` (*aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient* module), 38
- `AddValue` (class in *aisquared.config.preprocessing.image.Steps*), 14
- `aisquared` module, 47
- `aisquared.base` module, 2
- `aisquared.base.BaseObject` module, 1
- `aisquared.base.CustomObject` module, 1
- `aisquared.base.rendering` module, 2
- `aisquared.base.stages` module, 2
- `aisquared.config` module, 38
- `aisquared.config.analytic` module, 5
- `aisquared.config.analytic.DeployedAnalytic` module, 2
- `aisquared.config.analytic.DeployedModel` module, 3
- `aisquared.config.analytic.LocalAnalytic` module, 4
- `aisquared.config.analytic.LocalModel` module, 4
- `aisquared.config.analytic.ReverseMLWorkflow` module, 5
- `aisquared.config.feedback` module, 8
- `aisquared.config.feedback.BinaryFeedback` module, 6
- `aisquared.config.feedback.ModelFeedback` module, 6
- `aisquared.config.feedback.MulticlassFeedback` module, 7
- `aisquared.config.feedback.QualitativeFeedback` module, 7
- `aisquared.config.feedback.RegressionFeedback` module, 8
- `aisquared.config.feedback.SimpleFeedback` module, 8
- `aisquared.config.GraphConfiguration` module, 35
- `aisquared.config.harvesting` module, 11
- `aisquared.config.harvesting.ImageHarvester` module, 9
- `aisquared.config.harvesting.InputHarvester` module, 9
- `aisquared.config.harvesting.QueryParameterHarvester` module, 10
- `aisquared.config.harvesting.TextHarvester` module, 10
- `aisquared.config.ModelConfiguration` module, 36
- `aisquared.config.postprocessing` module, 13
- `aisquared.config.postprocessing.BinaryClassification` module, 11
- `aisquared.config.postprocessing.MulticlassClassification` module, 12
- `aisquared.config.postprocessing.ObjectDetection` module, 12
- `aisquared.config.postprocessing.Regression` module, 13
- `aisquared.config.preprocessing` module, 22

aisquared.config.preprocessing.image	aisquared.serving
module, 16	module, 45
aisquared.config.preprocessing.image.ImagePreprocessing	aisquared.serving.deploy_model
module, 13	module, 44
aisquared.config.preprocessing.image.Steps	aisquared.serving.get_remote_prediction
module, 14	module, 45
aisquared.config.preprocessing.tabular	aisquared.utils
module, 19	module, 47
aisquared.config.preprocessing.tabular.Steps	aisquared.utils.utils
module, 16	module, 46
aisquared.config.preprocessing.tabular.TabularPreprocessing	aisquared.utils.exception, 38
module, 18	AISquaredPlatformClient (class in
aisquared.config.preprocessing.text	aisquared.platform.AISquaredPlatformClient),
module, 22	38
aisquared.config.preprocessing.text.Steps	all (aisquared.config.analytic.LocalAnalytic.LocalAnalytic
module, 19	property), 4
aisquared.config.preprocessing.text.TextPreprocessing	analytic (aisquared.config.ModelConfiguration.ModelConfiguration
module, 21	property), 36
aisquared.config.rendering	analytic_dict (aisquared.config.ModelConfiguration.ModelConfiguration
module, 35	property), 36
aisquared.config.rendering.BarChartRendering	anchor_selector (aisquared.config.rendering.DashboardReplacementRe
module, 22	property), 25
aisquared.config.rendering.ContainerRendering	attributes (aisquared.config.harvesting.QueryParameterHarvester.Query
module, 23	property), 10
aisquared.config.rendering.DashboardReplacementRendering	auto_run (aisquared.config.GraphConfiguration.GraphConfiguration
module, 24	property), 35
aisquared.config.rendering.DocumentRendering	auto_run (aisquared.config.ModelConfiguration.ModelConfiguration
module, 25	property), 36
aisquared.config.rendering.DoughnutChartRendering	
module, 26	<b>B</b>
aisquared.config.rendering.FilterRendering	badge_color (aisquared.config.rendering.ImageRendering.ImageRendering
module, 27	property), 29
aisquared.config.rendering.HTMLTagRendering	badge_color (aisquared.config.rendering.ObjectRendering.ObjectRenderi
module, 28	property), 31
aisquared.config.rendering.ImageRendering	badge_color (aisquared.config.rendering.WordRendering.WordRendering
module, 29	property), 34
aisquared.config.rendering.LineChartRendering	badge_shape (aisquared.config.rendering.WordRendering.WordRendering
module, 30	property), 34
aisquared.config.rendering.ObjectRendering	BarChartRendering (class in
module, 31	aisquared.config.rendering.BarChartRendering),
aisquared.config.rendering.PieChartRendering	22
module, 32	base_url (aisquared.platform.AISquaredPlatformClient.AISquaredPlatfor
aisquared.config.rendering.SOSRendering	property), 38
module, 33	BaseObject (class in aisquared.base.BaseObject), 1
aisquared.config.rendering.TableRendering	BinaryClassification (class in
module, 33	aisquared.config.postprocessing.BinaryClassification),
aisquared.config.rendering.WordRendering	11
module, 34	BinaryFeedback (class in
aisquared.logging	aisquared.config.feedback.BinaryFeedback), 6
module, 38	body_only (aisquared.config.harvesting.TextHarvester.TextHarvester
aisquared.platform	property), 11
module, 44	bucket (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflo
aisquared.platform.AISquaredPlatformClient	property), 5
module, 38	

## C

[can\\_toggle](#) ([aisquared.config.rendering.SOSRendering.SOSRendering](#) property), 33  
[classes](#) ([aisquared.config.rendering.DocumentRendering.DocumentRendering](#) property), 25  
[classes](#) ([aisquared.config.rendering.ImageRendering.ImageRendering](#) property), 29  
[classes](#) ([aisquared.config.rendering.WordRendering.WordRendering](#) property), 34  
[color](#) ([aisquared.config.preprocessing.image.Steps.ConvertToColor](#) property), 14  
[color](#) ([aisquared.config.rendering.ImageRendering.ImageRendering](#) property), 29  
[color](#) ([aisquared.config.rendering.ObjectRendering.ObjectRendering](#) property), 31  
[column](#) ([aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow](#) property), 5  
[column](#) ([aisquared.config.preprocessing.tabular.Steps.DropColumn](#) property), 16  
[column](#) ([aisquared.config.preprocessing.tabular.Steps.OneHot](#) property), 17  
[columns](#) ([aisquared.config.preprocessing.tabular.Steps.MinMax](#) property), 17  
[columns](#) ([aisquared.config.preprocessing.tabular.Steps.ZScore](#) property), 18  
[compile\(\)](#) ([aisquared.config.GraphConfiguration.GraphConfiguration](#) method), 35  
[compile\(\)](#) ([aisquared.config.ModelConfiguration.ModelConfiguration](#) method), 36  
[ContainerRendering](#) (class in [aisquared.config.rendering.ContainerRendering](#)), 23  
[content\\_key](#) ([aisquared.config.rendering.WordRendering.WordRendering](#) property), 34  
[ConvertToCase](#) (class in [aisquared.config.preprocessing.text.Steps](#)), 19  
[ConvertToColor](#) (class in [aisquared.config.preprocessing.image.Steps](#)), 14  
[ConvertToVocabulary](#) (class in [aisquared.config.preprocessing.text.Steps](#)), 19  
[CustomObject](#) (class in [aisquared.base.CustomObject](#)), 1

## D

[DashboardReplacementRendering](#) (class in [aisquared.config.rendering.DashboardReplacementRendering](#)), 24  
[delete\\_model\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 38  
[deploy\\_model\(\)](#) (in module [aisquared.serving.deploy\\_model](#)), 44

[DeployedAnalytic](#) (class in [aisquared.config.analytic.DeployedAnalytic](#)), 2  
[DeployedModel](#) (class in [aisquared.config.analytic.DeployedModel](#)), 3  
[description](#) ([aisquared.config.GraphConfiguration.GraphConfiguration](#) property), 35  
[description](#) ([aisquared.config.ModelConfiguration.ModelConfiguration](#) property), 37  
[display](#) ([aisquared.config.rendering.ContainerRendering.ContainerRendering](#) property), 24  
[DivideValue](#) (class in [aisquared.config.preprocessing.image.Steps](#)), 14  
[DocumentRendering](#) (class in [aisquared.config.rendering.DocumentRendering](#)), 25  
[documents](#) ([aisquared.config.rendering.DocumentRendering.DocumentRendering](#) property), 26  
[DoughnutChartRendering](#) (class in [aisquared.config.rendering.DoughnutChartRendering](#)), 26  
[DropColumn](#) (class in [aisquared.config.preprocessing.tabular.Steps](#)), 16  
**F**  
[features](#) ([aisquared.config.harvesting.InputHarvester.InputHarvester](#) property), 9  
[feedback\\_dict](#) ([aisquared.config.ModelConfiguration.ModelConfiguration](#) property), 37  
[feedback\\_steps](#) ([aisquared.config.ModelConfiguration.ModelConfiguration](#) property), 37  
[filename](#) ([aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow](#) property), 5  
[FilterRendering](#) (class in [aisquared.config.rendering.FilterRendering](#)), 27  
[flags](#) ([aisquared.config.harvesting.TextHarvester.TextHarvester](#) property), 11  
[font\\_color](#) ([aisquared.config.rendering.ImageRendering.ImageRendering](#) property), 29  
[font\\_color](#) ([aisquared.config.rendering.ObjectRendering.ObjectRendering](#) property), 31  
[font\\_size](#) ([aisquared.config.rendering.ImageRendering.ImageRendering](#) property), 29  
[font\\_size](#) ([aisquared.config.rendering.ObjectRendering.ObjectRendering](#) property), 31  
**G**  
[get\\_filenames\(\)](#) ([aisquared.config.GraphConfiguration.GraphConfiguration](#) method), 35  
[get\\_model\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 39  
[get\\_model\(\)](#) (in module [aisquared.utils.utils](#)), 46

get\_model\_filenames() (aisquared.config.ModelConfiguration.ModelConfiguration (aisquared.config.rendering.ImageRendering.ImageRendering method), 37  
get\_model\_id\_by\_name() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient (aisquared.config.rendering.ObjectRendering.ObjectRendering method), 39  
get\_remote\_prediction() (in module aisquared.serving.get\_remote\_prediction), 45  
get\_user\_id\_by\_name() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient (aisquared.config.analytic.LocalAnalytic.LocalAnalytic method), 39  
get\_user\_usage\_metrics() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient (aisquared.config.analytic.LocalModel.LocalModel method), 40  
GraphConfiguration (class in aisquared.config.GraphConfiguration), 35  
H  
harvester\_dict (aisquared.config.ModelConfiguration.ModelConfiguration (aisquared.config.harvesting.InputHarvester.InputHarvester property), 37  
harvesting\_steps (aisquared.config.ModelConfiguration.ModelConfiguration (aisquared.config.rendering.FilterRendering.FilterRendering property), 37  
header (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic (aisquared.config.analytic.DeployedModel.DeployedModel property), 3  
header (aisquared.config.analytic.DeployedModel.DeployedModel (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 3  
headers (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient (aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering property), 40  
height (aisquared.config.rendering.ContainerRendering.ContainerRendering (aisquared.config.rendering.SOSRendering.SOSRendering property), 24  
how (aisquared.config.harvesting.ImageHarvester.ImageHarvester (aisquared.config.feedback.BinaryFeedback.BinaryFeedback property), 9  
how (aisquared.config.harvesting.TextHarvester.TextHarvester (aisquared.config.feedback.MulticlassFeedback.MulticlassFeedback property), 11  
HTMLTagRendering (class in aisquared.config.rendering.HTMLTagRendering), 28  
I  
id (aisquared.config.rendering.ContainerRendering.ContainerRendering (aisquared.config.postprocessing.ObjectDetection.ObjectDetection property), 24  
ImageHarvester (class in aisquared.config.harvesting.ImageHarvester), 9  
ImagePreprocessor (class in aisquared.config.preprocessing.image.ImagePreprocessor), 13  
ImageRendering (class in aisquared.config.rendering.ImageRendering), 29  
include\_probability (aisquared.config.rendering.DocumentRendering.DocumentRendering (aisquared.config.rendering.DocumentRendering property), 26  
include\_probability (aisquared.config.rendering.ObjectRendering.ObjectRendering (aisquared.config.rendering.ObjectRendering property), 29  
include\_probability (aisquared.config.rendering.ObjectRendering.ObjectRendering (aisquared.config.rendering.ObjectRendering property), 31  
input\_type (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic (aisquared.config.analytic.DeployedModel.DeployedModel property), 3  
input\_type (aisquared.config.analytic.DeployedModel.DeployedModel (aisquared.config.analytic.LocalAnalytic.LocalAnalytic property), 4  
input\_type (aisquared.config.analytic.LocalModel.LocalModel (aisquared.config.analytic.LocalModel.LocalModel property), 4  
input\_type (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow.ReverseMLWorkflow (aisquared.config.harvesting.InputHarvester.InputHarvester property), 5  
input\_type (aisquared.config.harvesting.InputHarvester.InputHarvester (aisquared.config.harvesting.InputHarvester.InputHarvester property), 9  
InputHarvester (class in aisquared.config.harvesting.InputHarvester), 9  
K  
key (aisquared.config.rendering.FilterRendering.FilterRendering (aisquared.config.rendering.FilterRendering property), 28  
L  
label (aisquared.config.rendering.ContainerRendering.ContainerRendering (aisquared.config.rendering.ContainerRendering property), 24  
label (aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering (aisquared.config.rendering.DashboardReplacementRendering property), 25  
label (aisquared.config.rendering.SOSRendering.SOSRendering (aisquared.config.rendering.SOSRendering property), 33  
label\_map (aisquared.config.feedback.BinaryFeedback.BinaryFeedback (aisquared.config.feedback.BinaryFeedback property), 6  
label\_map (aisquared.config.feedback.MulticlassFeedback.MulticlassFeedback (aisquared.config.feedback.MulticlassFeedback property), 7  
label\_map (aisquared.config.postprocessing.BinaryClassification.BinaryClassification (aisquared.config.postprocessing.BinaryClassification property), 11  
label\_map (aisquared.config.postprocessing.MulticlassClassification.MulticlassClassification (aisquared.config.postprocessing.MulticlassClassification property), 12  
label\_map (aisquared.config.postprocessing.ObjectDetection.ObjectDetection (aisquared.config.postprocessing.ObjectDetection property), 12  
length (aisquared.config.preprocessing.text.Steps.PadSequences (aisquared.config.preprocessing.text.Steps.PadSequences property), 20  
limit (aisquared.config.harvesting.TextHarvester.TextHarvester (aisquared.config.harvesting.TextHarvester property), 11  
LineChartRendering (class in aisquared.config.rendering.LineChartRendering), 30  
list\_group\_users() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient (aisquared.config.rendering.DocumentRendering.DocumentRendering method), 40  
list\_groups() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient (aisquared.config.rendering.DocumentRendering.DocumentRendering method), 40

[list\\_model\\_feedback\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 40  
[list\\_model\\_prediction\\_feedback\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 41  
[list\\_model\\_predictions\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 41  
[list\\_model\\_users\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 41  
[list\\_models\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 41  
[list\\_prediction\\_feedback\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 41  
[list\\_users\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 41  
[load\\_mann\\_model\(\)](#) (in module [aisquared.serving.deploy\\_model](#)), 44  
[LocalAnalytic](#) (class in [aisquared.config.analytic.LocalAnalytic](#)), 4  
[LocalModel](#) (class in [aisquared.config.analytic.LocalModel](#)), 4  
[login\(\)](#) ([aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient](#) method), 42  
[lowercase](#) ([aisquared.config.preprocessing.text.Steps.ConvertToLowerCase](#) property), 19  
**M**  
[max](#) ([aisquared.config.postprocessing.Regression.Regression](#) property), 13  
[max\\_length](#) ([aisquared.config.harvesting.InputHarvester.InputHarvester](#) property), 9  
[max\\_vocab](#) ([aisquared.config.preprocessing.text.Steps.ConvertToVocabulary](#) property), 19  
[maxs](#) ([aisquared.config.preprocessing.tabular.Steps.MinMax](#) property), 17  
[means](#) ([aisquared.config.preprocessing.tabular.Steps.ZScore](#) property), 18  
[method](#) ([aisquared.config.preprocessing.image.Steps.Resize](#) property), 15  
[mimic\\_model\(\)](#) (in module [aisquared.utils.utils](#)), 46  
[min](#) ([aisquared.config.postprocessing.Regression.Regression](#) property), 13  
[MinMax](#) (class in [aisquared.config.preprocessing.tabular.Steps](#)), 16  
[mins](#) ([aisquared.config.preprocessing.tabular.Steps.MinMax](#) property), 17  
[mlflow\\_token](#) ([aisquared.config.GraphConfiguration.GraphConfiguration](#) property), 35  
[mlflow\\_token](#) ([aisquared.config.ModelConfiguration.ModelConfiguration](#) property), 37  
[mlflow\\_uri](#) ([aisquared.config.GraphConfiguration.GraphConfiguration](#) property), 35  
[mlflow\\_uri](#) ([aisquared.config.ModelConfiguration.ModelConfiguration](#) property), 37  
[mlflow\\_user](#) ([aisquared.config.ModelConfiguration.ModelConfiguration](#) property), 37  
[ModelConfiguration](#) (class in [aisquared.config.ModelConfiguration](#)), 36  
[ModelFeedback](#) (class in [aisquared.config.feedback.ModelFeedback](#)), 6  
[module](#)  
[aisquared](#), 47  
[aisquared.base](#), 2  
[aisquared.base.BaseObject](#), 1  
[aisquared.base.CustomObject](#), 1  
[aisquared.base.rendering](#), 2  
[aisquared.base.stages](#), 2  
[aisquared.config](#), 38  
[aisquared.config.analytic](#), 5  
[aisquared.config.analytic.DeployedAnalytic](#), 2  
[aisquared.config.analytic.DeployedModel](#), 3  
[aisquared.config.analytic.LocalAnalytic](#), 4  
[aisquared.config.analytic.LocalModel](#), 4  
[aisquared.config.analytic.ReverseMLWorkflow](#), 5  
[aisquared.config.feedback](#), 8  
[aisquared.config.feedback.BinaryFeedback](#), 6  
[aisquared.config.feedback.ModelFeedback](#), 6  
[aisquared.config.feedback.MulticlassFeedback](#), 7  
[aisquared.config.feedback.QualitativeFeedback](#), 7  
[aisquared.config.feedback.RegressionFeedback](#), 8  
[aisquared.config.feedback.SimpleFeedback](#), 8  
[aisquared.config.GraphConfiguration](#), 35  
[aisquared.config.harvesting](#), 11  
[aisquared.config.harvesting.ImageHarvester](#), 9  
[aisquared.config.harvesting.InputHarvester](#), 9  
[aisquared.config.harvesting.QueryParameterHarvester](#), 10  
[aisquared.config.harvesting.TextHarvester](#), 10  
[aisquared.config.ModelConfiguration](#), 36



[aisquared.config.postprocessing](#), 13  
[aisquared.config.postprocessing.BinaryClassification](#), 34  
[aisquared.config.postprocessing.MulticlassClassification](#), 38  
[aisquared.config.postprocessing.ObjectDetection](#), 38  
[aisquared.config.postprocessing.Regression](#), 45  
[aisquared.config.preprocessing](#), 22  
[aisquared.config.preprocessing.image](#), 16  
[aisquared.config.preprocessing.image.ImagePreprocessing](#), 12  
[aisquared.config.preprocessing.image.Steps](#), 12  
[aisquared.config.preprocessing.tabular](#), 7  
[aisquared.config.preprocessing.tabular.Steps](#), 15  
[aisquared.config.preprocessing.tabular.TabularPreprocessing](#), 15  
[aisquared.config.preprocessing.text](#), 22  
[aisquared.config.preprocessing.text.Steps](#), name ([aisquared.config.GraphConfiguration.GraphConfiguration](#) property), 36  
[aisquared.config.preprocessing.text.TextPreprocessing](#), name ([aisquared.config.ModelConfiguration.ModelConfiguration](#) property), 37  
[aisquared.config.rendering](#), 35  
[aisquared.config.rendering.BarChartRendering](#), 22  
[aisquared.config.rendering.ContainerRendering](#), 23  
[aisquared.config.rendering.DashboardReplacementRendering](#), 24  
[aisquared.config.rendering.DocumentRendering](#), 31  
[aisquared.config.rendering.DoughnutChartRendering](#), 17  
[aisquared.config.rendering.FilterRendering](#), 27  
[aisquared.config.rendering.HTMLTagRendering](#), 28  
[aisquared.config.rendering.ImageRendering](#), 29  
[aisquared.config.rendering.LineChartRendering](#), 30  
[aisquared.config.rendering.ObjectRendering](#), 31  
[aisquared.config.rendering.PieChartRendering](#), 32  
[aisquared.config.rendering.SOSRendering](#), 33  
[aisquared.config.rendering.TableRendering](#), 33  
[aisquared.config.rendering.WordRendering](#), 34  
[aisquared.logging](#), 38  
[aisquared.platform](#), 44  
[aisquared.platform.AISquaredPlatformClient](#), 38  
[aisquared.serving](#), 45  
[aisquared.serving.deploy\\_model](#), 44  
[aisquared.serving.get\\_remote\\_prediction](#), 45  
[aisquared.utils](#), 47  
[aisquared.utils.utils](#), 46  
[aisquared.config.postprocessing.MulticlassClassification](#) (class in [aisquared.config.postprocessing.MulticlassClassification](#)), 12  
[aisquared.config.feedback.MulticlassFeedback](#) (class in [aisquared.config.feedback.MulticlassFeedback](#)), 7  
[aisquared.config.preprocessing.image.Steps.MultiplyValue](#) (class in [aisquared.config.preprocessing.image.Steps](#)), 15  
[aisquared.config.preprocessing.tabular.Steps.ConvertToVocabulary](#) (class in [aisquared.config.preprocessing.tabular.Steps](#)), 19  
[aisquared.config.rendering.ContainerRendering.ContainerRendering](#) (class in [aisquared.config.rendering.ContainerRendering](#)), 24  
[aisquared.config.GraphConfiguration.GraphConfiguration](#) (class in [aisquared.config.GraphConfiguration](#)), 36  
[aisquared.config.ModelConfiguration.ModelConfiguration](#) (class in [aisquared.config.ModelConfiguration](#)), 37  
[aisquared.config.preprocessing.text.Steps.PadSequences](#) (class in [aisquared.config.preprocessing.text.Steps](#)), 20  
[aisquared.config.preprocessing.text.Steps.PadSequences](#) (class in [aisquared.config.preprocessing.text.Steps](#)), 20  
[aisquared.config.preprocessing.text.Steps.PadSequences](#) (class in [aisquared.config.preprocessing.text.Steps](#)), 20

password (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient), 42  
 path (aisquared.config.analytic.LocalAnalytic.LocalAnalytic property), 4  
 path (aisquared.config.analytic.LocalModel.LocalModel property), 4  
 period (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow property), 5  
 PieChartRendering (class in aisquared.config.rendering.PieChartRendering), 32  
 placement (aisquared.config.rendering.ImageRendering.ImageRendering property), 29  
 placement (aisquared.config.rendering.ObjectRendering.ObjectRendering property), 31  
 position (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 24  
 postprocessor\_dict (aisquared.config.ModelConfiguration.ModelConfiguration property), 37  
 postprocessing\_steps (aisquared.config.ModelConfiguration.ModelConfiguration property), 37  
 prediction\_key (aisquared.config.rendering.DocumentRendering.DocumentRendering property), 26  
 preprocessor\_dict (aisquared.config.ModelConfiguration.ModelConfiguration property), 37  
 preprocessing\_steps (aisquared.config.ModelConfiguration.ModelConfiguration property), 37  
 preserve\_aspect\_ratio (aisquared.config.preprocessing.image.Steps.Resize property), 15  
 probability\_key (aisquared.config.rendering.DocumentRendering.DocumentRendering property), 26  
**Q**  
 qualifier (aisquared.config.rendering.FilterRendering.FilterRendering property), 28  
 QualitativeFeedback (class in aisquared.config.feedback.QualitativeFeedback), 7  
 query\_keys (aisquared.config.harvesting.QueryParameterHarvester.QueryParameterHarvester property), 10  
 query\_selector (aisquared.config.rendering.ContainerRendering.ContainerRendering property), 24  
 QueryParameterHarvester (class in aisquared.config.harvesting.QueryParameterHarvester), 10  
**R**  
 regex (aisquared.config.harvesting.TextHarvester.TextHarvester property), 11  
 Regression (class in aisquared.config.postprocessing.Regression.Regression), 13  
 RegressionFeedback (class in aisquared.config.feedback.RegressionFeedback), 8  
 remove\_digits (aisquared.config.preprocessing.text.Steps.RemoveCharacters property), 20  
 remove\_punctuation (aisquared.config.preprocessing.text.Steps.RemoveCharacters property), 20  
 remove\_user\_from\_group() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient method), 42  
 RemoveCharacters (class in aisquared.config.preprocessing.text.Steps.RemoveCharacters), 20  
 render\_results (aisquared.config.ModelConfiguration.ModelConfiguration property), 37  
 RenderingEngine (aisquared.config.ModelConfiguration.ModelConfiguration property), 37  
 Resize (aisquared.config.preprocessing.image.Steps.Resize property), 15  
 result\_key (aisquared.config.rendering.WordRendering.WordRendering property), 34  
 ReverseMLWorkflow (class in aisquared.config.analytic.ReverseMLWorkflow), 5  
 role (aisquared.config.postprocessing.Regression.Regression property), 13  
 S  
 secret (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic property), 3  
 secret (aisquared.config.analytic.DeployedModel.DeployedModel property), 3  
 secret (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow property), 5  
 share\_model\_with\_group() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient method), 42  
 share\_model\_with\_user() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient method), 42  
 SimpleFeedback (class in aisquared.config.feedback.SimpleFeedback), 8  
 size (aisquared.config.preprocessing.image.Steps.Resize property), 15  
 SOSRendering (class in aisquared.config.rendering.SOSRendering), 33  
 source (aisquared.config.rendering.FilterRendering.FilterRendering property), 28  
 split\_sentences (aisquared.config.preprocessing.text.Steps.Tokenize property), 21  
 split\_words (aisquared.config.preprocessing.text.Steps.Tokenize property), 21  
 stage (aisquared.config.GraphConfiguration.GraphConfiguration property), 36

stage (aisquared.config.ModelConfiguration.ModelConfiguration property), 37  
start\_character (aisquared.config.preprocessing.text.Steps.AddValue property), 19  
stds (aisquared.config.preprocessing.tabular.Steps.ZScore to\_dict() (aisquared.config.analytic.LocalModel.LocalModel method), 4  
step\_dict (aisquared.config.preprocessing.image.ImagePreprocessing.to\_dict() (aisquared.config.analytic.ReverseMLWorkflow.ReverseMLWorkflow method), 5  
step\_dict (aisquared.config.preprocessing.text.TextPreprocessing.to\_dict() (aisquared.config.feedback.BinaryFeedback.BinaryFeedback method), 6  
SubtractValue (class in to\_dict() (aisquared.config.feedback.ModelFeedback.ModelFeedback method), 7  
aisquared.config.preprocessing.image.Steps), to\_dict() (aisquared.config.feedback.MulticlassFeedback.MulticlassFeedback method), 7  
15  
T  
TableRendering (class in to\_dict() (aisquared.config.feedback.QualitativeFeedback.QualitativeFeedback method), 8  
aisquared.config.rendering.TableRendering), to\_dict() (aisquared.config.feedback.RegressionFeedback.RegressionFeedback method), 8  
33  
TabularPreprocessor (class in to\_dict() (aisquared.config.feedback.SimpleFeedback.SimpleFeedback method), 8  
aisquared.config.preprocessing.tabular.TabularPreprocessing), to\_dict() (aisquared.config.GraphConfiguration.GraphConfiguration method), 8  
18  
test\_connection() (aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient to\_dict() (aisquared.config.harvesting.ImageHarvester.ImageHarvester method), 9  
method), 43  
TextHarvester (class in to\_dict() (aisquared.config.harvesting.InputHarvester.InputHarvester method), 9  
aisquared.config.harvesting.TextHarvester), 10  
TextPreprocessor (class in to\_dict() (aisquared.config.harvesting.QueryParameterHarvester.QueryParameterHarvester method), 10  
aisquared.config.preprocessing.text.TextPreprocessing), 21  
thickness (aisquared.config.rendering.ImageRendering.ImageRendering to\_dict() (aisquared.config.harvesting.TextHarvester.TextHarvester method), 11  
property), 30  
thickness (aisquared.config.rendering.ObjectRendering.ObjectRendering to\_dict() (aisquared.config.ModelConfiguration.ModelConfiguration method), 37  
property), 31  
threshold (aisquared.config.postprocessing.BinaryClassification.BinaryClassification to\_dict() (aisquared.config.postprocessing.BinaryClassification.BinaryClassification method), 12  
property), 12  
threshold (aisquared.config.postprocessing.ObjectDetection.ObjectDetection to\_dict() (aisquared.config.postprocessing.MulticlassClassification.MulticlassClassification method), 12  
property), 12  
threshold\_key (aisquared.config.rendering.DocumentRendering.DocumentRendering to\_dict() (aisquared.config.postprocessing.ObjectDetection.ObjectDetection method), 12  
property), 26  
threshold\_key (aisquared.config.rendering.ImageRendering.ImageRendering to\_dict() (aisquared.config.postprocessing.Regression.Regression method), 13  
property), 30  
threshold\_key (aisquared.config.rendering.WordRendering.WordRendering to\_dict() (aisquared.config.preprocessing.image.ImagePreprocessing.ImagePreprocessing method), 14  
property), 34  
threshold\_value (aisquared.config.rendering.DocumentRendering.DocumentRendering to\_dict() (aisquared.config.preprocessing.image.Steps.AddValue method), 14  
property), 26  
threshold\_value (aisquared.config.rendering.ImageRendering.ImageRendering to\_dict() (aisquared.config.preprocessing.image.Steps.ConvertToColor method), 14  
property), 30  
threshold\_value (aisquared.config.rendering.WordRendering.WordRendering to\_dict() (aisquared.config.preprocessing.image.Steps.DivideValue method), 15  
property), 34  
to\_dict() (aisquared.base.BaseObject.BaseObject to\_dict() (aisquared.config.preprocessing.image.Steps.MultiplyValue method), 15  
method), 1  
to\_dict() (aisquared.base.CustomObject.CustomObject to\_dict() (aisquared.config.preprocessing.image.Steps.Resize method), 15  
method), 2  
to\_dict() (aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic to\_dict() (aisquared.config.preprocessing.image.Steps.SubtractValue method), 16  
method), 3



`to_dict()` (`aisquared.config.preprocessing.tabular.Steps.DropColumns` property), 43  
`to_dict()` (`aisquared.config.preprocessing.tabular.Steps.MimicPattern` property), 21  
`to_dict()` (`aisquared.config.preprocessing.tabular.Steps.OneHot` property), 20  
`to_dict()` (`aisquared.config.preprocessing.tabular.Steps.Trim` property), 21  
`to_dict()` (`aisquared.config.preprocessing.tabular.TabularPreprocessing.TabularPreprocessor` property), 20  
`to_dict()` (`aisquared.config.preprocessing.text.Steps.ConvertToCase` method), 19  
`to_dict()` (`aisquared.config.preprocessing.text.Steps.ConvertToUnicode` method), 20  
`to_dict()` (`aisquared.config.preprocessing.text.Steps.PadSequences` method), 20  
`to_dict()` (`aisquared.config.preprocessing.text.Steps.RemoveCharacter` method), 43  
`to_dict()` (`aisquared.config.preprocessing.text.Steps.Tokenize` method), 43  
`to_dict()` (`aisquared.config.preprocessing.text.Steps.Trim` method), 21  
`to_dict()` (`aisquared.config.preprocessing.text.TextPreprocessing.TextPreprocessor` property), 22  
`to_dict()` (`aisquared.config.rendering.BarChartRendering.BarChartRendering` property), 36  
`to_dict()` (`aisquared.config.rendering.ContainerRendering.ContainerRendering` property), 24  
`to_dict()` (`aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering` property), 25  
`to_dict()` (`aisquared.config.rendering.DocumentRendering.DocumentRendering` property), 26  
`to_dict()` (`aisquared.config.rendering.DoughnutChartRendering.DoughnutChartRendering` property), 27  
`to_dict()` (`aisquared.config.rendering.FilterRendering.FilterRendering` property), 14  
`to_dict()` (`aisquared.config.rendering.HTMLTagRendering.HTMLTagRendering` property), 15  
`to_dict()` (`aisquared.config.rendering.ImageRendering.ImageRendering` property), 15  
`to_dict()` (`aisquared.config.rendering.LineChartRendering.LineChartRendering` property), 16  
`to_dict()` (`aisquared.config.rendering.ObjectRendering.ObjectRendering` property), 28  
`to_dict()` (`aisquared.config.rendering.PieChartRendering.PieChartRendering` property), 17  
`to_dict()` (`aisquared.config.rendering.SOSRendering.SOSRendering` property), 36  
`to_dict()` (`aisquared.config.rendering.TableRendering.TableRendering` property), 37  
`to_dict()` (`aisquared.config.rendering.WordRendering.WordRendering` property), 20  
`to_json()` (`aisquared.base.BaseObject.BaseObject` method), 1  
`upload_model()` (`aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient` method), 43  
`url` (`aisquared.config.analytic.DeployedAnalytic.DeployedAnalytic` property), 3  
`url` (`aisquared.config.analytic.DeployedModel.DeployedModel` property), 3  
`url` (`aisquared.config.GraphConfiguration.GraphConfiguration` property), 36  
`url` (`aisquared.config.ModelConfiguration.ModelConfiguration` property), 37  
`url_locations` (`aisquared.config.harvesting.QueryParameterHarvester.QueryParameterHarvester` property), 3  
`username` (`aisquared.platform.AISquaredPlatformClient.AISquaredPlatformClient` property), 4  
`value` (`aisquared.config.preprocessing.image.Steps.AddValue` property), 14  
`value` (`aisquared.config.preprocessing.image.Steps.DivideValue` property), 15  
`value` (`aisquared.config.preprocessing.image.Steps.MultiplyValue` property), 15  
`value` (`aisquared.config.preprocessing.image.Steps.SubtractValue` property), 16  
`value` (`aisquared.config.rendering.FilterRendering.FilterRendering` property), 28  
`values` (`aisquared.config.preprocessing.tabular.Steps.OneHot` property), 17  
`version` (`aisquared.config.GraphConfiguration.GraphConfiguration` property), 36  
`version` (`aisquared.config.ModelConfiguration.ModelConfiguration` property), 37  
`vocabulary` (`aisquared.config.preprocessing.text.Steps.ConvertToVocabulary` property), 20  
`where_replace` (`aisquared.config.rendering.DashboardReplacementRendering.DashboardReplacementRendering` property), 25

*property*), [25](#)  
`width` (*aisquared.config.rendering.ContainerRendering.ContainerRendering*  
*property*), [24](#)  
`word_list` (*aisquared.config.rendering.WordRendering.WordRendering*  
*property*), [34](#)  
`WordRendering` (class in  
*aisquared.config.rendering.WordRendering*),  
[34](#)  
`words` (*aisquared.config.rendering.DocumentRendering.DocumentRendering*  
*property*), [26](#)

## X

`xOffset` (*aisquared.config.rendering.ContainerRendering.ContainerRendering*  
*property*), [24](#)

## Y

`yOffset` (*aisquared.config.rendering.ContainerRendering.ContainerRendering*  
*property*), [24](#)

## Z

`ZScore` (class in *aisquared.config.preprocessing.tabular.Steps*),  
[17](#)