Package: sagemaker.common (via r-universe)

July 18, 2024

Type Package

Title R6sagemaker lower level api calls

Version 0.2.0.9000

Description `R6sagemaker` lower level api calls.

Imports data.table, lgr, R6, fs, jsonlite, urltools, utils, zip, stats, sagemaker.debugger, sagemaker.core

Suggests mockthat, crayon, testthat (>= 3.0.0)

Remotes DyfanJones/sagemaker-r-debugger, DyfanJones/sagemaker-r-core

License Apache License (>= 2.0)

Encoding UTF-8

RoxygenNote 7.2.0

Collate 'r_utils.R' 'analytics.R' 'dataset_definition_inputs.R' 'network.R' 'job.R' 'processing.R' 'clarify.R' 'debugger.R' 'debugger_utils.R' 'debugger_profiler_constants.R' 'debugger_metrics_config.R' 'debugger_framework_profile.R' 'debugger_profiler_config.R' 'drift_check_baselines.R' 'lambda_helper.R' 'lineage_api_types.R' 'lineage_action.R' 'lineage_association.R' 'lineage_utils.R' 'lineage_artifact.R' 'lineage_context.R' 'lineage_visualizer.R' 'metadata_properties.R' 'serverless_serverless_inference_config.R' 'transformer.R' 'zzz.R'

Config/testthat/edition 3

Repository https://dyfanjones.r-universe.dev

RemoteUrl https://github.com/DyfanJones/sagemaker-r-common

RemoteRef HEAD

RemoteSha 50d2a8ee8a07ade4e76e74df8ee5fad0b59d74f4

Contents

`sagemaker.common-package` *sagemaker.common: this is just a placeholder*

Description

'R6sagemaker' lower level api calls.

Author(s)

Maintainer: Dyfan Jones <dyfan.r.jones@gmail.com>

Other contributors:

- Amazon.com, Inc. [copyright holder]

`AnalyticsMetricsBase` *AnalyticsMetricsBase Class*

Description

Base class for tuning job or training job analytics classes. Understands common functionality like persistence and caching.

Methods

Public methods:

- `AnalyticsMetricsBase$new()`
- `AnalyticsMetricsBase$export_csv()`
- `AnalyticsMetricsBase$dataframe()`
- `AnalyticsMetricsBase$clear_cache()`
- `AnalyticsMetricsBase$format()`
- `AnalyticsMetricsBase$clone()`

Method `new()`**:** Initialize a "AnalyticsMetricsBase" instance.

Usage:

```
AnalyticsMetricsBase$new()
```

Method `export_csv()`**:** ersists the analytics dataframe to a file.

Usage:

```
AnalyticsMetricsBase$export_csv(filename)
```

Arguments:

`filename` (str): The name of the file to save to.

Method `dataframe()`**:** A dataframe with lots of interesting results about this object. Created by calling SageMaker List and Describe APIs and converting them into a convenient tabular summary.

Usage:

```
AnalyticsMetricsBase$dataframe(force_refresh = FALSE)
```

Arguments:

`force_refresh` (bool): Set to True to fetch the latest data from SageMaker API.

Method `clear_cache()`**:** Clear the object of all local caches of API methods, so that the next time any properties are accessed they will be refreshed from the service.

Usage:

```
AnalyticsMetricsBase$clear_cache()
```

Method `format()`**:** format class

Usage:

```
AnalyticsMetricsBase$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
AnalyticsMetricsBase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`AthenaDatasetDefinition` *DatasetDefinition for Athena.*

Description

With this input, SQL queries will be executed using Athena to generate datasets to S3.

Super class

`sagemaker.core::ApiObject` -> `AthenaDatasetDefinition`

Methods

Public methods:

- AthenaDatasetDefinition$new()
- AthenaDatasetDefinition$clone()

Method `new()`**:** Initialize AthenaDatasetDefinition.

Usage:

```
AthenaDatasetDefinition$new(
  catalog = NULL,
  database = NULL,
  query_string = NULL,
  output_s3_uri = NULL,
  work_group = NULL,
  kms_key_id = NULL,
  output_format = NULL,
  output_compression = NULL
)
```

Arguments:

`catalog` (str, default=None): The name of the data catalog used in Athena query execution.

`database` (str, default=None): The name of the database used in the Athena query execution.

`query_string` (str, default=None): The SQL query statements, to be executed.

`output_s3_uri` (str, default=None): The location in Amazon S3 where Athena query results are stored.

`work_group` (str, default=None): The name of the workgroup in which the Athena query is being started.

`kms_key_id` (str, default=None): The AWS Key Management Service (AWS KMS) key that Amazon SageMaker uses to encrypt data generated from an Athena query execution.

`output_format` (str, default=None): The data storage format for Athena query results. Valid options are "PARQUET", "ORC", "AVRO", "JSON", "TEXTFILE"

`output_compression` (str, default=None): The compression used for Athena query results. Valid options are "GZIP", "SNAPPY", "ZLIB"

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
AthenaDatasetDefinition$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

BiasConfig *BiasConfig Class*

Description

Config object related to bias configurations of the input dataset.

Public fields

`analysis_config` Analysis config dictionary

Methods

Public methods:

- `BiasConfig$new()`
- `BiasConfig$get_config()`
- `BiasConfig$format()`
- `BiasConfig$clone()`

Method `new()`**:** Initializes a configuration of the sensitive groups in the dataset.

Usage:

```
BiasConfig$new(
  label_values_or_threshold,
  facet_name,
  facet_values_or_threshold = NULL,
  group_name = NULL
)
```

Arguments:

`label_values_or_threshold` (Any): List of label values or threshold to indicate positive outcome used for bias metrics.

`facet_name` (str): Sensitive attribute in the input data for which we like to compare metrics.

`facet_values_or_threshold` (list): Optional list of values to form a sensitive group or threshold for a numeric facet column that defines the lower bound of a sensitive group. Defaults to considering each possible value as sensitive group and computing metrics vs all the other examples.

`group_name` (str): Optional column name or index to indicate a group column to be used for the bias metric 'Conditional Demographic Disparity in Labels - CDDL' or 'Conditional Demographic Disparity in Predicted Labels - CDDPL'.

Method `get_config()`**:** Returns part of an analysis config dictionary.

Usage:

```
BiasConfig$get_config()
```

Method `format()`**:** format class

Usage:

```
BiasConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
BiasConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`CollectionConfig` *CollectionConfig Class*

Description

Creates tensor collections for SageMaker Debugger

Public fields

`name` The name of the collection configuration.

`parameters` The parameters for the collection configuration.

Methods

Public methods:

- `CollectionConfig$new()`
- `CollectionConfig$to_request_list()`
- `CollectionConfig$format()`
- `CollectionConfig$clone()`

Method `new()`**:** Constructor for collection configuration.

Usage:

```
CollectionConfig$new(name, parameters = NULL)
```

Arguments:

`name` (str): Required. The name of the collection configuration.

`parameters` (dict): Optional. The parameters for the collection configuration.

Method `to_request_list()`**:** Generate a request dictionary using the parameters initializing the object.

Usage:

```
CollectionConfig$to_request_list()
```

Returns: dict: A portion of an API request as a dictionary.

Method `format()`**:** format class

Usage:

```
CollectionConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
CollectionConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`cProfileTimer` *cProfileTimer enum environment list*

Description

List the possible cProfile timers for Python profiling.

Usage

```
cProfileTimer
```

Format

An object of class `cProfileTimer` (inherits from `Enum`, `environment`) of length 4.

Value

environment containing [TOTAL_TIME, CPU_TIME, OFF_CPU_TIME, DEFAULT]

`DataConfig` *DataConfig Class*

Description

Config object related to configurations of the input and output dataset.

Public fields

`s3_data_input_path` Dataset S3 prefix/object URI.

`s3_output_path` S3 prefix to store the output.

`s3_analysis_config_output_path` S3 prefix to store the analysis_config output.

`s3_data_distribution_type` Valid options are "FullyReplicated" or "ShardedByS3Key".

`s3_compression_type` Valid options are "None" or "Gzip".

`label` Target attribute of the model required by bias metrics

`headers` A list of column names in the input dataset.

`features` JSONPath for locating the feature columns

`analysis_config` Analysis config dictionary

Methods

Public methods:

- DataConfig$new()
- DataConfig$get_config()
- DataConfig$format()
- DataConfig$clone()

Method `new()`**:** Initializes a configuration of both input and output datasets.

Usage:

```
DataConfig$new(
  s3_data_input_path,
  s3_output_path,
  s3_analysis_config_output_path = NULL,
  label = NULL,
  headers = NULL,
  features = NULL,
 dataset_type = c("text/csv", "application/jsonlines", "application/x-parquet",
    "application/x-image"),
  s3_data_distribution_type = "FullyReplicated",
  s3_compression_type = c("None", "Gzip"),
  joinsource = NULL
)
```

Arguments:

`s3_data_input_path` (str): Dataset S3 prefix/object URI.

`s3_output_path` (str): S3 prefix to store the output.

`s3_analysis_config_output_path` (str): S3 prefix to store the analysis_config output If this field is None, then the s3_output_path will be used to store the analysis_config output

`label` (str): Target attribute of the model required by bias metrics (optional for SHAP) Specified as column name or index for CSV dataset, or as JSONPath for JSONLines.

`headers` (list[str]): A list of column names in the input dataset.

`features` (str): JSONPath for locating the feature columns for bias metrics if the dataset format is JSONLines.

`dataset_type` (str): Format of the dataset. Valid values are "text/csv" for CSV and "application/jsonlines" for JSONLines.

`s3_data_distribution_type` (str): Valid options are "FullyReplicated" or "ShardedByS3Key".

`s3_compression_type` (str): Valid options are "None" or "Gzip".

`joinsource` (str): The name or index of the column in the dataset that acts as an identifier column (for instance, while performing a join). This column is only used as an identifier, and not used for any other computations. This is an optional field in all cases except when the dataset contains more than one file, and ‘save_local_shap_values‘ is set to true in SHAPConfig.

Method `get_config()`**:** Returns part of an analysis config dictionary.

Usage:

```
DataConfig$get_config()
```

Method `format()`**:** format class

Usage:

```
DataConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DataConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DataloaderProfilingConfig`

DataloaderProfilingConfig Class

Description

The configuration for framework metrics to be collected for data loader profiling.

Super class

`sagemaker.common::MetricsConfigBase` -> `DataloaderProfilingConfig`

Methods

Public methods:

- `DataloaderProfilingConfig$new()`
- `DataloaderProfilingConfig$clone()`

Method `new()`**:** Specify target steps or a target duration to profile. By default, it profiles step 7 of training. If `profile_default_steps` is set to 'True' and none of the other range parameters is specified, the class uses the default config for dataloader profiling.

Usage:

```
DataloaderProfilingConfig$new(
  start_step = NULL,
  num_steps = NULL,
  start_unix_time = NULL,
  duration = NULL,
  profile_default_steps = FALSE,
  metrics_regex = ".*"
)
```

Arguments:

`start_step` (int): The step to start profiling. The default is step 7.

`num_steps` (int): The number of steps to profile. The default is for 1 step.

`start_unix_time` (int): The Unix time to start profiling. The default is for 1 step.

`duration` (float): The duration in seconds to profile.

`profile_default_steps` (bool): Indicates whether the default config should be used.

`metrics_regex` (str): Regex pattern

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DataloaderProfilingConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DatasetDefinition` *DatasetDefinition input.*

Description

DatasetDefinition input.

DatasetDefinition input.

Super class

`sagemaker.core::ApiObject` -> `DatasetDefinition`

Methods

Public methods:

- `DatasetDefinition$new()`
- `DatasetDefinition$clone()`

Method `new()`**:** Initialize DatasetDefinition.

Usage:

```
DatasetDefinition$new(
  data_distribution_type = "ShardedByS3Key",
  input_mode = "File",
  local_path = NULL,
  redshift_dataset_definition = NULL,
  athena_dataset_definition = NULL
)
```

Arguments:

`data_distribution_type` (str, default="ShardedByS3Key"): Whether the generated dataset is FullyReplicated or ShardedByS3Key (default).

`input_mode` (str, default="File"): Whether to use File or Pipe input mode. In File (default) mode, Amazon SageMaker copies the data from the input source onto the local Amazon Elastic Block Store (Amazon EBS) volumes before starting your training algorithm. This is the most commonly used input mode. In Pipe mode, Amazon SageMaker streams input data from the source directly to your algorithm without using the EBS volume.

`local_path` (str, default=None): The local path where you want Amazon SageMaker to download the Dataset Definition inputs to run a processing job. LocalPath is an absolute path to the input data. This is a required parameter when 'AppManaged' is False (default).

`redshift_dataset_definition` (:class:`sagemaker.common::RedshiftDatasetDefinition`, default=None): Configuration for Redshift Dataset Definition input.

`athena_dataset_definition` (:class:`sagemaker.common::AthenaDatasetDefinition`, default=None): Configuration for Athena Dataset Definition input.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DatasetDefinition$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DebuggerHookConfig` *Create a Debugger hook configuration object to save the tensor for debugging.*

Description

`DebuggerHookConfig` provides options to customize how debugging information is emitted and saved. This high-level `DebuggerHookConfig` class runs based on the 'smdebug.SaveConfig https://github.com/awslabs/sagemaker-debugger/blob/master/docs/api.md#saveconfig class.

Public fields

`s3_output_path` The location in Amazon S3 to store the output tensors

`container_local_output_path` The local path in the container

`hook_parameters` A dictionary of parameters

`collection_configs` A list of :class:'~sagemaker.debugger.CollectionConfig

Methods

Public methods:

- `DebuggerHookConfig$new()`
- `DebuggerHookConfig$to_request_list()`
- `DebuggerHookConfig$format()`
- `DebuggerHookConfig$clone()`

Method `new()`**:** Initialize the DebuggerHookConfig instance.

Usage:

```
DebuggerHookConfig$new(
  s3_output_path = NULL,
  container_local_output_path = NULL,
  hook_parameters = NULL,
  collection_configs = NULL
)
```

Arguments:

`s3_output_path` (str): Optional. The location in Amazon S3 to store the output tensors. The default Debugger output path is created under the default output path of the :class:'~sagemaker.estimator.Estimator' class. For example, s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/debug-output/.

`container_local_output_path` (str): Optional. The local path in the container.

`hook_parameters` (dict): Optional. A dictionary of parameters.

`collection_configs` ([sagemaker.debugger.CollectionConfig]): Required. A list of :class:'~sagemaker.debugger.Colle objects to be saved at the `s3_output_path`.

Method `to_request_list()`**:** Generate a request dictionary using the parameters when initializing the object.

Usage:

```
DebuggerHookConfig$to_request_list()
```

Returns: dict: An portion of an API request as a dictionary.

Method `format()`**:** format class

Usage:

```
DebuggerHookConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DebuggerHookConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DetailedProfilingConfig` *DetailedProfilingConfig Class*

Description

The configuration for framework metrics to be collected for detailed profiling.

Super class

`sagemaker.common::MetricsConfigBase` -> `DetailedProfilingConfig`

Methods

Public methods:

- DetailedProfilingConfig$new()
- DetailedProfilingConfig$clone()

Method `new()`**:** Specify target steps or a target duration to profile. By default, it profiles step 5 of training. If `profile_default_steps` is set to 'True' and none of the other range parameters is specified, the class uses the default configuration for detailed profiling.

Usage:

```
DetailedProfilingConfig$new(
  start_step = NULL,
  num_steps = NULL,
  start_unix_time = NULL,
  duration = NULL,
  profile_default_steps = FALSE
)
```

Arguments:

`start_step` (int): The step to start profiling. The default is step 5.

`num_steps` (int): The number of steps to profile. The default is for 1 step.

`start_unix_time` (int): The Unix time to start profiling.

`duration` (float): The duration in seconds to profile.

`profile_default_steps` (bool): Indicates whether the default config should be used.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DetailedProfilingConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`DriftCheckBaselines` *DriftCheckBaselines class*

Description

Accepts drift check baselines parameters for conversion to request dict.

Methods

Public methods:

- DriftCheckBaselines$new()
- DriftCheckBaselines$to_request_list()
- DriftCheckBaselines$clone()

Method `new()`**:** Initialize a "DriftCheckBaselines" instance and turn parameters into dict.

Usage:

```
DriftCheckBaselines$new(
  model_statistics = NULL,
  model_constraints = NULL,
  model_data_statistics = NULL,
  model_data_constraints = NULL,
  bias_config_file = NULL,
  bias_pre_training_constraints = NULL,
  bias_post_training_constraints = NULL,
  explainability_constraints = NULL,
  explainability_config_file = NULL
)
```

Arguments:

`model_statistics` (MetricsSource): A metric source object that represents

`model_constraints` (MetricsSource): A metric source object that represents

`model_data_statistics` (MetricsSource): A metric source object that represents

`model_data_constraints` (MetricsSource): A metric source object that represents

`bias_config_file` (FileSource): A file source object that represents bias config

`bias_pre_training_constraints` (MetricsSource):

`bias_post_training_constraints` (MetricsSource):

`explainability_constraints` (MetricsSource):

`explainability_config_file` (FileSource): A file source object that represents

Method `to_request_list()`**:** Generates a request dictionary using the parameters provided to the class.

Usage:

```
DriftCheckBaselines$to_request_list()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DriftCheckBaselines$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ExperimentAnalytics` *ExperimentAnalytics class*

Description

Fetch trial component data and make them accessible for analytics.

Super class

`sagemaker.common::AnalyticsMetricsBase` -> `ExperimentAnalytics`

Public fields

`MAX_TRIAL_COMPONENTS` class metadata

Active bindings

`name` Name of the Experiment being analyzed

Methods

Public methods:

- `ExperimentAnalytics$new()`
- `ExperimentAnalytics$clear_cache()`
- `ExperimentAnalytics$clone()`

Method `new()`**:** Initialize a "ExperimentAnalytics" instance.

Usage:

```
ExperimentAnalytics$new(
  experiment_name = NULL,
  search_expression = NULL,
  sort_by = NULL,
  sort_order = NULL,
  metric_names = NULL,
  parameter_names = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`experiment_name` (str, optional): Name of the experiment if you want to constrain the search to only trial components belonging to an experiment.

`search_expression` (dict, optional): The search query to find the set of trial components to use to populate the data frame.

`sort_by` (str, optional): The name of the resource property used to sort the set of trial components.

`sort_order` (str optional): How trial components are ordered, valid values are Ascending and Descending. The default is Descending.

`metric_names` (list, optional): string names of all the metrics to be shown in the data frame. If not specified, all metrics will be shown of all trials.

`parameter_names` (list, optional): string names of the parameters to be shown in the data frame. If not specified, all parameters will be shown of all trials.

`sagemaker_session` (sagemaker.session.Session): Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, one is created using the default AWS configuration chain.

Method `clear_cache()`**:** Clear the object of all local caches of API methods.

Usage:

```
ExperimentAnalytics$clear_cache()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ExperimentAnalytics$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ExplainabilityConfig` *ExplainabilityConfig Class*

Description

Abstract config class to configure an explainability method.

Methods

Public methods:

- `ExplainabilityConfig$get_explainability_config()`
- `ExplainabilityConfig$format()`
- `ExplainabilityConfig$clone()`

Method `get_explainability_config()`**:** Returns config.

Usage:

```
ExplainabilityConfig$get_explainability_config()
```

Method `format()`**:** format class

Usage:

```
ExplainabilityConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ExplainabilityConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

FeatureStoreOutput *Amazon SageMaker Feature Store class*

Description

Configuration for processing job outputs in Amazon SageMaker Feature Store

Super class

`sagemaker.core::ApiObject` -> `FeatureStoreOutput`

Public fields

`feature_group_name` placeholder

Methods

Public methods:

- `FeatureStoreOutput$clone()`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FeatureStoreOutput$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

FrameworkProcessor *FrameworkProcessor class*

Description

Handles Amazon SageMaker processing tasks for jobs using a machine learning framework

Super classes

`sagemaker.common::Processor` -> `sagemaker.common::ScriptProcessor` -> `FrameworkProcessor`

Public fields

`framework_entrypoint_command`

Methods

Public methods:

- FrameworkProcessor$new()
- FrameworkProcessor$get_run_args()
- FrameworkProcessor$run()
- FrameworkProcessor$clone()

Method `new()`**:** Initializes a "FrameworkProcessor" instance. The "FrameworkProcessor" handles Amazon SageMaker Processing tasks for jobs using a machine learning framework, which allows for a set of Python scripts to be run as part of the Processing Job.

Usage:

```
FrameworkProcessor$new(
  estimator_cls,
  framework_version,
  role,
  instance_count,
  instance_type,
  py_version = "py3",
  image_uri = NULL,
  command = NULL,
  volume_size_in_gb = 30,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  code_location = NULL,
  max_runtime_in_seconds = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  env = NULL,
  tags = NULL,
  network_config = NULL
)
```

Arguments:

`estimator_cls` (type): A subclass of the :class:'~sagemaker.estimator.Framework' estimator

`framework_version` (str): The version of the framework. Value is ignored when "image_uri" is provided.

`role` (str): An AWS IAM role name or ARN. Amazon SageMaker Processing uses this role to access AWS resources, such as data stored in Amazon S3.

`instance_count` (int): The number of instances to run a processing job with.

`instance_type` (str): The type of EC2 instance to use for processing, for example, 'ml.c4.xlarge'.

`py_version` (str): Python version you want to use for executing your model training code. One of 'py2' or 'py3'. Defaults to 'py3'. Value is ignored when "image_uri" is provided.

`image_uri` (str): The URI of the Docker image to use for the processing jobs (default: None).

`command` ([str]): The command to run, along with any command-line flags to *precede* the "'code script"'. Example: ["python3", "-v"]. If not provided, ["python"] will be chosen (default: None).

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data during processing (default: 30).

`volume_kms_key` (str): A KMS key for the processing volume (default: None).

`output_kms_key` (str): The KMS key ID for processing job outputs (default: None).

`code_location` (str): The S3 prefix URI where custom code will be uploaded (default: None). The code file uploaded to S3 is 'code_location/job-name/source/sourcedir.tar.gz'. If not specified, the default "code location" is 's3://sagemaker-default-bucket'

`max_runtime_in_seconds` (int): Timeout in seconds (default: None). After this amount of time, Amazon SageMaker terminates the job, regardless of its current status. If 'max_runtime_in_seconds' is not specified, the default value is 24 hours.

`base_job_name` (str): Prefix for processing name. If not specified, the processor generates a default job name, based on the processing image name and current timestamp (default: None).

`sagemaker_session` (:class:'~sagemaker.session.Session'): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain (default: None).

`env` (dict[str, str]): Environment variables to be passed to the processing jobs (default: None).

`tags` (list[dict]): List of tags to be passed to the processing job (default: None). For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

`network_config` (:class:'~sagemaker.network.NetworkConfig'): A :class:'~sagemaker.network.NetworkConfig' object that configures network isolation, encryption of inter-container traffic, security group IDs, and subnets (default: None).

Method `get_run_args()`**:** This object contains the normalized inputs, outputs and arguments needed when using a "FrameworkProcessor" in a :class:'~sagemaker.workflow.steps.ProcessingStep'.

Usage:

```
FrameworkProcessor$get_run_args(
  code,
  source_dir = NULL,
  dependencies = NULL,
  git_config = NULL,
  inputs = NULL,
  outputs = NULL,
  arguments = NULL,
  job_name = NULL
)
```

Arguments:

`code` (str): This can be an S3 URI or a local path to a file with the framework script to run. See the "code" argument in 'sagemaker.processing.FrameworkProcessor.run()'.

`source_dir` (str): Path (absolute, relative, or an S3 URI) to a directory wit any other processing source code dependencies aside from the entrypoint file (default: None). See the "source_dir" argument in 'sagemaker.processing.FrameworkProcessor.run()'

`dependencies` (list[str]): A list of paths to directories (absolute or relative) with any additional libraries that will be exported to the container (default: []). See the "dependencies" argument in 'sagemaker.processing.FrameworkProcessor.run()'.

`git_config` (dict[str, str]): Git configurations used for cloning files. See the 'git_config' argument in 'sagemaker.processing.FrameworkProcessor.run()'.

`inputs` (list[:class:'~sagemaker.processing.ProcessingInput']): Input files for the processing job. These must be provided as :class:'~sagemaker.processing.ProcessingInput' objects (default: None).

`outputs` (list[:class:'~sagemaker.processing.ProcessingOutput']): Outputs for the processing job. These can be specified as either path strings or :class:'~sagemaker.processing.ProcessingOutput' objects (default: None).

`arguments` (list[str]): A list of string arguments to be passed to a processing job (default: None).

`job_name` (str): Processing job name. If not specified, the processor generates a default job name, based on the base job name and current timestamp.

Returns: Returns a RunArgs object.

Method `run()`**:** Runs a processing job.

Usage:

```
FrameworkProcessor$run(
  code,
  source_dir = NULL,
  dependencies = NULL,
  git_config = NULL,
  inputs = NULL,
  outputs = NULL,
  arguments = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL,
  experiment_config = NULL,
  kms_key = NULL
)
```

Arguments:

`code` (str): This can be an S3 URI or a local path to a file with the framework script to run.Path (absolute or relative) to the local Python source file which should be executed as the entry point to training. When 'code' is an S3 URI, ignore 'source_dir', 'dependencies, and 'git_config'. If "source_dir" is specified, then "code" must point to a file located at the root of "source_dir".

`source_dir` (str): Path (absolute, relative or an S3 URI) to a directory with any other processing source code dependencies aside from the entry point file (default: None). If "source_dir" is an S3 URI, it must point to a tar.gz file. Structure within this directory are preserved when processing on Amazon SageMaker (default: None).

`dependencies` (list[str]): A list of paths to directories (absolute or relative) with any additional libraries that will be exported to the container (default: []). The library folders will be copied to SageMaker in the same folder where the entrypoint is copied. If 'git_config' is provided, 'dependencies' should be a list of relative locations to directories with any additional libraries needed in the Git repo (default: None).

`git_config` (dict[str, str]): Git configurations used for cloning files, including “repo“, “branch“, “commit“, “2FA_enabled“, “username“, “password“ and “token“. The “repo“ field is required. All other fields are optional. “repo“ specifies the Git repository where your training script is stored. If you don’t provide “branch“, the default value ’master’ is used. If you don’t provide “commit“, the latest commit in the specified branch is used. results in cloning the repo specified in ’repo’, then checkout the ’master’ branch, and checkout the specified commit. “2FA_enabled“, “username“, “password“ and “token“ are used for authentication. For GitHub (or other Git) accounts, set “2FA_enabled“ to ’True’ if two-factor authentication is enabled for the account, otherwise set it to ’False’. If you do not provide a value for “2FA_enabled“, a default value of ’False’ is used. CodeCommit does not support two-factor authentication, so do not provide "2FA_enabled" with CodeCommit repositories. For GitHub and other Git repos, when SSH URLs are provided, it doesn’t matter whether 2FA is enabled or disabled; you should either have no passphrase for the SSH key pairs, or have the ssh-agent configured so that you will not be prompted for SSH passphrase when you do ’git clone’ command with SSH URLs. When HTTPS URLs are provided: if 2FA is disabled, then either token or username+password will be used for authentication if provided (token prioritized); if 2FA is enabled, only token will be used for authentication if provided. If required authentication info is not provided, python SDK will try to use local credentials storage to authenticate. If that fails either, an error message will be thrown. For CodeCommit repos, 2FA is not supported, so ’2FA_enabled’ should not be provided. There is no token in CodeCommit, so ’token’ should not be provided too. When ’repo’ is an SSH URL, the requirements are the same as GitHub-like repos. When ’repo’ is an HTTPS URL, username+password will be used for authentication if they are provided; otherwise, python SDK will try to use either CodeCommit credential helper or local credential storage for authentication.

`inputs` (list[:class:‘~sagemaker.processing.ProcessingInput‘]): Input files for the processing job. These must be provided as :class:‘~sagemaker.processing.ProcessingInput‘ objects (default: None).

`outputs` (list[:class:‘~sagemaker.processing.ProcessingOutput‘]): Outputs for the processing job. These can be specified as either path strings or :class:‘~sagemaker.processing.ProcessingOutput‘ objects (default: None).

`arguments` (list[str]): A list of string arguments to be passed to a processing job (default: None).

`wait` (bool): Whether the call should wait until the job completes (default: True).

`logs` (bool): Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).

`job_name` (str): Processing job name. If not specified, the processor generates a default job name, based on the base job name and current timestamp.

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: ’ExperimentName’, ’TrialName’, and ’TrialComponentDisplayName’.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the user code file (default: None).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FrameworkProcessor$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`FrameworkProfile` *Sets up the profiling configuration for framework metrics.*

Description

Validates user inputs and fills in default values if no input is provided. There are three main profiling options to choose from: :class:‘~sagemaker.debugger.metrics_config.DetailedProfilingConfig‘, :class:‘~sagemaker.debugger.metrics_config.DataloaderProfilingConfig‘, and :class:‘~sagemaker.debugger.metrics_config.Py
The following list shows available scenarios of configuring the profiling options.

1. None of the profiling configuration, step range, or time range is specified. SageMaker Debugger activates framework profiling based on the default settings of each profiling option.
2. Target step or time range is specified to this :class:‘~sagemaker.debugger.metrics_config.FrameworkProfile‘ class. The requested target step or time range setting propagates to all of the framework profiling options. For example, if you configure this class as following, all of the profiling options profiles the 6th step.
3. Individual profiling configurations are specified through the “*_profiling_config“ parameters. SageMaker Debugger profiles framework metrics only for the specified profiling configurations. For example, if the :class:‘~sagemaker.debugger.metrics_config.DetailedProfilingConfig‘ class is configured but not the other profiling options, Debugger only profiles based on the settings specified to the :class:‘~sagemaker.debugger.metrics_config.DetailedProfilingConfig‘ class. For example, the following example shows a profiling configuration to perform detailed profiling at step 10, data loader profiling at step 9 and 10, and Python profiling at step 12. If the individual profiling configurations are specified in addition to the step or time range, SageMaker Debugger prioritizes the individual profiling configurations and ignores the step or time range. For example, in the following code, the “start_step=1“ and “num_steps=10“ will be ignored.

Methods

Public methods:

- `FrameworkProfile$new()`
- `FrameworkProfile$format()`
- `FrameworkProfile$clone()`

Method `new()`**:** Initialize the FrameworkProfile class object.

Usage:

```
FrameworkProfile$new(
  local_path = BASE_FOLDER_DEFAULT,
  file_max_size = MAX_FILE_SIZE_DEFAULT,
  file_close_interval = CLOSE_FILE_INTERVAL_DEFAULT,
  file_open_fail_threshold = FILE_OPEN_FAIL_THRESHOLD_DEFAULT,
```

```
  detailed_profiling_config = NULL,
  dataloader_profiling_config = NULL,
  python_profiling_config = NULL,
  horovod_profiling_config = NULL,
  smdataparallel_profiling_config = NULL,
  start_step = NULL,
  num_steps = NULL,
  start_unix_time = NULL,
  duration = NULL
)
```

Arguments:

`local_path` (str):

`file_max_size` (int):

`file_close_interval` (int):

`file_open_fail_threshold` (int):

`detailed_profiling_config` (DetailedProfilingConfig): The configuration for detailed profiling. Configure it using the :class:‘~sagemaker.debugger.metrics_config.DetailedProfilingConfig‘ class. Pass “DetailedProfilingConfig()“ to use the default configuration.

`dataloader_profiling_config` (DataloaderProfilingConfig): The configuration for dataloader metrics profiling. Configure it using the :class:‘~sagemaker.debugger.metrics_config.DataloaderProfilingConfig‘ class. Pass “DataloaderProfilingConfig()“ to use the default configuration.

`python_profiling_config` (PythonProfilingConfig): The configuration for stats collected by the Python profiler (cProfile or Pyinstrument). Configure it using the :class:‘~sagemaker.debugger.metrics_config.Pyt class. Pass “PythonProfilingConfig()“ to use the default configuration.

`horovod_profiling_config` :

`smdataparallel_profiling_config` :

`start_step` (int): The step at which to start profiling.

`num_steps` (int): The number of steps to profile.

`start_unix_time` (int): The Unix time at which to start profiling.

`duration` (float): The duration in seconds to profile.

Method `format()`**:** format class

Usage:

```
FrameworkProfile$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
FrameworkProfile$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
library(sagemaker.common)
profiler_config=ProfilerConfig$new(
```

```
    framework_profile_params=FrameworkProfile$new(
    start_step=1,
    num_steps=10,
   detailed_profiling_config=DetailedProfilingConfig$new(start_step=10, num_steps=1),
  dataloader_profiling_config=DataloaderProfilingConfig$new(start_step=9, num_steps=2),
    python_profiling_config=PythonProfilingConfig$new(start_step=12, num_steps=1)
  )
)
```

`get_default_profiler_rule` *Return the default built-in profiler rule with a unique name.*

Description

Return the default built-in profiler rule with a unique name.

Usage

```
get_default_profiler_rule()
```

Value

ProfilerRule: The instance of the built-in ProfilerRule.

`get_rule_container_image_uri` *Return the Debugger rule image URI for the given AWS Region.*

Description

For a full list of rule image URIs, see 'Use Debugger Docker Images for Built-in or Custom Rules https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-docker-images-rules.html.

Usage

```
get_rule_container_image_uri(region)
```

Arguments

`region`	(str): A string of AWS Region. For example, "'us-east-1'".

Value

str : Formatted image URI for the given AWS Region and the rule container type.

HorovodProfilingConfig *HorovodProfilingConfig Class*

Description

The configuration for framework metrics from Horovod distributed training.

Super class

`sagemaker.common::MetricsConfigBase` -> `HorovodProfilingConfig`

Methods

Public methods:

- `HorovodProfilingConfig$new()`
- `HorovodProfilingConfig$clone()`

Method `new()`**:** Specify target steps or a target duration to profile. By default, it profiles step 13 of training. If `profile_default_steps` is set to 'True' and none of the other range parameters is specified, the class uses the default config for horovod profiling.

Usage:

```
HorovodProfilingConfig$new(
  start_step = NULL,
  num_steps = NULL,
  start_unix_time = NULL,
  duration = NULL,
  profile_default_steps = FALSE
)
```

Arguments:

`start_step` (int): The step to start profiling. The default is step 13.

`num_steps` (int): The number of steps to profile. The default is for 1 steps.

`start_unix_time` (int): The Unix time to start profiling.

`duration` (float): The duration in seconds to profile.

`profile_default_steps` (bool): Indicates whether the default config should be used.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HorovodProfilingConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

HyperparameterTuningJobAnalytics
HyperparameterTuningJobAnalytics Class

Description

Fetch results about a hyperparameter tuning job and make them accessible for analytics.

Super class

`sagemaker.common::AnalyticsMetricsBase` -> `HyperparameterTuningJobAnalytics`

Active bindings

`name` Name of the HyperparameterTuningJob being analyzed

`tuning_ranges` A dictionary describing the ranges of all tuned hyperparameters. The keys are the names of the hyperparameter, and the values are the ranges. The output can take one of two forms: * If the 'TrainingJobDefinition' field is present in the job description, the output is a dictionary constructed from 'ParameterRanges' in 'HyperParameterTuningJobConfig' of the job description. The keys are the parameter names, while the values are the parameter ranges. Example: »> »> "eta": "MaxValue": "1", "MinValue": "0", "Name": "eta", »> "gamma": "MaxValue": "10", "MinValue": "0", "Name": "gamma", »> "iterations": "MaxValue": "100", "MinValue": "50", "Name": "iterations", »> "num_layers": "MaxValue": "30", "MinValue": "5", "Name": "num_layers", »> * If the 'TrainingJobDefinitions' field (list) is present in the job description, the output is a dictionary with keys as the 'DefinitionName' values from all items in 'TrainingJobDefinitions', and each value would be a dictionary constructed from 'HyperParameterRanges' in each item in 'TrainingJobDefinitions' in the same format as above Example: »> »> "estimator_1": »> "eta": "MaxValue": "1", "MinValue": "0", "Name": "eta", »> "gamma": "MaxValue": "10", "MinValue": "0", "Name": "gamma", »> , »> "estimator_2": »> "framework": "Values": ["TF", "MXNet"], "Name": "framework", »> "gamma": "MaxValue": "1.0", "MinValue": "0.2", "Name": "gamma" »> »> For more details about the 'TrainingJobDefinition' and 'TrainingJobDefinitions' fields in job description, see https://botocore.readthedocs.io/en/latest/reference/services/sagemaker.html#SageMaker.Client.create_hyper_parameter

Methods

Public methods:

- `HyperparameterTuningJobAnalytics$new()`
- `HyperparameterTuningJobAnalytics$description()`
- `HyperparameterTuningJobAnalytics$training_job_summaries()`
- `HyperparameterTuningJobAnalytics$clear_cache()`
- `HyperparameterTuningJobAnalytics$clone()`

Method `new()`**:** Initialize a "HyperparameterTuningJobAnalytics" instance.

Usage:

```
HyperparameterTuningJobAnalytics$new(
  hyperparameter_tuning_job_name,
  sagemaker_session = NULL
)
```

Arguments:

`hyperparameter_tuning_job_name` (str): name of the HyperparameterTuningJob to analyze.

`sagemaker_session` (sagemaker.session.Session): Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, one is created using the default AWS configuration chain.

Method `description()`**:** Call "DescribeHyperParameterTuningJob" for the hyperparameter tuning job.

Usage:

```
HyperparameterTuningJobAnalytics$description(force_refresh = FALSE)
```

Arguments:

`force_refresh` (bool): Set to True to fetch the latest data from SageMaker API.

Returns: dict: The Amazon SageMaker response for "DescribeHyperParameterTuningJob".

Method `training_job_summaries()`**:** A (paginated) list of everything from "ListTrainingJobsForTuningJob".

Usage:

```
HyperparameterTuningJobAnalytics$training_job_summaries(force_refresh = FALSE)
```

Arguments:

`force_refresh` (bool): Set to True to fetch the latest data from SageMaker API.

Returns: dict: The Amazon SageMaker response for "ListTrainingJobsForTuningJob".

Method `clear_cache()`**:** Clear the object of all local caches of API methods.

Usage:

```
HyperparameterTuningJobAnalytics$clear_cache()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
HyperparameterTuningJobAnalytics$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

ImageConfig *Config object for handling images*

Description

Config object for handling images

Config object for handling images

Super class

sagemaker.common::ExplainabilityConfig -> ImageConfig

Public fields

image_config Image config

Methods

Public methods:

- ImageConfig$new()
- ImageConfig$get_image_config()
- ImageConfig$clone()

Method new(): Initializes all configuration parameters needed for SHAP CV explainability

Usage:

```
ImageConfig$new(
  model_type,
  num_segments = NULL,
  feature_extraction_method = NULL,
  segment_compactness = NULL,
  max_objects = NULL,
  iou_threshold = NULL,
  context = NULL
)
```

Arguments:

model_type (str): Specifies the type of CV model. Options: (IMAGE_CLASSIFICATION | OBJECT_DETECTION).

num_segments (None or int): Clarify uses SKLearn's SLIC method for image segmentation to generate features/superpixels. num_segments specifies approximate number of segments to be generated. Default is None. SLIC will default to 100 segments.

feature_extraction_method (NULL or str): method used for extracting features from the image.ex. "segmentation". Default is segmentation.

segment_compactness (NULL or float): Balances color proximity and space proximity. Higher values give more weight to space proximity, making superpixel shapes more square/cubic. We recommend exploring possible values on a log scale, e.g., 0.01, 0.1, 1, 10, 100, before refining around a chosen value.

`max_objects` (NULL or int): maximum number of objects displayed. Object detection algorithm may detect more than max_objects number of objects in a single image. The top max_objects number of objects according to confidence score will be displayed.

`iou_threshold` (NULL or float): minimum intersection over union for the object bounding box to consider its confidence score for computing SHAP values [0.0, 1.0]. This parameter is used for the object detection case.

`context` (NULL or float): refers to the portion of the image outside of the bounding box. Scale is [0.0, 1.0]. If set to 1.0, whole image is considered, if set to 0.0 only the image inside bounding box is considered.

Method `get_image_config()`**:** Returns the image config part of an analysis config dictionary.

Usage:

```
ImageConfig$get_image_config()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ImageConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`is_valid_regex` *Helper function to determine whether the provided regex is valid.*

Description

Helper function to determine whether the provided regex is valid.

Usage

```
is_valid_regex(regex)
```

Arguments

`regex`	(str): The user provided regex.

Value

bool: Indicates whether the provided regex was valid or not.

Lambda *Lambda class*

Description

Contains lambda boto3 wrappers to Create, Update, Delete and Invoke Lambda functions.

Methods

Public methods:

- Lambda$new()
- Lambda$create()
- Lambda$update()
- Lambda$invoke()
- Lambda$delete()
- Lambda$format()
- Lambda$clone()

Method `new()`**:** Constructs a Lambda instance. This instance represents a Lambda function and provides methods for updating, deleting and invoking the function. This class can be used either for creating a new Lambda function or using an existing one. When using an existing Lambda function, only the function_arn argument is required. When creating a new one the function_name, execution_role_arn and handler arguments are required, as well as either script or zipped_code_dir.

Usage:

```
Lambda$new(
  function_arn = NULL,
  function_name = NULL,
  execution_role_arn = NULL,
  zipped_code_dir = NULL,
  s3_bucket = NULL,
  script = NULL,
  handler = NULL,
  session = NULL,
  timeout = 120,
  memory_size = 128,
  runtime = "python3.8"
)
```

Arguments:

`function_arn` (str): The arn of the Lambda function.

`function_name` (str): The name of the Lambda function. Function name must be provided to create a Lambda function.

`execution_role_arn` (str): The role to be attached to Lambda function.

`zipped_code_dir` (str): The path of the zipped code package of the Lambda function.

`s3_bucket` (str): The bucket where zipped code is uploaded. If not provided, default session bucket is used to upload zipped_code_dir.

`script` (str): The path of Lambda function script for direct zipped upload

`handler` (str): The Lambda handler. The format for handler should be file_name.function_name. For ex: if the name of the Lambda script is hello_world.py and Lambda function definition in that script is

`session` (sagemaker.session.Session): Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, new session is created.

`timeout` (int): Timeout of the Lambda function in seconds. Default is 120 seconds.

`memory_size` (int): Memory of the Lambda function in megabytes. Default is 128 MB.

`runtime` (str): Runtime of the Lambda function. Default is set to python3.8.

`lambda_handler(event, context)`, the handler should be hello_world.lambda_handler

Method `create()`**:** Method to create a lambda function.

Usage:

```
Lambda$create()
```

Returns: boto3 response from Lambda's create_function method.

Method `update()`**:** Method to update a lambda function.

Usage:

```
Lambda$update()
```

Returns: : paws response from Lambda's update_function method.

Method `invoke()`**:** Method to invoke a lambda function.

Usage:

```
Lambda$invoke()
```

Returns: paws response from Lambda's invoke method.

Method `delete()`**:** Method to delete a lambda function.

Usage:

```
Lambda$delete()
```

Returns: paws response from Lambda's delete_function method.

Method `format()`**:** foramt class

Usage:

```
Lambda$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Lambda$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`MetricsConfigBase` *The base class for the metrics configuration.*

Description

It determines the step or time range that needs to be profiled and validates the input value pairs. Available profiling range parameter pairs are (`start_step` and `num_steps`) and (`start_unix_time` and `duration`). The two parameter pairs are mutually exclusive, and this class validates if one of the two pairs is used. If both pairs are specified, a FOUND_BOTH_STEP_AND_TIME_FIELDS error occurs.

Methods

Public methods:

- `MetricsConfigBase$new()`
- `MetricsConfigBase$to_json_string()`
- `MetricsConfigBase$format()`
- `MetricsConfigBase$clone()`

Method `new()`**:** Validate the provided range fields and set the range to be profiled accordingly.

Usage:

```
MetricsConfigBase$new(
  name,
  start_step = NULL,
  num_steps = NULL,
  start_unix_time = NULL,
  duration = NULL
)
```

Arguments:

`name` (str): The name of the metrics config.

`start_step` (int): The step to start profiling.

`num_steps` (int): The number of steps to profile.

`start_unix_time` (int): The Unix time to start profiling.

`duration` (float): The duration in seconds to profile.

Method `to_json_string()`**:** Convert this metrics configuration to dictionary formatted as a string. Calling eval on the return value is the same as calling _to_json directly.

Usage:

```
MetricsConfigBase$to_json_string()
```

Returns: str: This metrics configuration as a dictionary and formatted as a string.

Method `format()`**:** format class

Usage:

```
MetricsConfigBase$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
MetricsConfigBase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ModelConfig` *Model Config*

Description

Config object related to a model and its endpoint to be created.

Public fields

`predictor_config` Predictor dictionary of the analysis config

Methods

Public methods:

- `ModelConfig$new()`
- `ModelConfig$get_predictor_config()`
- `ModelConfig$format()`
- `ModelConfig$clone()`

Method `new()`**:** Initializes a configuration of a model and the endpoint to be created for it.

Usage:

```
ModelConfig$new(
  model_name,
  instance_count,
  instance_type,
  accept_type = NULL,
  content_type = NULL,
  content_template = NULL,
  custom_attributes = NULL,
  accelerator_type = NULL,
  endpoint_name_prefix = NULL
)
```

Arguments:

`model_name` (str): Model name (as created by 'CreateModel').

`instance_count` (int): The number of instances of a new endpoint for model inference.

`instance_type` (str): The type of EC2 instance to use for model inference, for example, 'ml.c5.xlarge'.

`accept_type` (str): The model output format to be used for getting inferences with the shadow endpoint. Valid values are "text/csv" for CSV and "application/jsonlines". Default is the same as content_type.

`content_type` (str): The model input format to be used for getting inferences with the shadow endpoint. Valid values are "text/csv" for CSV and "application/jsonlines". Default is the same as dataset format.

`content_template` (str): A template string to be used to construct the model input from dataset instances. It is only used when "model_content_type" is "application/jsonlines". The template should have one and only one placeholder $features which will be replaced by a features list for to form the model inference input.

`custom_attributes` (str): Provides additional information about a request for an inference submitted to a model hosted at an Amazon SageMaker endpoint. The information is an opaque value that is forwarded verbatim. You could use this value, for example, to provide an ID that you can use to track a request or to provide other metadata that a service endpoint was programmed to process. The value must consist of no more than 1024 visible US-ASCII characters as specified in Section 3.3.6. Field Value Components (https://tools.ietf.org/html/rfc7230#section-3.2.6) of the Hypertext Transfer Protocol (HTTP/1.1).

`accelerator_type` (str): The Elastic Inference accelerator type to deploy to the model endpoint instance for making inferences to the model, see https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html.

`endpoint_name_prefix` (str): The endpoint name prefix of a new endpoint. Must follow pattern "^[a-zA-Z0-9](-*[a-zA-Z0-9]".

Method `get_predictor_config()`**:** Returns part of the predictor dictionary of the analysis config.

Usage:

```
ModelConfig$get_predictor_config()
```

Method `format()`**:** format class

Usage:

```
ModelConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ModelConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ModelPredictedLabelConfig` *ModelPredictedLabelConfig Class*

Description

Config object to extract a predicted label from the model output.

Public fields

`label` Predicted label of the same type as the label in the dataset

`probability` Optional index or JSONPath location in the model

`probability_threshold` An optional value for binary prediction task

`predictor_config` Predictor dictionary of the analysis config.

Methods

Public methods:

- `ModelPredictedLabelConfig$new()`
- `ModelPredictedLabelConfig$get_predictor_config()`
- `ModelPredictedLabelConfig$format()`
- `ModelPredictedLabelConfig$clone()`

Method `new()`**:** Initializes a model output config to extract the predicted label. The following examples show different parameter configurations depending on the endpoint:

- Regression Task: The model returns the score, e.g. 1.2. we don't need to specify anything. For json output, e.g. `list('score'=1.2)` we can set ''label='score"'
- Binary classification:
- The model returns a single probability and we would like to classify as 'yes' those with a probability exceeding 0.2. We can set ''probability_threshold=0.2, label_headers='yes"'.
- The model returns `list('probability'=0.3)`, for which we would like to apply a threshold of 0.5 to obtain a predicted label in `list(0, 1)`. In this case we can set ''label='probability"'.
- The model returns a tuple of the predicted label and the probability. In this case we can set ''label=0'.
- Multiclass classification:
- The model returns `list('labels'= c('cat', 'dog', 'fish'), 'probabilities'=c(0.35, 0.25, 0.4))`. In this case we would set the ''probability='probabilities"' and ''label='labels"' and infer the predicted label to be ''fish.''
- The model returns `list('predicted_label'='fish', 'probabilities'=c(0.35, 0.25, 0.4]))`. In this case we would set the ''label='predicted_label"'.
- The model returns `c(0.35, 0.25, 0.4)`. In this case, we can set ''label_headers=['cat','dog','fish']"' and infer the predicted label to be ''fish.''

Usage:

```
ModelPredictedLabelConfig$new(
  label = NULL,
  probability = NULL,
  probability_threshold = NULL,
  label_headers = NULL
)
```

Arguments:

`label` (str or [integer] or list[integer]): Optional index or JSONPath location in the model output for the prediction. In case, this is a predicted label of the same type as the label in the dataset no further arguments need to be specified.

`probability` (str or [integer] or list[integer]): Optional index or JSONPath location in the model output for the predicted scores.

`probability_threshold` (float): An optional value for binary prediction tasks in which the model returns a probability, to indicate the threshold to convert the prediction to a boolean value. Default is 0.5.

`label_headers` (list): List of label values - one for each score of the "probability".

Method `get_predictor_config()`: Returns probability_threshold, predictor config.

Usage:

```
ModelPredictedLabelConfig$get_predictor_config()
```

Method `format()`: format class

Usage:

```
ModelPredictedLabelConfig$format()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ModelPredictedLabelConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NetworkConfig` *NetworkConfig class*

Description

Accepts network configuration parameters and provides a method to turn these parameters into a dictionary.

Methods

Public methods:

- `NetworkConfig$new()`
- `NetworkConfig$to_request_list()`
- `NetworkConfig$format()`
- `NetworkConfig$clone()`

Method `new()`: Initialize a "NetworkConfig" instance. NetworkConfig accepts network configuration parameters and provides a method to turn these parameters into a dictionary.

Usage:

```
NetworkConfig$new(
  enable_network_isolation = FALSE,
  security_group_ids = NULL,
  subnets = NULL,
  encrypt_inter_container_traffic = NULL
)
```

Arguments:

`enable_network_isolation` (bool): Boolean that determines whether to enable network isolation.

`security_group_ids` ([str]): A list of strings representing security group IDs.

`subnets` ([str]): A list of strings representing subnets.

`encrypt_inter_container_traffic` (bool): Boolean that determines whether to encrypt inter-container traffic. Default value is None.

Method `to_request_list()`**:** Generates a request dictionary using the parameters provided to the class.

Usage:

```
NetworkConfig$to_request_list()
```

Method `format()`**:** format class

Usage:

```
NetworkConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NetworkConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`PDPConfig` *Config class for Partial Dependence Plots (PDP).*

Description

If PDP is requested, the Partial Dependence Plots will be included in the report, and the corresponding values will be included in the analysis output.

Super class

`sagemaker.common::ExplainabilityConfig` -> `PDPConfig`

Public fields

`pdp_config` PDP Config

Methods

Public methods:

- PDPConfig$new()
- PDPConfig$get_explainability_config()
- PDPConfig$clone()

Method `new()`**:** Initializes config for PDP.

Usage:

```
PDPConfig$new(features = NULL, grid_resolution = 15, top_k_features = 10)
```

Arguments:

`features` (None or list): List of features names or indices for which partial dependence plots must be computed and plotted. When ShapConfig is provided, this parameter is optional as Clarify will try to compute the partial dependence plots for top feature based on SHAP attributions. When ShapConfig is not provided, 'features' must be provided.

`grid_resolution` (int): In case of numerical features, this number represents that number of buckets that range of values must be divided into. This decides the granularity of the grid in which the PDP are plotted.

`top_k_features` (int): Set the number of top SHAP attributes to be selected to compute partial dependence plots.

Method `get_explainability_config()`**:** Returns config.

Usage:

```
PDPConfig$get_explainability_config()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PDPConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ProcessingInput` *ProcessingInput Class*

Description

Accepts parameters that specify an Amazon S3 input for a processing job and provides a method to turn those parameters into a dictionary.

Methods

Public methods:

- ProcessingInput$new()
- ProcessingInput$to_request_list()
- ProcessingInput$format()
- ProcessingInput$clone()

Method `new()`**:** Initializes a "ProcessingInput" instance. "ProcessingInput" accepts parameters that specify an Amazon S3 input for a processing job and provides a method to turn those parameters into a dictionary.

Usage:

```
ProcessingInput$new(
  source = NULL,
  destination = NULL,
  input_name = NULL,
  s3_data_type = c("S3Prefix", "ManifestFile"),
  s3_input_mode = c("File", "Pipe"),
  s3_data_distribution_type = c("FullyReplicated", "ShardedByS3Key"),
  s3_compression_type = c("None", "Gzip"),
  s3_input = NULL,
  dataset_definition = NULL,
  app_managed = FALSE
)
```

Arguments:

`source` (str): The source for the input. If a local path is provided, it will automatically be uploaded to S3 under: "s3://<default-bucket-name>/<job-name>/input/<input-name>".

`destination` (str): The destination of the input.

`input_name` (str): The name for the input. If a name is not provided, one will be generated (eg. "input-1").

`s3_data_type` (str): Valid options are "ManifestFile" or "S3Prefix".

`s3_input_mode` (str): Valid options are "Pipe" or "File".

`s3_data_distribution_type` (str): Valid options are "FullyReplicated" or "ShardedByS3Key".

`s3_compression_type` (str): Valid options are "None" or "Gzip".

`s3_input` (:class:'~sagemaker.dataset_definition.S3Input') Metadata of data objects stored in S3

`dataset_definition` (:class:'~sagemaker.dataset_definition.DatasetDefinition') DatasetDefinition input

`app_managed` (bool): Whether the input are managed by SageMaker or application

Method `to_request_list()`**:** Generates a request dictionary using the parameters provided to the class.

Usage:

```
ProcessingInput$to_request_list()
```

Method `format()`**:** format class

Usage:

```
ProcessingInput$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ProcessingInput$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Processor: `ProcessingJob`, `ProcessingOutput`, `Processor`, `ScriptProcessor`

`ProcessingJob` *ProccesingJob Class*

Description

Provides functionality to start, describe, and stop processing jobs.

Super class

`sagemaker.common::.Job` -> `ProcessingJob`

Public fields

`inputs` A list of :class:‘~sagemaker.processing.ProcessingInput‘ objects.

`outputs` A list of :class:‘~sagemaker.processing.ProcessingOutput‘ objects.

`output_kms_key` The output KMS key associated with the job

Methods

Public methods:

- `ProcessingJob$new()`
- `ProcessingJob$start_new()`
- `ProcessingJob$from_processing_name()`
- `ProcessingJob$from_processing_arn()`
- `ProcessingJob$wait()`
- `ProcessingJob$describe()`
- `ProcessingJob$stop()`
- `ProcessingJob$prepare_app_specification()`
- `ProcessingJob$prepare_output_config()`
- `ProcessingJob$prepare_processing_resources()`
- `ProcessingJob$prepare_stopping_condition()`

- ProcessingJob$clone()

Method `new()`**:** Initializes a Processing job.

Usage:

```
ProcessingJob$new(
  sagemaker_session = NULL,
  job_name = NULL,
  inputs = NULL,
  outputs = NULL,
  output_kms_key = NULL
)
```

Arguments:

`sagemaker_session` (:class:‘~sagemaker.session.Session‘): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain.

`job_name` (str): Name of the Processing job.

`inputs` (list[:class:‘~sagemaker.processing.ProcessingInput‘]): A list of :class:‘~sagemaker.processing.ProcessingInput‘ objects.

`outputs` (list[:class:‘~sagemaker.processing.ProcessingOutput‘]): A list of :class:‘~sagemaker.processing.ProcessingOu objects.

`output_kms_key` (str): The output KMS key associated with the job (default: None).

Method `start_new()`**:** Starts a new processing job using the provided inputs and outputs.

Usage:

```
ProcessingJob$start_new(processor, inputs, outputs, experiment_config)
```

Arguments:

`processor` (:class:‘~sagemaker.processing.Processor‘): The “Processor“ instance that started the job.

`inputs` (list[:class:‘~sagemaker.processing.ProcessingInput‘]): A list of :class:‘~sagemaker.processing.ProcessingInput‘ objects.

`outputs` (list[:class:‘~sagemaker.processing.ProcessingOutput‘]): A list of :class:‘~sagemaker.processing.ProcessingOu objects.

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: ’ExperimentName’, ’TrialName’, and ’TrialComponentDisplayName’.

Returns: :class:‘~sagemaker.processing.ProcessingJob‘: The instance of “ProcessingJob“ created using the “Processor“.

Method `from_processing_name()`**:** Initializes a “ProcessingJob“ from a processing job name.

Usage:

```
ProcessingJob$from_processing_name(sagemaker_session, processing_job_name)
```

Arguments:

`sagemaker_session` (:class:‘~sagemaker.session.Session‘): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain.

`processing_job_name` (str): Name of the processing job.

Returns: :class:‘~sagemaker.processing.ProcessingJob‘: The instance of “ProcessingJob“ created from the job name.

Method `from_processing_arn()`**:** Initializes a “ProcessingJob“ from a Processing ARN.

Usage:

```
ProcessingJob$from_processing_arn(sagemaker_session, processing_job_arn)
```

Arguments:

`sagemaker_session` (:class:‘~sagemaker.session.Session‘): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain.

`processing_job_arn` (str): ARN of the processing job.

Returns: :class:‘~sagemaker.processing.ProcessingJob‘: The instance of “ProcessingJob“ created from the processing job’s ARN.

Method `wait()`**:** Waits for the processing job to complete.

Usage:

```
ProcessingJob$wait(logs = TRUE)
```

Arguments:

`logs` (bool): Whether to show the logs produced by the job (default: True).

Method `describe()`**:** Prints out a response from the DescribeProcessingJob API call.

Usage:

```
ProcessingJob$describe()
```

Method `stop()`**:** the processing job.

Usage:

```
ProcessingJob$stop()
```

Method `prepare_app_specification()`**:** Prepares a dict that represents a ProcessingJob’s AppSpecification.

Usage:

```
ProcessingJob$prepare_app_specification(
  container_arguments,
  container_entrypoint,
  image_uri
)
```

Arguments:

`container_arguments` (list[str]): The arguments for a container used to run a processing job.

`container_entrypoint` (list[str]): The entrypoint for a container used to run a processing job.

`image_uri` (str): The container image to be run by the processing job.

Returns: dict: Represents AppSpecification which configures the processing job to run a specified Docker container image.

Method `prepare_output_config()`**:** Prepares a dict that represents a ProcessingOutputConfig.

Usage:

```
ProcessingJob$prepare_output_config(kms_key_id, outputs)
```

Arguments:

`kms_key_id` (str): The AWS Key Management Service (AWS KMS) key that Amazon SageMaker uses to encrypt the processing job output. KmsKeyId can be an ID of a KMS key, ARN of a KMS key, alias of a KMS key, or alias of a KMS key. The KmsKeyId is applied to all outputs.

`outputs` (list[dict]): Output configuration information for a processing job.

Returns: dict: Represents output configuration for the processing job.

Method `prepare_processing_resources()`**:** Prepares a dict that represents the ProcessingResources.

Usage:

```
ProcessingJob$prepare_processing_resources(
  instance_count,
  instance_type,
  volume_kms_key_id,
  volume_size_in_gb
)
```

Arguments:

`instance_count` (int): The number of ML compute instances to use in the processing job. For distributed processing jobs, specify a value greater than 1. The default value is 1.

`instance_type` (str): The ML compute instance type for the processing job.

`volume_kms_key_id` (str): The AWS Key Management Service (AWS KMS) key that Amazon SageMaker uses to encrypt data on the storage volume attached to the ML compute instance(s) that run the processing job.

`volume_size_in_gb` (int): The size of the ML storage volume in gigabytes that you want to provision. You must specify sufficient ML storage for your scenario.

Returns: dict: Represents ProcessingResources which identifies the resources, ML compute instances, and ML storage volumes to deploy for a processing job.

Method `prepare_stopping_condition()`**:** Prepares a dict that represents the job's StoppingCondition.

Usage:

```
ProcessingJob$prepare_stopping_condition(max_runtime_in_seconds)
```

Arguments:

`max_runtime_in_seconds` (int): Specifies the maximum runtime in seconds.

Returns: list

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ProcessingJob$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Processor: ProcessingInput, ProcessingOutput, Processor, ScriptProcessor

---

ProcessingOutput *ProcessingOutput Class*

Description

Accepts parameters that specify an Amazon S3 output for a processing job and provides a method to turn those parameters into a dictionary.

Methods

Public methods:

- ProcessingOutput$new()
- ProcessingOutput$to_request_list()
- ProcessingOutput$format()
- ProcessingOutput$clone()

Method `new()`**:** Initializes a "ProcessingOutput" instance. "ProcessingOutput" accepts parameters that specify an Amazon S3 output for a processing job and provides a method to turn those parameters into a dictionary.

Usage:

```
ProcessingOutput$new(
  source = NULL,
  destination = NULL,
  output_name = NULL,
  s3_upload_mode = c("EndOfJob", "Continuous"),
  app_managed = FALSE,
  feature_store_output = NULL
)
```

Arguments:

`source` (str): The source for the output.

`destination` (str): The destination of the output. If a destination is not provided, one will be generated: "s3://<default-bucket-name>/<job-name>/output/<output-name>".

`output_name` (str): The name of the output. If a name is not provided, one will be generated (eg. "output-1").

`s3_upload_mode` (str): Valid options are "EndOfJob" or "Continuous". s3_upload_mode (str): Valid options are "EndOfJob" or "Continuous".

`app_managed` (bool): Whether the input are managed by SageMaker or application

`feature_store_output` (:class:'~sagemaker.processing.FeatureStoreOutput') Configuration for processing job outputs of FeatureStore.

Method `to_request_list()`**:** Generates a request dictionary using the parameters provided to the class.

Usage:

```
ProcessingOutput$to_request_list()
```

Method `format()`**:** format class

Usage:

```
ProcessingOutput$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ProcessingOutput$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Processor: `ProcessingInput`, `ProcessingJob`, `Processor`, `ScriptProcessor`

`Processor` *Processor Class*

Description

Handles Amazon SageMaker Processing tasks.

Public fields

`role` An AWS IAM role name or ARN

`image_uri` The URI of the Docker image to use

`instance_count` The number of instances to run

`instance_type` The type of EC2 instance to use

`entrypoint` The entrypoint for the processing job

`volume_size_in_gb` Size in GB of the EBS volume

`volume_kms_key` A KMS key for the processing

`output_kms_key` The KMS key ID for processing job outputs

`max_runtime_in_seconds` Timeout in seconds

`base_job_name` Prefix for processing job name

`sagemaker_session` Session object which manages interactions with Amazon SageMaker

`env` Environment variables

`tags` List of tags to be passed

`network_config` A :class:‘~sagemaker.network.NetworkConfig‘

`jobs` Jobs ran /running

`latest_job` Previously ran jobs

`.current_job_name` Current job

`arguments` extra agruments

Methods

Public methods:

- Processor$new()
- Processor$run()
- Processor$format()
- Processor$clone()

Method `new()`**:** Initializes a "Processor" instance. The "Processor" handles Amazon SageMaker Processing tasks.

Usage:

```
Processor$new(
  role,
  image_uri,
  instance_count,
  instance_type,
  entrypoint = NULL,
  volume_size_in_gb = 30,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  env = NULL,
  tags = NULL,
  network_config = NULL
)
```

Arguments:

`role` (str): An AWS IAM role name or ARN. Amazon SageMaker Processing uses this role to access AWS resources, such as data stored in Amazon S3.

`image_uri` (str): The URI of the Docker image to use for the processing jobs.

`instance_count` (int): The number of instances to run a processing job with.

`instance_type` (str): The type of EC2 instance to use for processing, for example, 'ml.c4.xlarge'.

`entrypoint` (list[str]): The entrypoint for the processing job (default: NULL). This is in the form of a list of strings that make a command.

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data during processing (default: 30).

`volume_kms_key` (str): A KMS key for the processing volume (default: NULL).

`output_kms_key` (str): The KMS key ID for processing job outputs (default: NULL).

`max_runtime_in_seconds` (int): Timeout in seconds (default: NULL). After this amount of time, Amazon SageMaker terminates the job, regardless of its current status. If 'max_runtime_in_seconds' is not specified, the default value is 24 hours.

`base_job_name` (str): Prefix for processing job name. If not specified, the processor generates a default job name, based on the processing image name and current timestamp.

`sagemaker_session` (:class:'~sagemaker.session.Session'): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain.

env (dict[str, str]): Environment variables to be passed to the processing jobs (default: NULL).

tags (list[dict]): List of tags to be passed to the processing job (default: NULL). For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

network_config (:class:‘~sagemaker.network.NetworkConfig‘): A :class:‘~sagemaker.network.NetworkConfig‘ object that configures network isolation, encryption of inter-container traffic, security group IDs, and subnets.

Method `run()`**:** Runs a processing job.

Usage:

```
Processor$run(
  inputs = NULL,
  outputs = NULL,
  arguments = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL,
  experiment_config = NULL
)
```

Arguments:

inputs (list[:class:‘~sagemaker.processing.ProcessingInput‘]): Input files for the processing job. These must be provided as :class:‘~sagemaker.processing.ProcessingInput‘ objects (default: NULL).

outputs (list[:class:‘~sagemaker.processing.ProcessingOutput‘]): Outputs for the processing job. These can be specified as either path strings or :class:‘~sagemaker.processing.ProcessingOutput‘ objects (default: NULL).

arguments (list[str]): A list of string arguments to be passed to a processing job (default: NULL).

wait (bool): Whether the call should wait until the job completes (default: True).

logs (bool): Whether to show the logs produced by the job. Only meaningful when “wait“ is True (default: True).

job_name (str): Processing job name. If not specified, the processor generates a default job name, based on the base job name and current timestamp.

experiment_config (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: 'ExperimentName', 'TrialName', and 'TrialComponentDisplayName'.

Method `format()`**:** format class

Usage:

```
Processor$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Processor$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other Processor: ProcessingInput, ProcessingJob, ProcessingOutput, ScriptProcessor

---

ProfilerConfig *Configuration for collecting system and framework metrics of SageMaker training jobs.*

Description

SageMaker Debugger collects system and framework profiling information of training jobs and identify performance bottlenecks.

Methods

Public methods:

- ProfilerConfig$new()
- ProfilerConfig$to_request_list()
- ProfilerConfig$format()
- ProfilerConfig$clone()

Method `new()`**:** Initialize a "ProfilerConfig" instance. Pass the output of this class to the "profiler_config" parameter of the generic :class:'~sagemaker.estimator.Estimator' class and SageMaker Framework estimators.

Usage:

```
ProfilerConfig$new(
  s3_output_path = NULL,
  system_monitor_interval_millis = NULL,
  framework_profile_params = NULL
)
```

Arguments:

`s3_output_path` (str): The location in Amazon S3 to store the output. The default Debugger output path for profiling data is created under the default output path of the :class:'~sagemaker.estimator.Estimator' class. For example, s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/profiler-output/.

`system_monitor_interval_millis` (int): The time interval in milliseconds to collect system metrics. Available values are 100, 200, 500, 1000 (1 second), 5000 (5 seconds), and 60000 (1 minute) milliseconds. The default is 500 milliseconds.

`framework_profile_params` (:class:'~sagemaker.debugger.FrameworkProfile'): A parameter object for framework metrics profiling. Configure it using the :class:'~sagemaker.debugger.FrameworkProfile' class. To use the default framework profile parameters, pass "FrameworkProfile()". For more information about the default values, see :class:'~sagemaker.debugger.FrameworkProfile'.

Examples:

```
# The following example shows the basic ``profiler_config``
# parameter configuration, enabling system monitoring every 5000 milliseconds
# and framework profiling with default parameter values.
library(sagemaker.common)

profiler_config = ProfilerConfig$new(
      system_monitor_interval_millis = 5000,
      framework_profile_params = FrameworkProfile$new()
 )
```

Method `to_request_list()`**:** Generate a request dictionary using the parameters provided when initializing the object.

Usage:

```
ProfilerConfig$to_request_list()
```

Returns: dict: An portion of an API request as a dictionary.

Method `format()`**:** format class

Usage:

```
ProfilerConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ProfilerConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `ProfilerConfig$new`
## ------------------------------------------------

# The following example shows the basic ``profiler_config``
# parameter configuration, enabling system monitoring every 5000 milliseconds
# and framework profiling with default parameter values.
library(sagemaker.common)

profiler_config = ProfilerConfig$new(
      system_monitor_interval_millis = 5000,
      framework_profile_params = FrameworkProfile$new()
 )
```

`ProfilerRule` *The SageMaker Debugger ProfilerRule class configures *profiling* rules.*

Description

SageMaker Debugger profiling rules automatically analyze hardware system resource utilization and framework metrics of a training job to identify performance bottlenecks. SageMaker Debugger comes pre-packaged with built-in *profiling* rules. For example, the profiling rules can detect if GPUs are underutilized due to CPU bottlenecks or IO bottlenecks. For a full list of built-in rules for debugging, see 'List of Debugger Built-in Rules' https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-built-in-rules.html. You can also write your own profiling rules using the Amazon SageMaker Debugger APIs.

Super class

`sagemaker.common::RuleBase` -> `ProfilerRule`

Methods

Public methods:

- `ProfilerRule$sagemaker()`
- `ProfilerRule$custom()`
- `ProfilerRule$to_profiler_rule_config_list()`
- `ProfilerRule$clone()`

Method `sagemaker()`**:** Initialize a "ProfilerRule" object for a *built-in* profiling rule. The rule analyzes system and framework metrics of a given training job to identify performance bottlenecks.

Usage:

```
ProfilerRule$sagemaker(
  base_config,
  name = NULL,
  container_local_output_path = NULL,
  s3_output_path = NULL
)
```

Arguments:

`base_config` (`sagemaker.debugger::ProfilerReport`): The base rule configuration object returned from the `sagemaker.debugger` method. For example, `sagemaker.debugger::ProfilerReport$new()`. For a full list of built-in rules for debugging, see 'List of Debugger Built-in Rules' https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-built-in-rules.html.

`name` (str): The name of the profiler rule. If one is not provided, the name of the base_config will be used.

`container_local_output_path` (str): The path in the container.

s3_output_path (str): The location in Amazon S3 to store the profiling output data. The default Debugger output path for profiling data is created under the default output path of the :class:‘~sagemaker.estimator.Estimator‘ class. For example, s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/profiler-output/.

Returns: :class:‘~sagemaker.debugger.ProfilerRule‘: The instance of the built-in ProfilerRule.

Method `custom()`**:** Initialize a “ProfilerRule“ object for a *custom* profiling rule. You can create a rule that analyzes system and framework metrics emitted during the training of a model and monitors conditions that are critical for the success of a training job.

Usage:

```
ProfilerRule$custom(
  name,
  image_uri,
  instance_type,
  volume_size_in_gb,
  source = NULL,
  rule_to_invoke = NULL,
  container_local_output_path = NULL,
  s3_output_path = NULL,
  rule_parameters = NULL
)
```

Arguments:

name (str): The name of the profiler rule.

image_uri (str): The URI of the image to be used by the proflier rule.

instance_type (str): Type of EC2 instance to use, for example, ’ml.c4.xlarge’.

volume_size_in_gb (int): Size in GB of the EBS volume to use for storing data.

source (str): A source file containing a rule to invoke. If provided, you must also provide rule_to_invoke. This can either be an S3 uri or a local path.

rule_to_invoke (str): The name of the rule to invoke within the source. If provided, you must also provide the source.

container_local_output_path (str): The path in the container.

s3_output_path (str): The location in Amazon S3 to store the output. The default Debugger output path for profiling data is created under the default output path of the :class:‘~sagemaker.estimator.Estimator‘ class. For example, s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/profiler-output/.

rule_parameters (dict): A dictionary of parameters for the rule.

Returns: :class:‘~sagemaker.debugger.ProfilerRule‘: The instance of the custom ProfilerRule.

Method `to_profiler_rule_config_list()`**:** Generates a request dictionary using the parameters provided when initializing object.

Usage:

```
ProfilerRule$to_profiler_rule_config_list()
```

Returns: lict: An portion of an API request as a dictionary.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ProfilerRule$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`PythonProfiler` *PythonProfiler enum environment list*

Description

List the Python profiler options for Python profiling.

Usage

```
PythonProfiler
```

Format

An object of class `PythonProfiler` (inherits from `Enum`, `environment`) of length 2.

Value

environment containing [CPROFILE, PYINSTRUMENT]

`PythonProfilingConfig` *PythonProfilingConfig Class*

Description

The configuration for framework metrics to be collected for Python profiling.

Super class

`sagemaker.common::MetricsConfigBase` -> `PythonProfilingConfig`

Methods

Public methods:

- `PythonProfilingConfig$new()`
- `PythonProfilingConfig$clone()`

Method `new()`**:** Choose a Python profiler: cProfile or Pyinstrument. Specify target steps or a target duration to profile. If no parameter is specified, it profiles based on profiling configurations preset by the `profile_default_steps` parameter, which is set to 'True' by default. If you specify the following parameters, then the `profile_default_steps` parameter will be ignored.

Usage:

```
PythonProfilingConfig$new(
  start_step = NULL,
  num_steps = NULL,
  start_unix_time = NULL,
  duration = NULL,
  profile_default_steps = FALSE,
  python_profiler = PythonProfiler$CPROFILE,
  cprofile_timer = cProfileTimer$TOTAL_TIME
)
```

Arguments:

`start_step` (int): The step to start profiling. The default is step 9.

`num_steps` (int): The number of steps to profile. The default is for 3 steps.

`start_unix_time` (int): The Unix time to start profiling.

`duration` (float): The duration in seconds to profile.

`profile_default_steps` (bool): Indicates whether the default configuration should be used. If set to 'True', Python profiling will be done at step 9, 10, and 11 of training, using cProfiler and collecting metrics based on the total time, cpu time, and off cpu time for these three steps respectively. The default is "True".

`python_profiler` (PythonProfiler): The Python profiler to use to collect python profiling stats. Available options are ""cProfile"" and ""Pyinstrument"". The default is ""cProfile"". Instead of passing the string values, you can also use the enumerator util, :class:'~sagemaker.debugger.utils.PythonProfi to choose one of the available options.

`cprofile_timer` (cProfileTimer): The timer to be used by cProfile when collecting python profiling stats. Available options are ""total_time"", ""cpu_time"", and ""off_cpu_time"". The default is ""total_time"". If you choose Pyinstrument, this parameter is ignored. Instead of passing the string values, you can also use the enumerator util, :class:'~sagemaker.debugger.utils.cProfileTimer', to choose one of the available options.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
PythonProfilingConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`RedshiftDatasetDefinition` *DatasetDefinition for Redshift.*

Description

With this input, SQL queries will be executed using Redshift to generate datasets to S3.

Super class

`sagemaker.core::ApiObject` -> `RedshiftDatasetDefinition`

Methods

Public methods:

- `RedshiftDatasetDefinition$new()`
- `RedshiftDatasetDefinition$clone()`

Method `new()`**:** Initialize RedshiftDatasetDefinition.

Usage:

```
RedshiftDatasetDefinition$new(
  cluster_id = NULL,
  database = NULL,
  db_user = NULL,
  query_string = NULL,
  cluster_role_arn = NULL,
  output_s3_uri = NULL,
  kms_key_id = NULL,
  output_format = NULL,
  output_compression = NULL
)
```

Arguments:

`cluster_id` (str, default=None): The Redshift cluster Identifier. database (str, default=None): The name of the Redshift database used in Redshift query execution.

`database` (str, default=None): The name of the Redshift database used in Redshift query execution.

`db_user` (str, default=None): The database user name used in Redshift query execution.

`query_string` (str, default=None): The SQL query statements to be executed.

`cluster_role_arn` (str, default=None): The IAM role attached to your Redshift cluster that Amazon SageMaker uses to generate datasets.

`output_s3_uri` (str, default=None): The location in Amazon S3 where the Redshift query results are stored.

`kms_key_id` (str, default=None): The AWS Key Management Service (AWS KMS) key that Amazon SageMaker uses to encrypt data from a Redshift execution.

`output_format` (str, default=None): The data storage format for Redshift query results. Valid options are "PARQUET", "CSV"

`output_compression` (str, default=None): The compression used for Redshift query results. Valid options are "None", "GZIP", "SNAPPY", "ZSTD", "BZIP2"

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RedshiftDatasetDefinition$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Rule *Debug Rule Class*

Description

The SageMaker Debugger Rule class configures *debugging* rules to debug your training job. The debugging rules analyze tensor outputs from your training job and monitor conditions that are critical for the success of the training job. SageMaker Debugger comes pre-packaged with built-in *debugging* rules. For example, the debugging rules can detect whether gradients are getting too large or too small, or if a model is overfitting. For a full list of built-in rules for debugging, see 'List of Debugger Built-in Rules' https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-built-in-rules.html. You can also write your own rules using the custom rule classmethod.

Super class

sagemaker.common::RuleBase -> Rule

Public fields

collection_configs A list of :class:'~sagemaker.debugger.CollectionConfig

actions Placeholder

Methods

Public methods:

- Rule$new()
- Rule$sagemaker()
- Rule$custom()
- Rule$prepare_actions()
- Rule$to_debugger_rule_config_list()
- Rule$clone()

Method new(): Configure the debugging rules using the following classmethods.

Usage:

```
Rule$new(
  name = NULL,
  image_uri = NULL,
  instance_type = NULL,
  container_local_output_path = NULL,
  s3_output_path = NULL,
  volume_size_in_gb = NULL,
  rule_parameters = NULL,
  collections_to_save = NULL,
  actions = NULL
)
```

Arguments:

`name` (str): The name of the rule.

`image_uri` (str): The image URI to use the rule.

`instance_type` (str): Type of EC2 instance to use. For example, 'ml.c4.xlarge'.

`container_local_output_path` (str): The local path to store the Rule output.

`s3_output_path` (str): The location in S3 to store the output.

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data.

`rule_parameters` (dict): A dictionary of parameters for the rule.

`collections_to_save` ([sagemaker.debugger.CollectionConfig]): Optional. A list of :class:'~sagemaker.debugger.Colle objects to be saved.

`actions` :

Method `sagemaker()`**:** Initialize a "Rule" object for a `built-in` debugging rule.

Usage:

```
Rule$sagemaker(
  base_config,
  name = NULL,
  container_local_output_path = NULL,
  s3_output_path = NULL,
  other_trials_s3_input_paths = NULL,
  rule_parameters = NULL,
  collections_to_save = NULL,
  actions = NULL
)
```

Arguments:

`base_config` (dict): Required. This is the base rule config dictionary returned from the :class:`sagemaker.debugger` method. For example, `sagemaker.debugger::dead_relu()`. For a full list of built-in rules for debugging, see 'List of Debugger Built-in Rules' https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-built-in-rules.html.

`name` (str): Optional. The name of the debugger rule. If one is not provided, the name of the base_config will be used.

`container_local_output_path` (str): Optional. The local path in the rule processing container.

`s3_output_path` (str): Optional. The location in Amazon S3 to store the output tensors. The default Debugger output path for debugging data is created under the default output path of the :class:'~sagemaker.estimator.Estimator' class. For example, s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/debug-output/.

`other_trials_s3_input_paths` ([str]): Optional. The Amazon S3 input paths of other trials to use the SimilarAcrossRuns rule.

`rule_parameters` (dict): Optional. A dictionary of parameters for the rule.

`collections_to_save` (:class:`sagemaker.debugger::CollectionConfig`): Optional. A list of :class:`sagemaker.debugger::CollectionConfig` objects to be saved.

`actions` :

Returns: :class:'~sagemaker.debugger.Rule': An instance of the built-in rule.

Method `custom()`: Initialize a "Rule" object for a *custom* debugging rule. You can create a custom rule that analyzes tensors emitted during the training of a model and monitors conditions that are critical for the success of a training job. For more information, see 'Create Debugger Custom Rules for Training Job Analysis' https://docs.aws.amazon.com/sagemaker/latest/dg/debugger-custom-rules.html.

Usage:

```
Rule$custom(
  name,
  image_uri,
  instance_type,
  volume_size_in_gb,
  source = NULL,
  rule_to_invoke = NULL,
  container_local_output_path = NULL,
  s3_output_path = NULL,
  other_trials_s3_input_paths = NULL,
  rule_parameters = NULL,
  collections_to_save = NULL,
  actions = NULL
)
```

Arguments:

`name` (str): Required. The name of the debugger rule.

`image_uri` (str): Required. The URI of the image to be used by the debugger rule.

`instance_type` (str): Required. Type of EC2 instance to use, for example, 'ml.c4.xlarge'.

`volume_size_in_gb` (int): Required. Size in GB of the EBS volume to use for storing data.

`source` (str): Optional. A source file containing a rule to invoke. If provided, you must also provide rule_to_invoke. This can either be an S3 uri or a local path.

`rule_to_invoke` (str): Optional. The name of the rule to invoke within the source. If provided, you must also provide source.

`container_local_output_path` (str): Optional. The local path in the container.

`s3_output_path` (str): Optional. The location in Amazon S3 to store the output tensors. The default Debugger output path for debugging data is created under the default output path of the :class:'~sagemaker.estimator.Estimator' class. For example, s3://sagemaker-<region>-<12digit_account_id>/<training-job-name>/debug-output/.

`other_trials_s3_input_paths` ([str]): Optional. The Amazon S3 input paths of other trials to use the SimilarAcrossRuns rule.

`rule_parameters` (dict): Optional. A dictionary of parameters for the rule.

`collections_to_save` ([sagemaker.debugger.CollectionConfig]): Optional. A list of :class:'~sagemaker.debugger.Colle objects to be saved.

`actions` :

Returns: :class:'~sagemaker.debugger.Rule': The instance of the custom rule.

Method `prepare_actions()`: Prepare actions for Debugger Rule.

Usage:

```
Rule$prepare_actions(training_job_name)
```

Arguments:

`training_job_name` (str): The training job name. To be set as the default training job prefix for the StopTraining action if it is specified.

Method `to_debugger_rule_config_list()`**:** Generates a request dictionary using the parameters provided when initializing object.

Usage:

```
Rule$to_debugger_rule_config_list()
```

Returns: dict: An portion of an API request as a dictionary.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Rule$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`RuleBase` *The SageMaker Debugger rule base class that cannot be instantiated directly.*

Description

Debugger rule classes inheriting this RuleBase class are :class:‘~sagemaker.debugger.Rule‘ and :class:‘~sagemaker.debugger.ProfilerRule‘. Do not directly use the rule base class to instantiate a SageMaker Debugger rule. Use the :class:‘~sagemaker.debugger.Rule‘ classmethods for debugging and the :class:‘~sagemaker.debugger.ProfilerRule‘ classmethods for profiling.

Public fields

`name` (str): The name of the rule.

`image_uri` (str): The image URI to use the rule.

`instance_type` (str): Type of EC2 instance to use. For example, 'ml.c4.xlarge'.

`container_local_output_path` (str): The local path to store the Rule output.

`s3_output_path` (str): The location in S3 to store the output.

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data.

`rule_parameters` (dict): A dictionary of parameters for the rule.

Methods

Public methods:

- RuleBase$new()
- RuleBase$format()
- RuleBase$clone()

Method `new()`**:** Initialize RuleBase class

Usage:

```
RuleBase$new(
  name = NULL,
  image_uri = NULL,
  instance_type = NULL,
  container_local_output_path = NULL,
  s3_output_path = NULL,
  volume_size_in_gb = NULL,
  rule_parameters = NULL
)
```

Arguments:

`name` (str): The name of the rule.

`image_uri` (str): The image URI to use the rule.

`instance_type` (str): Type of EC2 instance to use. For example, 'ml.c4.xlarge'.

`container_local_output_path` (str): The local path to store the Rule output.

`s3_output_path` (str): The location in S3 to store the output.

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data.

`rule_parameters` (dict): A dictionary of parameters for the rule.

Method `format()`**:** format class

Usage:

```
RuleBase$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RuleBase$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

RunArgs *RunArgs Class*

Description

Accepts parameters that correspond to ScriptProcessors.

Public fields

`code` This can be an S3 URI or a local path to a file with the framework script to run

`inputs` Input files for the processing job

`outputs` Outputs for the processing job

`arguments` A list of string arguments to be passed to a processing job

Methods

Public methods:

- `RunArgs$new()`
- `RunArgs$clone()`

Method `new()`**:** An instance of this class is returned from the "get_run_args()" method on processors, and is used for normalizing the arguments so that they can be passed to :class:'~sagemaker.workflow.steps.ProcessingS

Usage:

```
RunArgs$new(code, inputs = NULL, outputs = NULL, arguments = NULL)
```

Arguments:

`code` (str): This can be an S3 URI or a local path to a file with the framework script to run.

`inputs` (list[:class:'~sagemaker.processing.ProcessingInput']): Input files for the processing job. These must be provided as :class:'~sagemaker.processing.ProcessingInput' objects (default: None).

`outputs` (list[:class:'~sagemaker.processing.ProcessingOutput']): Outputs for the processing job. These can be specified as either path strings or :class:'~sagemaker.processing.ProcessingOutput' objects (default: None).

`arguments` (list[str]): A list of string arguments to be passed to a processing job (default: None).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RunArgs$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

S3Input *Metadata of data objects stored in S3.*

Description

Two options are provided: specifying a S3 prefix or by explicitly listing the files in manifest file and referencing the manifest file's S3 path.

Super class

`sagemaker.core::ApiObject` -> `S3Input`

Methods

Public methods:

- `S3Input$new()`
- `S3Input$clone()`

Method `new()`**:** Initialize S3Input.

Usage:

```
S3Input$new(
  s3_uri = NULL,
  local_path = NULL,
  s3_data_type = "S3Prefix",
  s3_input_mode = "File",
  s3_data_distribution_type = "FullyReplicated",
  s3_compression_type = NULL
)
```

Arguments:

`s3_uri` (str, default=None): the path to a specific S3 object or a S3 prefix

`local_path` (str, default=None): the path to a local directory. If not provided, skips data download by SageMaker platform.

`s3_data_type` (str, default="S3Prefix"): Valid options are "ManifestFile" or "S3Prefix".

`s3_input_mode` (str, default="File"): Valid options are "Pipe" or "File".

`s3_data_distribution_type` (str, default="FullyReplicated"): Valid options are "FullyReplicated" or "ShardedByS3Key".

`s3_compression_type` (str, default=None): Valid options are "None" or "Gzip"

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
S3Input$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Note

Note: Strong consistency is not guaranteed if S3Prefix is provided here. S3 list operations are not strongly consistent. Use ManifestFile if strong consistency is required.

`SageMakerClarifyProcessor`

SageMakerClarifyProcessor Class

Description

Handles SageMaker Processing task to compute bias metrics and explain a model.

Super class

`sagemaker.common::Processor` -> `SageMakerClarifyProcessor`

Public fields

`job_name_prefix` Processing job name prefix

Methods

Public methods:

- `SageMakerClarifyProcessor$new()`
- `SageMakerClarifyProcessor$run()`
- `SageMakerClarifyProcessor$run_pre_training_bias()`
- `SageMakerClarifyProcessor$run_post_training_bias()`
- `SageMakerClarifyProcessor$run_bias()`
- `SageMakerClarifyProcessor$run_explainability()`
- `SageMakerClarifyProcessor$clone()`

Method `new()`**:** Initializes a "Processor" instance, computing bias metrics and model explanations.

Usage:

```
SageMakerClarifyProcessor$new(
  role,
  instance_count,
  instance_type,
  volume_size_in_gb = 30,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  sagemaker_session = NULL,
  env = NULL,
  tags = NULL,
```

```
  network_config = NULL,
  job_name_prefix = NULL,
  version = NULL
)
```

Arguments:

`role` (str): An AWS IAM role name or ARN. Amazon SageMaker Processing uses this role to access AWS resources, such as data stored in Amazon S3.

`instance_count` (int): The number of instances to run a processing job with.

`instance_type` (str): The type of EC2 instance to use for processing, for example, 'ml.c4.xlarge'.

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data during processing (default: 30).

`volume_kms_key` (str): A KMS key for the processing volume (default: None).

`output_kms_key` (str): The KMS key ID for processing job outputs (default: None).

`max_runtime_in_seconds` (int): Timeout in seconds (default: None). After this amount of time, Amazon SageMaker terminates the job, regardless of its current status. If 'max_runtime_in_seconds' is not specified, the default value is 24 hours.

`sagemaker_session` (:class:'~sagemaker.session.Session'): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain.

`env` (dict[str, str]): Environment variables to be passed to the processing jobs (default: None).

`tags` (list[dict]): List of tags to be passed to the processing job (default: None). For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

`network_config` (:class:'~sagemaker.network.NetworkConfig'): A :class:'~sagemaker.network.NetworkConfig' object that configures network isolation, encryption of inter-container traffic, security group IDs, and subnets.

`job_name_prefix` (str): Processing job name prefix.

`version` (str): Clarify version want to be used.

Method `run()`**:** Overriding the base class method but deferring to specific run_* methods.

Usage:

```
SageMakerClarifyProcessor$run()
```

Method `run_pre_training_bias()`**:** Runs a ProcessingJob to compute the requested bias 'methods' of the input data. Computes the requested methods that compare 'methods' (e.g. fraction of examples) for the sensitive group vs the other examples.

Usage:

```
SageMakerClarifyProcessor$run_pre_training_bias(
  data_config,
  data_bias_config,
  methods = "all",
  wait = TRUE,
  logs = TRUE,
  job_name = NULL,
  kms_key = NULL,
  experiment_config = NULL
)
```

Arguments:

`data_config` (:class:‘~sagemaker.clarify.DataConfig‘): Config of the input/output data.

`data_bias_config` (:class:‘~sagemaker.clarify.BiasConfig‘): Config of sensitive groups.

`methods` (str or list[str]): Selector of a subset of potential metrics:

- ‘CI‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ci.html
- ‘DPL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dpl.html
- ‘KL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-kl.html
- ‘JS‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-js.html
- ‘LP‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-lp.html
- ‘TVD‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-tvd.html
- ‘KS‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ks.html
- ‘CDDL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-cddl.html

Defaults to computing all.

`wait` (bool): Whether the call should wait until the job completes (default: True).

`logs` (bool): Whether to show the logs produced by the job. Only meaningful when “wait“ is True (default: True).

`job_name` (str): Processing job name. If not specified, a name is composed of "Clarify-Pretraining-Bias" and current timestamp.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the user code file (default: None).

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: ’ExperimentName’, ’TrialName’, and ’TrialComponentDisplayName’.

Method `run_post_training_bias()`**:** Runs a ProcessingJob to compute the requested bias ’methods’ of the model predictions. Spins up a model endpoint, runs inference over the input example in the ’s3_data_input_path’ to obtain predicted labels. Computes a the requested methods that compare ’methods’ (e.g. accuracy, precision, recall) for the sensitive group vs the other examples.

Usage:

```
SageMakerClarifyProcessor$run_post_training_bias(
  data_config,
  data_bias_config,
  model_config,
  model_predicted_label_config,
  methods = "all",
  wait = TRUE,
  logs = TRUE,
```

```
  job_name = NULL,
  kms_key = NULL,
  experiment_config = NULL
)
```

Arguments:

`data_config` (:class:‘~sagemaker.clarify.DataConfig‘): Config of the input/output data.

`data_bias_config` (:class:‘~sagemaker.clarify.BiasConfig‘): Config of sensitive groups.

`model_config` (:class:‘~sagemaker.clarify.ModelConfig‘): Config of the model and its endpoint to be created.

`model_predicted_label_config` (:class:‘~sagemaker.clarify.ModelPredictedLabelConfig‘): Config of how to extract the predicted label from the model output.

`methods` (str or list[str]): Selector of a subset of potential metrics:

- ‘CI‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ci.html
- ‘DPL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dpl
- ‘KL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-kl.html
- ‘JS‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-js.html
- ‘LP‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-lp.html
- ‘TVD‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-tvd
- ‘KS‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ks.html
- ‘CDDL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-c

  Defaults to computing all.

`wait` (bool): Whether the call should wait until the job completes (default: True).

`logs` (bool): Whether to show the logs produced by the job. Only meaningful when “wait“ is True (default: True).

`job_name` (str): Processing job name. If not specified, a name is composed of "Clarify-Posttraining-Bias" and current timestamp.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the user code file (default: None).

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: ’ExperimentName’, ’TrialName’, and ’TrialComponentDisplayName’.

Method `run_bias()`**:** Runs a ProcessingJob to compute the requested bias ’methods’ of the model predictions. Spins up a model endpoint, runs inference over the input example in the ’s3_data_input_path’ to obtain predicted labels. Computes a the requested methods that compare ’methods’ (e.g. accuracy, precision, recall) for the sensitive group vs the other examples.

Usage:

```
SageMakerClarifyProcessor$run_bias(
  data_config,
  bias_config,
  model_config,
  model_predicted_label_config = NULL,
  pre_training_methods = "all",
  post_training_methods = "all",
  wait = TRUE,
  logs = TRUE,
  job_name = NULL,
  kms_key = NULL,
  experiment_config = NULL
)
```

Arguments:

`data_config` (:class:‘~sagemaker.clarify.DataConfig‘): Config of the input/output data.

`bias_config` (:class:‘~sagemaker.clarify.BiasConfig‘): Config of sensitive groups.

`model_config` (:class:‘~sagemaker.clarify.ModelConfig‘): Config of the model and its endpoint to be created.

`model_predicted_label_config` (:class:‘~sagemaker.clarify.ModelPredictedLabelConfig‘): Config of how to extract the predicted label from the model output.

`pre_training_methods` (str or list[str]): Selector of a subset of potential metrics:

- ‘CI‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ci.html
- ‘DPL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dpl.html
- ‘KL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-kl.html
- ‘JS‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-js.html
- ‘LP‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-lp.html
- ‘TVD‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-tvd.html
- ‘KS‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ks.html
- ‘CDDL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-cddl.html

  Defaults to computing all.

`post_training_methods` (str or list[str]): Selector of a subset of potential metrics:

- ‘DPPL‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dppl.html
- ‘DI‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-di.html
- ‘DCA‘ https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dca.html

- 'DCR' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dcr
html
- 'RD' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-rd.
html
- 'DAR' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-dar
html
- 'DRR' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-drr
html
- 'AD' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ad.
html
- 'CDDPL' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-
html
- 'TE' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-te.
html
- 'FT' https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-post-training-bias-metric-ft.
html

Defaults to computing all.

`wait` (bool): Whether the call should wait until the job completes (default: True).

`logs` (bool): Whether to show the logs produced by the job. Only meaningful when "wait" is True (default: True).

`job_name` (str): Processing job name. If not specified, a name is composed of "Clarify-Bias" and current timestamp.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the user code file (default: None).

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: 'ExperimentName', 'TrialName', and 'TrialComponentDisplayName'.

Method `run_explainability()`**:** Runs a ProcessingJob computing for each example in the input the feature importance. Currently, only SHAP is supported as explainability method. Spins up a model endpoint. For each input example in the 's3_data_input_path' the SHAP algorithm determines feature importance, by creating 'num_samples' copies of the example with a subset of features replaced with values from the 'baseline'. Model inference is run to see how the prediction changes with the replaced features. If the model output returns multiple scores importance is computed for each of them. Across examples, feature importance is aggregated using 'agg_method'.

Usage:

```
SageMakerClarifyProcessor$run_explainability(
  data_config,
  model_config,
  explainability_config,
  model_scores = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL,
  kms_key = NULL,
  experiment_config = NULL
)
```

Arguments:

`data_config` (:class:‘~sagemaker.clarify.DataConfig‘): Config of the input/output data.

`model_config` (:class:‘~sagemaker.clarify.ModelConfig‘): Config of the model and its endpoint to be created.

`explainability_config` (:class:‘~sagemaker.clarify.ExplainabilityConfig‘): Config of the specific explainability method. Currently, only SHAP is supported.

`model_scores` : Index or JSONPath location in the model output for the predicted scores to be explained. This is not required if the model output is a single score.

`wait` (bool): Whether the call should wait until the job completes (default: True).

`logs` (bool): Whether to show the logs produced by the job. Only meaningful when “wait“ is True (default: True).

`job_name` (str): Processing job name. If not specified, a name is composed of "Clarify-Explainability" and current timestamp.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the user code file (default: None).

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: ’ExperimentName’, ’TrialName’, and ’TrialComponentDisplayName’.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SageMakerClarifyProcessor$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ScriptProcessor` *Script Processor class*

Description

Handles Amazon SageMaker processing tasks for jobs using a machine learning framework.

Super class

`sagemaker.common::Processor` -> `ScriptProcessor`

Methods

Public methods:

- `ScriptProcessor$new()`
- `ScriptProcessor$get_run_args()`
- `ScriptProcessor$run()`
- `ScriptProcessor$clone()`

Method `new()`**:** Initializes a "ScriptProcessor" instance. The "ScriptProcessor" handles Amazon SageMaker Processing tasks for jobs using a machine learning framework, which allows for providing a script to be run as part of the Processing Job.

Usage:

```
ScriptProcessor$new(
  role,
  image_uri,
  command,
  instance_count,
  instance_type,
  volume_size_in_gb = 30,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  env = NULL,
  tags = NULL,
  network_config = NULL
)
```

Arguments:

`role` (str): An AWS IAM role name or ARN. Amazon SageMaker Processing uses this role to access AWS resources, such as data stored in Amazon S3.

`image_uri` (str): The URI of the Docker image to use for the processing jobs.

`command` ([str]): The command to run, along with any command-line flags. Example: ["python3", "-v"].

`instance_count` (int): The number of instances to run a processing job with.

`instance_type` (str): The type of EC2 instance to use for processing, for example, 'ml.c4.xlarge'.

`volume_size_in_gb` (int): Size in GB of the EBS volume to use for storing data during processing (default: 30).

`volume_kms_key` (str): A KMS key for the processing volume (default: NULL).

`output_kms_key` (str): The KMS key ID for processing job outputs (default: NULL).

`max_runtime_in_seconds` (int): Timeout in seconds (default: NULL). After this amount of time, Amazon SageMaker terminates the job, regardless of its current status. If 'max_runtime_in_seconds' is not specified, the default value is 24 hours.

`base_job_name` (str): Prefix for processing name. If not specified, the processor generates a default job name, based on the processing image name and current timestamp.

`sagemaker_session` (:class:'~sagemaker.session.Session'): Session object which manages interactions with Amazon SageMaker and any other AWS services needed. If not specified, the processor creates one using the default AWS configuration chain.

`env` (dict[str, str]): Environment variables to be passed to the processing jobs (default: NULL).

`tags` (list[dict]): List of tags to be passed to the processing job (default: NULL). For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

`network_config` (:class:'~sagemaker.network.NetworkConfig'): A :class:'~sagemaker.network.NetworkConfig' object that configures network isolation, encryption of inter-container traffic, security group IDs, and subnets.

Method `get_run_args()`**:** Returns a RunArgs object. For processors (:class:‘~sagemaker.spark.processing.PySparkProce :class:‘~sagemaker.spark.processing.SparkJar‘) that have special run() arguments, this object contains the normalized arguments for passing to :class:‘~sagemaker.workflow.steps.ProcessingStep‘.

Usage:

```
ScriptProcessor$get_run_args(
  code,
  inputs = NULL,
  outputs = NULL,
  arguments = NULL
)
```

Arguments:

`code` (str): This can be an S3 URI or a local path to a file with the framework script to run.

`inputs` (list[:class:‘~sagemaker.processing.ProcessingInput‘]): Input files for the processing job. These must be provided as :class:‘~sagemaker.processing.ProcessingInput‘ objects (default: None).

`outputs` (list[:class:‘~sagemaker.processing.ProcessingOutput‘]): Outputs for the processing job. These can be specified as either path strings or :class:‘~sagemaker.processing.ProcessingOutput‘ objects (default: None).

`arguments` (list[str]): A list of string arguments to be passed to a processing job (default: None).

Method `run()`**:** Runs a processing job.

Usage:

```
ScriptProcessor$run(
  code,
  inputs = NULL,
  outputs = NULL,
  arguments = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL,
  experiment_config = NULL,
  kms_key = NULL
)
```

Arguments:

`code` (str): This can be an S3 URI or a local path to a file with the framework script to run.

`inputs` (list[:class:‘~sagemaker.processing.ProcessingInput‘]): Input files for the processing job. These must be provided as :class:‘~sagemaker.processing.ProcessingInput‘ objects (default: NULL).

`outputs` (list[:class:‘~sagemaker.processing.ProcessingOutput‘]): Outputs for the processing job. These can be specified as either path strings or :class:‘~sagemaker.processing.ProcessingOutput‘ objects (default: NULL).

`arguments` (list[str]): A list of string arguments to be passed to a processing job (default: NULL).

`wait` (bool): Whether the call should wait until the job completes (default: True).

`logs` (bool): Whether to show the logs produced by the job. Only meaningful when wait is True (default: True).

`job_name` (str): Processing job name. If not specified, the processor generates a default job name, based on the base job name and current timestamp.

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys: 'ExperimentName', 'TrialName', and 'TrialComponentDisplayName'.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the user code file (default: None).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ScriptProcessor$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Processor: `ProcessingInput`, `ProcessingJob`, `ProcessingOutput`, `Processor`

`ServerlessInferenceConfig` *Configuration object passed in when deploying models to Amazon SageMaker Endpoints.*

Description

This object specifies configuration related to serverless endpoint. Use this configuration when trying to create serverless endpoint and make serverless inference

Public fields

`memory_size_in_mb` The memory size of your serverless endpoint.

`max_concurrency` The maximum number of concurrent invocations your serverless endpoint can process

Methods

Public methods:

- `ServerlessInferenceConfig$new()`
- `ServerlessInferenceConfig$to_request_list()`
- `ServerlessInferenceConfig$format()`
- `ServerlessInferenceConfig$clone()`

Method `new()`**:** Initialize a ServerlessInferenceConfig object for serverless inference configuration.

Usage:

```
ServerlessInferenceConfig$new(memory_size_in_mb = 2048, max_concurrency = 5)
```

Arguments:

`memory_size_in_mb` (int): Optional. The memory size of your serverless endpoint. Valid values are in 1 GB increments: 1024 MB, 2048 MB, 3072 MB, 4096 MB, 5120 MB, or 6144 MB. If no value is provided, Amazon SageMaker will choose the default value for you. (Default: 2048)

`max_concurrency` (int): Optional. The maximum number of concurrent invocations your serverless endpoint can process. If no value is provided, Amazon SageMaker will choose the default value for you. (Default: 5)

Method `to_request_list()`**:** Generates a request dictionary using the parameters provided to the class.

Usage:

```
ServerlessInferenceConfig$to_request_list()
```

Method `format()`**:** Format class

Usage:

```
ServerlessInferenceConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ServerlessInferenceConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`SHAPConfig` *SHAPConfig Class*

Description

Config class of SHAP.

Super class

`sagemaker.common::ExplainabilityConfig` -> `SHAPConfig`

Public fields

`shap_config` Shap Config

Methods

Public methods:

- SHAPConfig$new()
- SHAPConfig$get_explainability_config()
- SHAPConfig$clone()

Method `new()`**:** Initializes config for SHAP.

Usage:

```
SHAPConfig$new(
  baseline,
  num_samples,
  agg_method = c("mean_abs", "median", "mean_sq"),
  use_logit = FALSE,
  save_local_shap_values = TRUE,
  seed = NULL,
  num_clusters = NULL,
  text_config = NULL,
  image_config = NULL
)
```

Arguments:

`baseline` (str or list): A list of rows (at least one) or S3 object URI to be used as the baseline dataset in the Kernel SHAP algorithm. The format should be the same as the dataset format. Each row should contain only the feature columns/values and omit the label column/values.

`num_samples` (int): Number of samples to be used in the Kernel SHAP algorithm. This number determines the size of the generated synthetic dataset to compute the SHAP values.

`agg_method` (str): Aggregation method for global SHAP values. Valid values are "mean_abs" (mean of absolute SHAP values for all instances), "median" (median of SHAP values for all instances) and "mean_sq" (mean of squared SHAP values for all instances).

`use_logit` (bool): Indicator of whether the logit function is to be applied to the model predictions. Default is False. If "use_logit" is true then the SHAP values will have log-odds units.

`save_local_shap_values` (bool): Indicator of whether to save the local SHAP values in the output location. Default is True.

`seed` (int): seed value to get deterministic SHAP values. Default is NULL.

`num_clusters` (NULL or int): If a baseline is not provided, Clarify automatically computes a baseline dataset via a clustering algorithm (K-means/K-prototypes). num_clusters is a parameter for this algorithm. num_clusters will be the resulting size of the baseline dataset. If not provided, Clarify job will use a default value.

`text_config` (:class:‘~sagemaker.clarify.TextConfig‘): Config to handle text features. Default is NULL

`image_config` (:class:‘~sagemaker.clarify.ImageConfig‘): Config to handle image features. Default is NULL

Method `get_explainability_config()`**:** Returns config.

Usage:

```
SHAPConfig$get_explainability_config()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SHAPConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`SMDataParallelProfilingConfig`
SMDataParallelProfilingConfig Class

Description

Configuration for framework metrics collected from a SageMaker Distributed training job.

Super class

`sagemaker.common::MetricsConfigBase` -> `SMDataParallelProfilingConfig`

Methods

Public methods:

- `SMDataParallelProfilingConfig$new()`
- `SMDataParallelProfilingConfig$clone()`

Method `new()`**:** Specify target steps or a target duration to profile. By default, it profiles step 15 of training. If `profile_default_steps` is set to 'True' and none of the other range parameters is specified, the class uses the default configuration for SageMaker Distributed profiling.

Usage:

```
SMDataParallelProfilingConfig$new(
  start_step = NULL,
  num_steps = NULL,
  start_unix_time = NULL,
  duration = NULL,
  profile_default_steps = FALSE
)
```

Arguments:

`start_step` (int): The step to start profiling. The default is step 15.

`num_steps` (int): The number of steps to profile. The default is for 1 steps.

`start_unix_time` (int): The Unix time to start profiling.

`duration` (float): The duration in seconds to profile.

`profile_default_steps` (bool): Indicates whether the default configuration should be used.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SMDataParallelProfilingConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`StepRange` *Configuration for the range of steps to profile.*

Description

It returns the target steps in dictionary format that you can pass to the :class:`~sagemaker.debugger.FrameworkProfile` class.

Methods

Public methods:

- `StepRange$new()`
- `StepRange$to_json()`
- `StepRange$format()`
- `StepRange$clone()`

Method `new()`**:** Set the start step and num steps. If the start step is not specified, Debugger starts profiling at step 0. If num steps is not specified, profile for 1 step.

Usage:

```
StepRange$new(start_step, num_steps)
```

Arguments:

`start_step` (int): The step to start profiling.

`num_steps` (int): The number of steps to profile.

Method `to_json()`**:** Convert the step range into a dictionary.

Usage:

```
StepRange$to_json()
```

Returns: list: The step range as a dictionary.

Method `format()`**:** format class

Usage:

```
StepRange$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
StepRange$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

TensorBoardOutputConfig

TensorBoardOutputConfig Class

Description

Create a tensor ouput configuration object for debugging visualizations on TensorBoard

Public fields

`s3_output_path` The location in Amazon S3 to store the output.

`container_local_output_path` The local path in the container.

Methods

Public methods:

- `TensorBoardOutputConfig$new()`
- `TensorBoardOutputConfig$to_request_list()`
- `TensorBoardOutputConfig$format()`
- `TensorBoardOutputConfig$clone()`

Method `new()`**:** Initialize the TensorBoardOutputConfig instance.

Usage:

```
TensorBoardOutputConfig$new(s3_output_path, container_local_output_path = NULL)
```

Arguments:

`s3_output_path` (str): Optional. The location in Amazon S3 to store the output.
`container_local_output_path` (str): Optional. The local path in the container.

Method `to_request_list()`**:** Generate a request dictionary using the instances attributes.

Usage:

```
TensorBoardOutputConfig$to_request_list()
```

Returns: dict: An portion of an API request as a dictionary.

Method `format()`**:** format class

Usage:

```
TensorBoardOutputConfig$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TensorBoardOutputConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TextConfig` *Config object to handle text features.*

Description

The SHAP analysis will break down longer text into chunks (e.g. tokens, sentences, or paragraphs) and replace them with the strings specified in the baseline for that feature. The shap value of a chunk then captures how much replacing it affects the prediction.

Super class

`sagemaker.common::ExplainabilityConfig` -> `TextConfig`

Public fields

`text_config` Text Config

Methods

Public methods:

- `TextConfig$new()`
- `TextConfig$get_text_config()`
- `TextConfig$clone()`

Method `new()`**:** Initializes a text configuration.

Usage:

```
TextConfig$new(granularity, language)
```

Arguments:

`granularity` (str): Determines the granularity in which text features are broken down to, can be "token", "sentence", or "paragraph". Shap values are computed for these units.

`language` (str): Specifies the language of the text features, can be "chinese", "danish", "dutch", "english", "french", "german", "greek", "italian", "japanese", "lithuanian", "multi-language", "norwegian bokmal", "polish", "portuguese", "romanian", "russian", "spanish", "afrikaans", "albanian", "arabic", "armenian", "basque", "bengali", "bulgarian", "catalan", "croatian", "czech", "estonian", "finnish", "gujarati", "hebrew", "hindi", "hungarian", "icelandic", "indonesian", "irish", "kannada", "kyrgyz", "latvian", "ligurian", "luxembourgish", "macedonian", "malayalam", "marathi", "nepali", "persian", "sanskrit", "serbian", "setswana", "sinhala", "slovak", "slovenian", "swedish", "tagalog", "tamil", "tatar", "telugu", "thai", "turkish", "ukrainian", "urdu", "vietnamese", "yoruba". Use "multi-language" for a mix of mulitple languages.

Method `get_text_config()`**:** Returns part of an analysis config dictionary.

Usage:

```
TextConfig$get_text_config()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TextConfig$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`TimeRange` *Configuration for the range of Unix time to profile.*

Description

It returns the target time duration in dictionary format that you can pass to the :class:'~sagemaker.debugger.FrameworkProfile' class.

Methods

Public methods:

- `TimeRange$new()`
- `TimeRange$to_json()`
- `TimeRange$format()`
- `TimeRange$clone()`

Method `new()`**:** Set the start Unix time and duration. If the start Unix time is not specified, profile starting at step 0. If the duration is not specified, profile for 1 step.

Usage:

```
TimeRange$new(start_unix_time = NULL, duration = NULL)
```

Arguments:

`start_unix_time` (int): The Unix time to start profiling.
`duration` (float): The duration in seconds to profile.

Method `to_json()`**:** Convert the time range into a dictionary.

Usage:

```
TimeRange$to_json()
```

Returns: dict: The time range as a dictionary.

Method `format()`**:** format class

Usage:

```
TimeRange$format()
```

Arguments:

`...` (ignored).

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TimeRange$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

TrainingJobAnalytics *TrainingJobAnalytics Class*

Description

Fetch training curve data from CloudWatch Metrics for a specific training job.

Super class

sagemaker.common::AnalyticsMetricsBase -> TrainingJobAnalytics

Public fields

CLOUDWATCH_NAMESPACE CloudWatch namespace to return Training Job Analytics data

Active bindings

name Name of the TrainingJob being analyzed

Methods

Public methods:

- TrainingJobAnalytics$new()
- TrainingJobAnalytics$clear_cache()
- TrainingJobAnalytics$clone()

Method new(): Initialize a "TrainingJobAnalytics" instance.

Usage:

```
TrainingJobAnalytics$new(
  training_job_name,
  metric_names = NULL,
  sagemaker_session = NULL,
  start_time = NULL,
  end_time = NULL,
  period = NULL
)
```

Arguments:

training_job_name (str): name of the TrainingJob to analyze.

metric_names (list, optional): string names of all the metrics to collect for this training job. If not specified, then it will use all metric names configured for this job.

sagemaker_session (sagemaker.session.Session): Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, one is specified using the default AWS configuration chain.

start_time :

end_time :

period :

Method `clear_cache()`**:** Clear the object of all local caches of API methods, so that the next time any properties are accessed they will be refreshed from the service.

Usage:

```
TrainingJobAnalytics$clear_cache()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TrainingJobAnalytics$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`Transformer` *Transformer class*

Description

A class for handling creating and interacting with Amazon SageMaker transform jobs

Methods

Public methods:

- `Transformer$new()`
- `Transformer$transform()`
- `Transformer$delete_model()`
- `Transformer$wait()`
- `Transformer$stop_transform_job()`
- `Transformer$attach()`
- `Transformer$format()`
- `Transformer$clone()`

Method `new()`**:** Initialize a "Transformer".

Usage:

```
Transformer$new(
  model_name,
  instance_count,
  instance_type,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  max_concurrent_transforms = NULL,
```

```
  max_payload = NULL,
  tags = NULL,
  env = NULL,
  base_transform_job_name = NULL,
  sagemaker_session = NULL,
  volume_kms_key = NULL
)
```

Arguments:

`model_name` (str): Name of the SageMaker model being used for the transform job.

`instance_count` (int): Number of EC2 instances to use.

`instance_type` (str): Type of EC2 instance to use, for example, 'ml.c4.xlarge'.

`strategy` (str): The strategy used to decide how to batch records in a single request (default: None). Valid values: 'MultiRecord' and 'SingleRecord'.

`assemble_with` (str): How the output is assembled (default: None). Valid values: 'Line' or 'None'.

`output_path` (str): S3 location for saving the transform result. If not specified, results are stored to a default bucket.

`output_kms_key` (str): Optional. KMS key ID for encrypting the transform output (default: None).

`accept` (str): The accept header passed by the client to the inference endpoint. If it is supported by the endpoint, it will be the format of the batch transform output.

`max_concurrent_transforms` (int): The maximum number of HTTP requests to be made to each individual transform container at one time.

`max_payload` (int): Maximum size of the payload in a single HTTP request to the container in MB.

`tags` (list[dict]): List of tags for labeling a transform job (default: None). For more, see the SageMaker API documentation for 'Tag <https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html>'_.

`env` (dict): Environment variables to be set for use during the transform job (default: None).

`base_transform_job_name` (str): Prefix for the transform job when the :meth:'~sagemaker.transformer.Transformer.tran
method launches. If not specified, a default prefix will be generated based on the training image name that was used to train the model associated with the transform job.

`sagemaker_session` (sagemaker.session.Session): Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.

`volume_kms_key` (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

Method `transform()`**:** Start a new transform job.

Usage:

```
Transformer$transform(
  data,
  data_type = "S3Prefix",
  content_type = NULL,
  compression_type = NULL,
  split_type = NULL,
  job_name = NULL,
```

```
  input_filter = NULL,
  output_filter = NULL,
  join_source = NULL,
  experiment_config = NULL,
  model_client_config = NULL,
  wait = TRUE,
  logs = TRUE,
  ...
)
```

Arguments:

`data` (str): Input data location in S3.

`data_type` (str): What the S3 location defines (default: 'S3Prefix'). Valid values:

- **'S3Prefix'** - the S3 URI defines a key name prefix. All objects with this prefix will be used as inputs for the transform job.
- **'ManifestFile'** - the S3 URI points to a single manifest file listing each S3 object to use as an input for the transform job.

`content_type` (str): MIME type of the input data (default: None).

`compression_type` (str): Compression type of the input data, if compressed (default: None). Valid values: 'Gzip', None.

`split_type` (str): The record delimiter for the input object (default: 'None'). Valid values: 'None', 'Line', 'RecordIO', and 'TFRecord'.

`job_name` (str): job name (default: None). If not specified, one will be generated.

`input_filter` (str): A JSONPath to select a portion of the input to pass to the algorithm container for inference. If you omit the field, it gets the value '$', representing the entire input. For CSV data, each row is taken as a JSON array, so only index-based JSONPaths can be applied, e.g. $[0], $[1:]. CSV data should follow the 'RFC format <https://tools.ietf.org/html/rfc4180>'_. See 'Supported JSONPath Operators <https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform-data-processing.html#data-processing-operators>'_ for a table of supported JSONPath operators. For more information, see the SageMaker API documentation for 'CreateTransformJob <https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateTransformJob.html>'_. Some examples: "$[1:]", "$.features" (default: None).

`output_filter` (str): A JSONPath to select a portion of the joined/original output to return as the output. For more information, see the SageMaker API documentation for 'CreateTransformJob <https://docs.aws.amazon.com/sagemaker/latest/dg/API_CreateTransformJob.html>'_. Some examples: "$[1:]", "$.prediction" (default: None).

`join_source` (str): The source of data to be joined to the transform output. It can be set to 'Input' meaning the entire input record will be joined to the inference result. You can use OutputFilter to select the useful portion before uploading to S3. (default: None). Valid values: Input, None.

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys, 'ExperimentName', 'TrialName', and 'TrialComponentDisplayName'. (default: "None").

`model_client_config` (dict[str, str]): Model configuration. Dictionary contains two optional keys, 'InvocationsTimeoutInSeconds', and 'InvocationsMaxRetries'. (default: "None").

`wait` (bool): Whether the call should wait until the job completes (default: TRUE).

`logs` (bool): Whether to show the logs produced by the job. Only meaningful when wait is True (default: TRUE).

... Other parameters (currently not used)

Returns: NULL invisible

Method `delete_model()`**:** Delete the corresponding SageMaker model for this Transformer.

Usage:

```
Transformer$delete_model()
```

Method `wait()`**:** Wait for latest running batch transform job

Usage:

```
Transformer$wait(logs = TRUE)
```

Arguments:

`logs` return logs

Method `stop_transform_job()`**:** Stop latest running batch transform job.

Usage:

```
Transformer$stop_transform_job(wait = TRUE)
```

Arguments:

`wait` wait for transform job

Method `attach()`**:** Attach an existing transform job to a new Transformer instance

Usage:

```
Transformer$attach(transform_job_name, sagemaker_session = NULL)
```

Arguments:

`transform_job_name` (str): Name for the transform job to be attached.

`sagemaker_session` (sagemaker.session.Session): Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, one will be created using the default AWS configuration chain.

Returns: Transformer (class): The Transformer instance with the specified transform job attached.

Method `format()`**:** format class

Usage:

```
Transformer$format()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Transformer$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index