Package: sagemaker.mlcore (via r-universe)

July 14, 2024

Type Package

Title sagemaker machine learning core classes and methods

Version 0.3.0

Description `sagemaker` machine learning core classes and methods.

Imports lgr, R6, fs, sagemaker.core, sagemaker.common, stats, data.table, jsonlite, methods, urltools, uuid

Suggests crayon, Matrix, readsparse, readr, RProtoBuf, reticulate, testthat, mockthat, tibble

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

License Apache License (>= 2.0)

Encoding UTF-8

RoxygenNote 7.1.2

Collate 'r_utils.R' 'deserializers.R' 'serializers.R' 'model_monitor_model_monitoring.R' 'predictor.R' 'model.R' 'estimator.R' 'parameter.R' 'algorithm.R' 'amazon_common.R' 'amazon_validation.R' 'amazon_hyperparameter.R' 'amazon_estimator.R' 'model_metrics.R' 'model_monitor_clarify_model_monitoring.R' 'model_monitor_cron_expression_generator.R' 'model_monitor_data_capture_config.R' 'model_monitor_dataset_format.R' 'model_monitor_monitoring_files.R' 'multidatamodel.R' 'pipeline.R' 'serverless_model.R' 'serverless_predictor.R' 'training_compiler_config.R' 'tuner.R' 'zzz.R'

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

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

RemoteRef HEAD

RemoteSha bee50eefe641d4699c47e862ca75870f6f899069

Contents

`sagemaker.mlcore-package`

r6 sagemaker: this is just a placeholder

Description

'sagemaker' machine learning core classes and methods.

Author(s)

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

Other contributors:

- Amazon.com, Inc. [copyright holder]

`AlgorithmEstimator` *AlgorithmEstimator Class*

Description

A generic Estimator to train using any algorithm object (with an "algorithm_arn"). The Algorithm can be your own, or any Algorithm from AWS Marketplace that you have a valid subscription for. This class will perform client-side validation on all the inputs.

Super class

`sagemaker.mlcore::EstimatorBase` -> `AlgorithmEstimator`

Public fields

`.hyperparameters_with_range` These Hyperparameter Types have a range definition.

Methods

Public methods:

- `AlgorithmEstimator$new()`
- `AlgorithmEstimator$validate_train_spec()`
- `AlgorithmEstimator$set_hyperparameter()`
- `AlgorithmEstimator$hyperparameters()`
- `AlgorithmEstimator$training_image_uri()`
- `AlgorithmEstimator$enable_network_isolation()`
- `AlgorithmEstimator$create_model()`
- `AlgorithmEstimator$transformer()`
- `AlgorithmEstimator$fit()`
- `AlgorithmEstimator$print()`
- `AlgorithmEstimator$clone()`

Method `new()`: Initialize an "AlgorithmEstimator" instance.

Usage:

```
AlgorithmEstimator$new(
  algorithm_arn,
  role,
  instance_count,
  instance_type,
  volume_size = 30,
  volume_kms_key = NULL,
  max_run = 24 * 60 * 60,
  input_mode = "File",
  output_path = NULL,
  output_kms_key = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  hyperparameters = NULL,
  tags = NULL,
  subnets = NULL,
  security_group_ids = NULL,
  model_uri = NULL,
  model_channel_name = "model",
  metric_definitions = NULL,
  encrypt_inter_container_traffic = FALSE,
  ...
)
```

Arguments:

`algorithm_arn` (str): algorithm arn used for training. Can be just the name if your account owns the algorithm.

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.

`instance_count` (int): Number of Amazon EC2 instances to use for training.

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

`volume_size` (int): Size in GB of the EBS volume to use for storing input data during training (default: 30). Must be large enough to store training data if File Mode is used (which is the default).

`volume_kms_key` (str): Optional. KMS key ID for encrypting EBS volume attached to the training instance (default: NULL).

`max_run` (int): Timeout in seconds for training (default: 24 * 60 * 60). After this amount of time Amazon SageMaker terminates the job regardless of its current status.

`input_mode` (str): The input mode that the algorithm supports (default: 'File'). Valid modes: * 'File' - Amazon SageMaker copies the training dataset from the S3 location to a local directory. * 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a Unix-named pipe. This argument can be overriden on a per-channel basis using "TrainingInput.input_mode".

`output_path` (str): S3 location for saving the training result (model artifacts and output files). If not specified, results are stored to a default bucket. If the bucket with the specific name does not exist, the estimator creates the bucket during the :meth:'~sagemaker.estimator.EstimatorBase.fit' method execution.

`output_kms_key` (str): Optional. KMS key ID for encrypting the training output (default: NULL).

`base_job_name` (str): Prefix for training job name when the :meth:'~sagemaker.estimator.EstimatorBase.fit' method launches. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

`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.

`hyperparameters` (dict): Dictionary containing the hyperparameters to initialize this estimator with.

`tags` (list[dict]): List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/

`subnets` (list[str]): List of subnet ids. If not specified training job will be created without VPC config.

`security_group_ids` (list[str]): List of security group ids. If not specified training job will be created without VPC config.

`model_uri` (str): URI where a pre-trained model is stored, either locally or in S3 (default: NULL). If specified, the estimator will create a channel pointing to the model so the training job can download it. This model can be a 'model.tar.gz' from a previous training job, or other artifacts coming from a different source. More information: https://docs.aws.amazon.com/sagemaker/latest/dg/ training.html#td-deserialization

`model_channel_name` (str): Name of the channel where 'model_uri' will be downloaded (default: 'model').

`metric_definitions` (list[dict]): A list of dictionaries that defines the metric(s) used to evaluate the training jobs. Each dictionary contains two keys: 'Name' for the name of the metric, and 'Regex' for the regular expression used to extract the metric from the logs.

`encrypt_inter_container_traffic` (bool): Specifies whether traffic between training containers is encrypted for the training job (default: "False").

`...` : Additional kwargs. This is unused. It's only added for AlgorithmEstimator to ignore the irrelevant arguments.

Method `validate_train_spec()`**:** Placeholder docstring

Usage:

```
AlgorithmEstimator$validate_train_spec()
```

Method `set_hyperparameter()`**:** formats hyperparameters for model tunning

Usage:

```
AlgorithmEstimator$set_hyperparameter(...)
```

Arguments:

`...` model hyperparameters

Method `hyperparameters()`**:** Returns the hyperparameters as a dictionary to use for training. The fit() method, that does the model training, calls this method to find the hyperparameters you specified.

Usage:

```
AlgorithmEstimator$hyperparameters()
```

Method `training_image_uri()`**:** Returns the docker image to use for training. The fit() method, that does the model training, calls this method to find the image to use for model training.

Usage:

```
AlgorithmEstimator$training_image_uri()
```

Method `enable_network_isolation()`**:** Return True if this Estimator will need network isolation to run. On Algorithm Estimators this depends on the algorithm being used. If this is algorithm owned by your account it will be False. If this is an an algorithm consumed from Marketplace it will be True.

Usage:

```
AlgorithmEstimator$enable_network_isolation()
```

Returns: bool: Whether this Estimator needs network isolation or not.

Method `create_model()`**:** Create a model to deploy. The serializer, deserializer, content_type, and accept arguments are only used to define a default Predictor They are ignored if an explicit predictor class is passed in. Other arguments are passed through to the Model class.

Usage:

```
AlgorithmEstimator$create_model(
  role = NULL,
  predictor_cls = NULL,
  serializer = IdentitySerializer$new(),
  deserializer = BytesDeserializer$new(),
  vpc_config_override = "VPC_CONFIG_DEFAULT",
  ...
)
```

Arguments:

`role` (str): The “ExecutionRoleArn“ IAM Role ARN for the “Model“, which is also used during transform jobs. If not specified, the role from the Estimator will be used.

`predictor_cls` (RealTimePredictor): The predictor class to use when deploying the model.

`serializer` (callable): Should accept a single argument, the input data, and return a sequence of bytes. May provide a content_type attribute that defines the endpoint request content type

`deserializer` (callable): Should accept two arguments, the result data and the response content type, and return a sequence of bytes. May provide a content_type attribute that defines the endpoint response Accept content type.

`vpc_config_override` (dict[str, list[str]]): Optional override for VpcConfig set on the model. Default: use subnets and security groups from this Estimator. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

`...` : Additional arguments for creating a :class:‘~sagemaker.model.ModelPackage‘. .. tip:: You can find additional parameters for using this method at :class:‘~sagemaker.model.ModelPackage‘ and :class:‘~sagemaker.model.Model‘.

Returns: a Model ready for deployment.

Method `transformer()`**:** Return a “Transformer“ that uses a SageMaker Model based on the training job. It reuses the SageMaker Session and base job name used by the Estimator.

Usage:

```
AlgorithmEstimator$transformer(
  instance_count,
  instance_type,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  env = NULL,
  max_concurrent_transforms = NULL,
  max_payload = NULL,
  tags = NULL,
  role = NULL,
  volume_kms_key = NULL
)
```

Arguments:

`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.

`env` (dict): Environment variables to be set for use during the transform job (default: None).

`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. If none specified, then the tags used for the training job are used for the transform job.

`role` (str): The "ExecutionRoleArn" IAM Role ARN for the "Model", which is also used during transform jobs. If not specified, the role from the Estimator will be used.

`volume_kms_key` (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

Method `fit()`**:** Train a model using the input training dataset. The API calls the Amazon SageMaker CreateTrainingJob API to start model training. The API uses configuration you provided to create the estimator and the specified input training data to send the CreatingTrainingJob request to Amazon SageMaker. This is a synchronous operation. After the model training successfully completes, you can call the "deploy()" method to host the model using the Amazon SageMaker hosting services.

Usage:

```
AlgorithmEstimator$fit(
  inputs = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL
)
```

Arguments:

`inputs` (str or dict or TrainingInput): Information about the training data. This can be one of three types:

- **(str)** the S3 location where training data is saved, or a file:// path in local mode.
- **(dict[str, str]** or dict[str, TrainingInput]) If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:‘~TrainingInput‘ objects.
- **(TrainingInput)** - channel configuration for S3 data sources that can provide additional information as well as the path to the training dataset. See :func:‘TrainingInput‘ for full details.
- **(sagemaker.session.FileSystemInput)** - channel configuration for a file system data source that can provide additional information as well as the path to the training dataset.

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

`logs` ([str]): A list of strings specifying which logs to print. Acceptable strings are "All", "NULL", "Training", or "Rules". To maintain backwards compatibility, boolean values are also accepted and converted to strings. Only meaningful when wait is True.

`job_name` (str): Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

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

Method `print()`**:** Printer.

Usage:

```
AlgorithmEstimator$print(...)
```

Arguments:

`...` (ignored).

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`AmazonAlgorithmEstimatorBase`

AmazonAlgorithmEstimatorBase Class

Description

Base class for Amazon first-party Estimator implementations. This class isn’t intended to be instantiated directly.

Super class

`sagemaker.mlcore::EstimatorBase` -> `AmazonAlgorithmEstimatorBase`

Public fields

`repo_name` The repo name for the account

`repo_version` Version fo repo to call

`.module` mimic python module

Active bindings

`data_location` The s3 prefix to upload RecordSet objects to, expressed as an S3 url

`feature_dim` Hyperparameter class for feature_dim

`mini_batch_size` Hyperparameter class for mini_batch_size

Methods

Public methods:

- `AmazonAlgorithmEstimatorBase$new()`
- `AmazonAlgorithmEstimatorBase$training_image_uri()`
- `AmazonAlgorithmEstimatorBase$hyperparameters()`
- `AmazonAlgorithmEstimatorBase$prepare_workflow_for_training()`
- `AmazonAlgorithmEstimatorBase$fit()`
- `AmazonAlgorithmEstimatorBase$record_set()`
- `AmazonAlgorithmEstimatorBase$wait()`
- `AmazonAlgorithmEstimatorBase$clone()`

Method `new()`**:** Initialize an AmazonAlgorithmEstimatorBase.

Usage:

```
AmazonAlgorithmEstimatorBase$new(
  role,
  instance_count,
  instance_type,
  data_location = NULL,
  enable_network_isolation = FALSE,
  ...
)
```

Arguments:

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.

`instance_count` (int): Number of Amazon EC2 instances to use for training.

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

`data_location` (str or None): The s3 prefix to upload RecordSet objects to, expressed as an S3 url. For example "s3://example-bucket/some-key-prefix/". Objects will be saved in a unique sub-directory of the specified location. If None, a default data location will be used.

`enable_network_isolation` (bool): Specifies whether container will run in network isolation mode. Network isolation mode restricts the container access to outside networks (such as the internet). Also known as internet-free mode (default: “False“).

`...` : Additional parameters passed to :class:‘~sagemaker.estimator.EstimatorBase‘.

Method `training_image_uri()`**:** Return algorithm image URI for the given AWS region, repository name, and repository version

Usage:

```
AmazonAlgorithmEstimatorBase$training_image_uri()
```

Method `hyperparameters()`**:** Return all non-None “hyperparameter“ values on “obj“ as a “dict[str,str].“

Usage:

```
AmazonAlgorithmEstimatorBase$hyperparameters()
```

Method `prepare_workflow_for_training()`**:** Calls _prepare_for_training. Used when setting up a workflow.

Usage:

```
AmazonAlgorithmEstimatorBase$prepare_workflow_for_training(
  records = NULL,
  mini_batch_size = NULL,
  job_name = NULL
)
```

Arguments:

`records` (:class:‘~RecordSet‘): The records to train this “Estimator“ on.

`mini_batch_size` (int or None): The size of each mini-batch to use when training. If “None“, a default value will be used.

`job_name` (str): Name of the training job to be created. If not specified, one is generated, using the base name given to the constructor if applicable.

Method `fit()`**:** Fit this Estimator on serialized Record objects, stored in S3. “records“ should be an instance of :class:‘~RecordSet‘. This defines a collection of S3 data files to train this “Estimator“ on. Training data is expected to be encoded as dense or sparse vectors in the "values" feature on each Record. If the data is labeled, the label is expected to be encoded as a list of scalas in the "values" feature of the Record label. More information on the Amazon Record format is available at: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html See :meth:‘~AmazonAlgorithmEstimatorBase.record_set‘ to construct a “RecordSet“ object from :class:‘~numpy.ndarray‘ arrays.

Usage:

```
AmazonAlgorithmEstimatorBase$fit(
  records,
  mini_batch_size = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL,
  experiment_config = NULL
)
```

Arguments:

`records` (:class:‘~RecordSet‘): The records to train this “Estimator“ on

`mini_batch_size` (int or None): The size of each mini-batch to use when training. If “None“, a default value will be used.

`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): Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

`experiment_config` (dict[str, str]): Experiment management configuration. Dictionary contains three optional keys, ’ExperimentName’, ’TrialName’, and ’TrialComponentName’ (default: “None“).

Method `record_set()`**:** Build a :class:‘~RecordSet‘ from a numpy :class:‘~ndarray‘ matrix and label vector. For the 2D “ndarray“ “train“, each row is converted to a :class:‘~Record‘ object. The vector is stored in the "values" entry of the “features“ property of each Record. If “labels“ is not None, each corresponding label is assigned to the "values" entry of the “labels“ property of each Record. The collection of “Record“ objects are protobuf serialized and uploaded to new S3 locations. A manifest file is generated containing the list of objects created and also stored in S3. The number of S3 objects created is controlled by the “instance_count“ property on this Estimator. One S3 object is created per training instance.

Usage:

```
AmazonAlgorithmEstimatorBase$record_set(
  train,
  labels = NULL,
  channel = "train",
  encrypt = FALSE
)
```

Arguments:

`train` (numpy.ndarray): A 2D numpy array of training data.

`labels` (numpy.ndarray): A 1D numpy array of labels. Its length must be equal to the number of rows in “train“.

`channel` (str): The SageMaker TrainingJob channel this RecordSet should be assigned to.

`encrypt` (bool): Specifies whether the objects uploaded to S3 are encrypted on the server side using AES-256 (default: “False“).

Returns: RecordSet: A RecordSet referencing the encoded, uploading training and label data.

Method `wait()`**:** Wait for an Amazon SageMaker job to complete.

Usage:

```
AmazonAlgorithmEstimatorBase$wait(logs = "All")
```

Arguments:

`logs` ([str]): A list of strings specifying which logs to print. Acceptable strings are "All", "NULL", "Training", or "Rules". To maintain backwards compatibility, boolean values are also accepted and converted to strings.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`BaseDeserializer` *Default BaseDeserializer Class*

Description

All BaseDeserializer are children of this class. If a custom BaseDeserializer is desired, inherit this class.

Active bindings

`ACCEPT` The content types that are expected from the inference endpoint.

Methods

Public methods:

- `BaseDeserializer$deserialize()`
- `BaseDeserializer$format()`
- `BaseDeserializer$clone()`

Method `deserialize()`**:** Deserialize data received from an inference endpoint.

Usage:

```
BaseDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` (botocore.response.StreamingBody): Data to be deserialized.

`content_type` (str): The MIME type of the data.

Returns: object: The data deserialized into an object.

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

Usage:

```
BaseDeserializer$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: BaseSerializer, BytesDeserializer, CSVDeserializer, CSVSerializer, DataTableDeserializer, IdentitySerializer, JSONDeserializer, JSONLinesDeserializer, JSONLinesSerializer, JSONSerializer, LibSVMSerializer, NumpyDeserializer, NumpySerializer, SimpleBaseDeserializer, SimpleBaseSerializer, SparseMatrixSerializer, StringDeserializer, TibbleDeserializer

---

`BaseliningJob` *Baselining Job Class*

Description

Provides functionality to retrieve baseline-specific files output from baselining job.

Super classes

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

Methods

Public methods:

- `BaseliningJob$new()`
- `BaseliningJob$from_processing_job()`
- `BaseliningJob$baseline_statistics()`
- `BaseliningJob$suggested_constraints()`
- `BaseliningJob$clone()`

Method `new()`**:** Initializes a Baselining job that tracks a baselining job kicked off by the suggest workflow.

Usage:

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

Arguments:

`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.

`job_name` (str): Name of the Amazon SageMaker Model Monitoring Baselining Job.

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

`outputs` ([sagemaker.processing.ProcessingOutput]): A list of ProcessingOutput objects.

`output_kms_key` (str): The output kms key associated with the job. Defaults to None if not provided.

Method `from_processing_job()`**:** Initializes a Baselining job from a processing job.

Usage:

```
BaseliningJob$from_processing_job(processing_job)
```

Arguments:

`processing_job` (sagemaker.processing.ProcessingJob): The ProcessingJob used for baselining instance.

Returns: sagemaker.processing.BaseliningJob: The instance of ProcessingJob created using the current job name.

Method `baseline_statistics()`**:** Returns a sagemaker.model_monitor.Statistics object representing the statistics JSON file generated by this baselining job.

Usage:

```
BaseliningJob$baseline_statistics(
  file_name = STATISTICS_JSON_DEFAULT_FILE_NAME,
  kms_key = NULL
)
```

Arguments:

`file_name` (str): The name of the json-formatted statistics file

`kms_key` (str): The kms key to use when retrieving the file.

Returns: sagemaker.model_monitor.Statistics: The Statistics object representing the file that was generated by the job.

Method `suggested_constraints()`**:** Returns a sagemaker.model_monitor.Constraints object representing the constraints JSON file generated by this baselining job.

Usage:

```
BaseliningJob$suggested_constraints(
  file_name = CONSTRAINTS_JSON_DEFAULT_FILE_NAME,
  kms_key = NULL
)
```

Arguments:

`file_name` (str): The name of the json-formatted constraints file

`kms_key` (str): The kms key to use when retrieving the file.

Returns: sagemaker.model_monitor.Constraints: The Constraints object representing the file that was generated by the job.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

BaseSerializer — *Default BaseSerializer Class*

Description

All serializer are children of this class. If a custom serializer is desired, inherit this class.

Active bindings

`CONTENT_TYPE` The MIME type of the data sent to the inference endpoint.

Methods

Public methods:

- `BaseSerializer$serialize()`
- `BaseSerializer$format()`
- `BaseSerializer$clone()`

Method `serialize()`**:** Take data of various data formats and serialize them into CSV.

Usage:

```
BaseSerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized

Returns: object: Serialized data used for a request.

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

Usage:

```
BaseSerializer$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`BiasAnalysisConfig` *BiasAnalysisConfig class*

Description

Analysis configuration for ModelBiasMonitor.

Public fields

`analysis_config` Analysis config dictionary

Methods

Public methods:

- `BiasAnalysisConfig$new()`
- `BiasAnalysisConfig$to_list()`
- `BiasAnalysisConfig$format()`
- `BiasAnalysisConfig$clone()`

Method `new()`**:** Creates an analysis config dictionary.

Usage:

```
BiasAnalysisConfig$new(bias_config, headers = NULL, label = NULL)
```

Arguments:

`bias_config` (sagemaker.clarify.BiasConfig): Config object related to bias configurations.

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

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

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

Usage:

```
BiasAnalysisConfig$to_list()
```

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

Usage:

```
BiasAnalysisConfig$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`BytesDeserializer` *BytesDerializer Class*

Description

Deserialize a stream of bytes into a bytes object.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `BytesDeserializer`

Methods

Public methods:

- `BytesDeserializer$deserialize()`
- `BytesDeserializer$clone()`

Method `deserialize()`**:** Read a stream of bytes returned from an inference endpoint.

Usage:

```
BytesDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` (raw): A stream of bytes.

`content_type` (str): The MIME type of the data.

Returns: bytes: The bytes object read from the stream.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`CategoricalParameter` *CategoricalParameter Class*

Description

A class for representing hyperparameters that have a discrete list of possible values.

Super class

`sagemaker.mlcore::ParameterRange` -> `CategoricalParameter`

Public fields

`.name` Helps to categorise Class

`values` The possible values for the hyperparameter

Methods

Public methods:

- `CategoricalParameter$new()`
- `CategoricalParameter$as_tuning_range()`
- `CategoricalParameter$as_json_range()`
- `CategoricalParameter$is_valid()`
- `CategoricalParameter$cast_to_type()`
- `CategoricalParameter$clone()`

Method `new()`**:** Initialize a “CategoricalParameter“.

Usage:

```
CategoricalParameter$new(values)
```

Arguments:

`values` (list or object): The possible values for the hyperparameter. This input will be converted into a list of strings.

Method `as_tuning_range()`**:** Represent the parameter range as a dicionary suitable for a request to create an Amazon SageMaker hyperparameter tuning job.

Usage:

```
CategoricalParameter$as_tuning_range(name)
```

Arguments:

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

Returns: dict[str, list[str]]: A dictionary that contains the name and values of the hyperparameter.

Method `as_json_range()`**:** Represent the parameter range as a dictionary suitable for a request to create an Amazon SageMaker hyperparameter tuning job using one of the deep learning frameworks. The deep learning framework images require that hyperparameters be serialized as JSON.

Usage:

```
CategoricalParameter$as_json_range(name)
```

Arguments:

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

Returns: dict[str, list[str]]: A dictionary that contains the name and values of the hyperparameter, where the values are serialized as JSON.

Method `is_valid()`**:** Determine if a value is valid within this CategoricalParameter

Usage:

```
CategoricalParameter$is_valid(value)
```

Arguments:

`value` (object): Value of the hyperparameter

Returns: boolean: TRUE‘ or ‘FALSE‘

Method `cast_to_type()`**:** cast value to numeric

Usage:

```
CategoricalParameter$cast_to_type(value)
```

Arguments:

`value` The value to be verified.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ClarifyBaseliningConfig` *ClarifyBaseliningConfig class*

Description

Data class to hold some essential analysis configuration of ClarifyBaseliningJob

Public fields

`analysis_config` analysis config from configurations of the baselining job.

`features_attribute` JSONpath to locate features in predictor request payload. Only required when predictor content type is JSONlines.

`inference_attribute` Index, header or JSONpath to locate predicted label in predictor response payload

`probability_attribute` Index or JSONpath location in the model output for probabilities or scores to be used for explainability.

`probability_threshold_attribute` Value to indicate the threshold to select the binary label in the case of binary classification. Default is 0.5.

Methods

Public methods:

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

Method `new()`**:** Initialization.

Usage:

```
ClarifyBaseliningConfig$new(
  analysis_config,
  features_attribute = NULL,
  inference_attribute = NULL,
  probability_attribute = NULL,
  probability_threshold_attribute = NULL
)
```

Arguments:

`analysis_config` (BiasAnalysisConfig or ExplainabilityAnalysisConfig): analysis config from configurations of the baselining job.

`features_attribute` (str): JSONpath to locate features in predictor request payload. Only required when predictor content type is JSONlines.

`inference_attribute` (str): Index, header or JSONpath to locate predicted label in predictor response payload.

`probability_attribute` (str): Index or JSONpath location in the model output for probabilities or scores to be used for explainability.

`probability_threshold_attribute` (float): Value to indicate the threshold to select the binary label in the case of binary classification. Default is 0.5.

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

Usage:

```
ClarifyBaseliningConfig$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ClarifyBaseliningJob` *ClarifyBaseliningJob class*

Description

Provides functionality to retrieve baseline-specific output from Clarify baselining job.

Super classes

`sagemaker.common::.Job` -> `sagemaker.common::ProcessingJob` -> `sagemaker.mlcore::BaseliningJob` -> `ClarifyBaseliningJob`

Methods

Public methods:

- `ClarifyBaseliningJob$new()`
- `ClarifyBaseliningJob$baseline_statistics()`
- `ClarifyBaseliningJob$suggested_constraints()`
- `ClarifyBaseliningJob$clone()`

Method `new()`**:** Initializes a ClarifyBaseliningJob that tracks a baselining job by suggest_baseline()

Usage:

```
ClarifyBaseliningJob$new(processing_job)
```

Arguments:

`processing_job` (sagemaker.processing.ProcessingJob): The ProcessingJob used for baselining instance.

Method `baseline_statistics()`**:** Not implemented. The class doesn't support statistics.

Usage:

```
ClarifyBaseliningJob$baseline_statistics()
```

Method `suggested_constraints()`**:** Returns a sagemaker.model_monitor. Constraints object representing the constraints JSON file generated by this baselining job.

Usage:

```
ClarifyBaseliningJob$suggested_constraints(file_name = NULL, kms_key = NULL)
```

Arguments:

`file_name` (str): Keep this parameter to align with method signature in super class, but it will be ignored.

`kms_key` (str): The kms key to use when retrieving the file.

Returns: sagemaker.model_monitor.Constraints: The Constraints object representing the file that was generated by the job.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ClarifyModelMonitor` *Base class of Amazon SageMaker Explainability API model monitors.*

Description

This class is an “abstract base class“, please instantiate its subclasses if you want to monitor bias metrics or feature attribution of an endpoint.

Super class

`sagemaker.mlcore::ModelMonitor` -> `ClarifyModelMonitor`

Methods

Public methods:

- `ClarifyModelMonitor$new()`
- `ClarifyModelMonitor$run_baseline()`
- `ClarifyModelMonitor$latest_monitorying_statistics()`
- `ClarifyModelMonitor$list_executions()`
- `ClarifyModelMonitor$clone()`

Method `new()`**:** Initializes a monitor instance. The monitor handles baselining datasets and creating Amazon SageMaker Monitoring Schedules to monitor SageMaker endpoints.

Usage:

```
ClarifyModelMonitor$new(
  role,
  instance_count = 1,
  instance_type = "ml.m5.xlarge",
  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. The Amazon SageMaker jobs use this role.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

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

`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.

`env` (dict): Environment variables to be passed to the job.

`tags` ([dict]): List of tags to be passed to the job.

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

Method `run_baseline()`**:** Not implemented. .run_baseline()' is only allowed for ModelMonitor objects. Please use ‘suggest_baseline‘ instead.

Usage:

```
ClarifyModelMonitor$run_baseline(...)
```

Arguments:

`...` : Unused argument

Method `latest_monitorying_statistics()`**:** Not implemented. The class doesn't support statistics.

Usage:

```
ClarifyModelMonitor$latest_monitorying_statistics(...)
```

Arguments:

`...` : Unused argument

Method `list_executions()`**:** Get the list of the latest monitoring executions in descending order of "ScheduledTime".

Usage:

```
ClarifyModelMonitor$list_executions()
```

Returns: [sagemaker.model_monitor.ClarifyMonitoringExecution]: List of ClarifyMonitoringExecution in descending order of "ScheduledTime".

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ClarifyMonitoringExecution` *ClarifyMonitoringExecution class*

Description

Provides functionality to retrieve monitoring-specific files output from executions.

Super classes

`sagemaker.common::.Job` -> `sagemaker.common::ProcessingJob` -> `sagemaker.mlcore::MonitoringExecution` -> `ClarifyMonitoringExecution`

Methods

Public methods:

- `ClarifyMonitoringExecution$new()`
- `ClarifyMonitoringExecution$statistics()`
- `ClarifyMonitoringExecution$clone()`

Method `new()`**:** Initializes an object that tracks a monitoring execution by a Clarify model monitor

Usage:

```
ClarifyMonitoringExecution$new(
  sagemaker_session,
  job_name,
  inputs,
  output,
  output_kms_key = NULL
)
```

Arguments:

`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.

`job_name` (str): The name of the monitoring execution job.

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

`output` (sagemaker.Processing.ProcessingOutput): The output associated with the monitoring execution.

`output_kms_key` (str): The output kms key associated with the job. Defaults to None if not provided.

Method `statistics()`**:** Not implemented. The class doesn't support statistics.

Usage:

```
ClarifyMonitoringExecution$statistics()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Constraints` *Constraints Class*

Description

Represents the constraints JSON file used in Amazon SageMaker Model Monitoring.

Super class

`sagemaker.mlcore::ModelMonitoringFile` -> `Constraints`

Methods

Public methods:

- `Constraints$new()`
- `Constraints$from_s3_uri()`
- `Constraints$from_string()`
- `Constraints$from_file_path()`
- `Constraints$set_monitoring()`
- `Constraints$clone()`

Method `new()`**:** Initializes the Constraints object used in Amazon SageMaker Model Monitoring.

Usage:

```
Constraints$new(
  body_dict = NULL,
  constraints_file_s3_uri = NULL,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`body_dict` (str): The body of the constraints JSON file.

`constraints_file_s3_uri` (str): The uri of the constraints JSON file.

`kms_key` (str): The kms key to be used to decrypt the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Method `from_s3_uri()`**:** Generates a Constraints object from an s3 uri.

Usage:

```
Constraints$from_s3_uri(
  constraints_file_s3_uri,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`constraints_file_s3_uri` (str): The uri of the constraints JSON file.

`kms_key` (str): The kms key to be used to decrypt the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.Constraints: The instance of Constraints generated from the s3 uri.

Method `from_string()`**:** Generates a Constraints object from an s3 uri.

Usage:

```
Constraints$from_string(
  constraints_file_string,
  kms_key = NULL,
  file_name = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`constraints_file_string` (str): The uri of the constraints JSON file.

`kms_key` (str): The kms key to be used to encrypt the file in S3.

`file_name` (str): The file name to use when uploading to S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.Constraints: The instance of Constraints generated from the s3 uri.

Method `from_file_path()`**:** Initializes a Constraints object from a file path.

Usage:

```
Constraints$from_file_path(
  constraints_file_path,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`constraints_file_path` (str): The path to the constraints file.

`kms_key` (str): The kms_key to use when encrypting the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.Constraints: The instance of Constraints generated from the local file path.

Method `set_monitoring()`**:** Sets the monitoring flags on this Constraints object. If feature-name is provided, modify the feature-level override. Else, modify the top-level monitoring flag.

Usage:

```
Constraints$set_monitoring(enable_monitoring, feature_name = NULL)
```

Arguments:

`enable_monitoring` (bool): Whether to enable monitoring or not.

`feature_name` (str): Sets the feature-level monitoring flag if provided. Otherwise, sets the file-level override.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ConstraintViolations` *ConstraintViolations*

Description

Represents the constraint violations JSON file used in Amazon SageMaker Model Monitoring.

Super class

`sagemaker.mlcore::ModelMonitoringFile` -> `ConstraintViolations`

Methods

Public methods:

- `ConstraintViolations$new()`
- `ConstraintViolations$from_s3_uri()`
- `ConstraintViolations$from_string()`
- `ConstraintViolations$from_file_path()`
- `ConstraintViolations$clone()`

Method `new()`: Initializes the ConstraintViolations object used in Amazon SageMaker Model Monitoring.

Usage:

```
ConstraintViolations$new(
  body_dict = NULL,
  constraint_violations_file_s3_uri = NULL,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`body_dict` (str): The body of the constraint violations JSON file.

`constraint_violations_file_s3_uri` (str): The uri of the constraint violations JSON file.

`kms_key` (str): The kms key to be used to decrypt the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Method `from_s3_uri()`: Generates a ConstraintViolations object from an s3 uri.

Usage:

```
ConstraintViolations$from_s3_uri(
  constraint_violations_file_s3_uri,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`constraint_violations_file_s3_uri` (str): The uri of the constraint violations JSON file.

`kms_key` (str): The kms key to be used to decrypt the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.ConstraintViolations: The instance of ConstraintViolations generated from the s3 uri.

Method `from_string()`: Generates a ConstraintViolations object from an s3 uri.

Usage:

```
ConstraintViolations$from_string(
  constraint_violations_file_string,
  kms_key = NULL,
  file_name = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`constraint_violations_file_string` (str): The uri of the constraint violations JSON file.

`kms_key` (str): The kms key to be used to encrypt the file in S3.

`file_name` (str): The file name to use when uploading to S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.ConstraintViolations: The instance of ConstraintViolations generated from the s3 uri.

Method `from_file_path()`**:** Initializes a ConstraintViolations object from a file path.

Usage:

```
ConstraintViolations$from_file_path(
  constraint_violations_file_path,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`constraint_violations_file_path` (str): The path to the constraint violations file.

`kms_key` (str): The kms_key to use when encrypting the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.ConstraintViolations: The instance of ConstraintViolations generated from the local file path.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ContinuousParameter` *ContinuousParameter Class*

Description

A class for representing hyperparameters that have a continuous range of possible values.

Super class

`sagemaker.mlcore::ParameterRange` -> `ContinuousParameter`

Public fields

`.name` Helps to categorise Class

Methods

Public methods:

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

Method `new()`: Initialize a ContinuousParameter

Usage:

```
ContinuousParameter$new(
  min_value,
  max_value,
  scaling_type = c("Auto", "Linear", "Logarithmic", "ReverseLogarithmic")
)
```

Arguments:

`min_value` (float): The minimum value for the range.

`max_value` (float): The maximum value for the range.

`scaling_type` (str): The scale used for searching the range during tuning (default: 'Auto'). Valid values: 'Auto', 'Linear', Logarithmic' and 'ReverseLogarithmic'.

Method `format()`: format class

Usage:

```
ContinuousParameter$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

create_identical_dataset_and_algorithm_tuner

Creates a new tuner with an identical dataset and algorithm.

Description

It does this identical creation by copying the request fields from the provided parent to the new instance of “HyperparameterTuner“ followed by addition of warm start configuration with the type as "IdenticalDataAndAlgorithm" and “parents“ as the union of provided list of “additional_parents“ and the “parent“.

Usage

```
create_identical_dataset_and_algorithm_tuner(
  parent,
  additional_parents = NULL,
  sagemaker_session = NULL
)
```

Arguments

`parent`	(str): Primary parent tuning job’s name from which the Tuner and Estimator configuration has to be copied
`additional_parents`	(setstr): Set of additional parent tuning job’s names along with the primary parent tuning job name to be used in warm starting the transfer learning tuner.
`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.

Value

sagemaker.tuner.HyperparameterTuner: a new “HyperparameterTuner“ object for the warm-started hyperparameter tuning job

create_transfer_learning_tuner

Creates a new “HyperParameterTuner“ instance from the parent.

Description

It creates the new tuner by copying the request fields from the provided parent to the new instance of “HyperparameterTuner“ followed by addition of warm start configuration with the type as "TransferLearning" and “parents“ as the union of provided list of “additional_parents“ and the “parent“.

Usage

```
create_transfer_learning_tuner(
  parent,
  additional_parents = NULL,
  estimator = NULL,
  sagemaker_session = NULL
)
```

Arguments

`parent`	(str): Primary parent tuning job's name from which the Tuner and Estimator configuration has to be copied
`additional_parents`	(setstr): Set of additional parent tuning job's names along with the primary parent tuning job name to be used in warm starting the identical dataset and algorithm tuner.
`estimator`	(sagemaker.estimator.EstimatorBase): An estimator object that has been initialized with the desired configuration. There does not need to be a training job associated with this instance.
`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.

Value

sagemaker.tuner.HyperparameterTuner: New instance of warm started HyperparameterTuner

---

`CronExpressionGenerator` *CronExpressionGenerator class*

Description

Generates cron expression strings for the SageMaker Model Monitoring Schedule API.

Methods

Public methods:

- `CronExpressionGenerator$hourly()`
- `CronExpressionGenerator$daily()`
- `CronExpressionGenerator$daily_every_x_hours()`
- `CronExpressionGenerator$format()`
- `CronExpressionGenerator$clone()`

Method `hourly()`**:** Generates hourly cron expression that denotes that a job runs at the top of every hour.

Usage:

```
CronExpressionGenerator$hourly()
```

Returns: str: The cron expression format accepted by the Amazon SageMaker Model Monitoring Schedule API.

Method `daily()`**:** Generates daily cron expression that denotes that a job runs at the top of every hour.

Usage:

```
CronExpressionGenerator$daily(hour = 0L)
```

Arguments:

`hour` (int): The hour in HH24 format (UTC) to run the job at, on a daily schedule.

Returns: str: The cron expression format accepted by the Amazon SageMaker Model Monitoring Schedule API.

Method `daily_every_x_hours()`**:** Generates "daily every x hours" cron expression. That denotes that a job runs every day at the specified hour, and then every x hours, as specified in hour_interval.

Usage:

```
CronExpressionGenerator$daily_every_x_hours(hour_interval, starting_hour = 0L)
```

Arguments:

`hour_interval` (int): The hour interval to run the job at.

`starting_hour` (int): The hour at which to begin in HH24 format (UTC).

Returns: str: The cron expression format accepted by the Amazon SageMaker Model Monitoring Schedule API.

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

Usage:

```
CronExpressionGenerator$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`CSVDeserializer` *Deserialize a stream of bytes into a list of lists.*

Description

Consider using `NumpyDeserializer` or `DataTableDeserializer` or `TibbleDeserializer` instead, if you'd like to convert text/csv responses directly into other data types.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `CSVDeserializer`

Public fields

`encoding` string encoding to be used

Methods

Public methods:

- `CSVDeserializer$new()`
- `CSVDeserializer$deserialize()`
- `CSVDeserializer$clone()`

Method `new()`**:** Initialize a “CSVDeserializer“ instance.

Usage:

```
CSVDeserializer$new(encoding = "UTF-8", accept = "text/csv")
```

Arguments:

`encoding` (str): The string encoding to use (default: "UTF-8").

`accept` (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: "text/csv").

Method `deserialize()`**:** Takes raw data stream and deserializes it.

Usage:

```
CSVDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` raw data to be deserialize

`content_type` (str): The MIME type of the data.

Returns: list: The data deserialized into a list of lists representing the contents of a CSV file.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`CSVSerializer` *CSVSerializer Class*

Description

Make Raw data using text/csv format

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `CSVSerializer`

Methods

Public methods:

- `CSVSerializer$new()`
- `CSVSerializer$serialize()`
- `CSVSerializer$clone()`

Method `new()`**:** Initialize a `CSVSerializer` instance.

Usage:

```
CSVSerializer$new(content_type = "text/csv")
```

Arguments:

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "text/csv").

Method `serialize()`**:** Take data of various data formats and serialize them into CSV.

Usage:

```
CSVSerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized. Any list of same length vectors; e.g. data.frame and data.table. If matrix, it gets internally coerced to data.table preserving col names but not row names

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`DataCaptureConfig` *DataCaptureConfig Class*

Description

Configuration object passed in when deploying models to Amazon SageMaker Endpoints. This object specifies configuration related to endpoint data capture for use with Amazon SageMaker Model Monitoring.

Public fields

`enable_capture` Whether data capture should be enabled or not.

`sampling_percentage` The percentage of data to sample.

`destination_s3_uri` Defaults to "s3://<default-session-bucket>/model-monitor/data-capture"

`kms_key_id` The kms key to use when writing to S3.

`capture_options` Denotes which data to capture between request and response.

`csv_content_types` Default=["text/csv"].

`json_content_types` Default=["application/json"].

`sagemaker_session` A SageMaker Session object

`API_MAPPING` Convert to API values or pass value directly through if unable to convert

Methods

Public methods:

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

Method `new()`**:** Initialize a DataCaptureConfig object for capturing data from Amazon SageMaker Endpoints.

Usage:

```
DataCaptureConfig$new(
  enable_capture = TRUE,
  sampling_percentage = 20L,
  destination_s3_uri = NULL,
  kms_key_id = NULL,
  capture_options = NULL,
  csv_content_types = NULL,
  json_content_types = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`enable_capture` (bool): Required. Whether data capture should be enabled or not.

`sampling_percentage` (int): Optional. Default=20. The percentage of data to sample. Must be between 0 and 100.

`destination_s3_uri` (str): Optional. Defaults to "s3://<default-session-bucket>/ model-monitor/data-capture".

`kms_key_id` (str): Optional. Default=None. The kms key to use when writing to S3.

`capture_options` ([str]): Optional. Must be a list containing any combination of the following values: "REQUEST", "RESPONSE". Default=["REQUEST", "RESPONSE"]. Denotes which data to capture between request and response.

`csv_content_types` ([str]): Optional. Default=["text/csv"].

`json_content_types` ([str]): Optional. Default=["application/json"].

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

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

Usage:

```
DataCaptureConfig$to_request_list()
```

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

Usage:

```
DataCaptureConfig$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`DatasetFormat` *DatasetFormat Class*

Description

Represents a Dataset Format that is used when calling a DefaultModelMonitor.

Methods

Public methods:

- `DatasetFormat$csv()`
- `DatasetFormat$json()`
- `DatasetFormat$sagemaker_capture_json()`
- `DatasetFormat$format()`
- `DatasetFormat$clone()`

Method `csv()`**:** Returns a DatasetFormat JSON string for use with a DefaultModelMonitor.

Usage:

```
DatasetFormat$csv(header = TRUE, output_columns_position = c("START", "END"))
```

Arguments:

`header` (bool): Whether the csv dataset to baseline and monitor has a header. Default: True.

`output_columns_position` (str): The position of the output columns. Must be one of ("START", "END"). Default: "START".

Returns: dict: JSON string containing DatasetFormat to be used by DefaultModelMonitor.

Method `json()`**:** Returns a DatasetFormat JSON string for use with a DefaultModelMonitor.

Usage:

```
DatasetFormat$json(lines = TRUE)
```

Arguments:

`lines` (bool): Whether the file should be read as a json object per line. Default: True.

Returns: dict: JSON string containing DatasetFormat to be used by DefaultModelMonitor.

Method `sagemaker_capture_json()`**:** Returns a DatasetFormat SageMaker Capture Json string for use with a DefaultModelMonitor.

Usage:

```
DatasetFormat$sagemaker_capture_json()
```

Returns: dict: JSON string containing DatasetFormat to be used by DefaultModelMonitor.

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

Usage:

```
DatasetFormat$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`DataTableDeserializer` *DataTableDeserializer Class*

Description

Deserialize CSV or JSON data from an inference endpoint into a data.table.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `DataTableDeserializer`

Public fields

`encoding` string encoding to be used

Methods

Public methods:

- `DataTableDeserializer$new()`
- `DataTableDeserializer$deserialize()`
- `DataTableDeserializer$clone()`

Method `new()`**:** Initialize a “DataTableDeserializer“ instance.

Usage:

```
DataTableDeserializer$new(
  encoding = "UTF-8",
  accept = c("text/csv", "application/json")
)
```

Arguments:

`encoding` (str): The string encoding to use (default: "UTF-8").

`accept` (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: ("text/csv","application/json")).

Method `deserialize()`**:** Deserialize CSV or JSON data from an inference endpoint into a data.table. If the data is JSON, the data should be formatted in the 'columns' orient.

Usage:

```
DataTableDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` (botocore.response.StreamingBody): Data to be deserialized.

`content_type` (str): The MIME type of the data.

Returns: data.table: The data deserialized into a data.table.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`DefaultModelMonitor` *DefaultModelMonitor Class*

Description

Sets up Amazon SageMaker Monitoring Schedules and baseline suggestions. Use this class when you want to utilize Amazon SageMaker Monitoring's plug-and-play solution that only requires your dataset and optional pre/postprocessing scripts. For a more customized experience, consider using the ModelMonitor class instead.

Super class

`sagemaker.mlcore::ModelMonitor` -> `DefaultModelMonitor`

Public fields

`JOB_DEFINITION_BASE_NAME` Model definition base name

Methods

Public methods:

- `DefaultModelMonitor$new()`
- `DefaultModelMonitor$monitoring_type()`
- `DefaultModelMonitor$suggest_baseline()`
- `DefaultModelMonitor$create_monitoring_schedule()`
- `DefaultModelMonitor$update_monitoring_schedule()`
- `DefaultModelMonitor$delete_monitoring_schedule()`
- `DefaultModelMonitor$run_baseline()`

- `DefaultModelMonitor$attach()`
- `DefaultModelMonitor$latest_monitoring_statistics()`
- `DefaultModelMonitor$latest_monitoring_constraint_violations()`
- `DefaultModelMonitor$clone()`

Method `new()`**:** Initializes a “Monitor“ instance. The Monitor handles baselining datasets and creating Amazon SageMaker Monitoring Schedules to monitor SageMaker endpoints.

Usage:

```
DefaultModelMonitor$new(
  role,
  instance_count = 1,
  instance_type = "ml.m5.xlarge",
  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. The Amazon SageMaker jobs use this role.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.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.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

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

`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.

`env` (dict): Environment variables to be passed to the job.

`tags` ([dict]): List of tags to be passed to the job.

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

Method `monitoring_type()`**:** Type of the monitoring job.

Usage:

```
DefaultModelMonitor$monitoring_type()
```

Method `suggest_baseline()`**:** Suggest baselines for use with Amazon SageMaker Model Monitoring Schedules.

Usage:

```
DefaultModelMonitor$suggest_baseline(
  baseline_dataset,
  dataset_format,
  record_preprocessor_script = NULL,
  post_analytics_processor_script = NULL,
  output_s3_uri = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL
)
```

Arguments:

`baseline_dataset` (str): The path to the baseline_dataset file. This can be a local path or an S3 uri.

`dataset_format` (dict): The format of the baseline_dataset.

`record_preprocessor_script` (str): The path to the record preprocessor script. This can be a local path or an S3 uri.

`post_analytics_processor_script` (str): The path to the record post-analytics processor script. This can be a local path or an S3 uri.

`output_s3_uri` (str): Desired S3 destination Destination of the constraint_violations and statistics json files. Default: "s3://<default_session_bucket>/<job_name>/output"

`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 image name and current timestamp.

Returns: sagemaker.processing.ProcessingJob: The ProcessingJob object representing the baselining job.

Method `create_monitoring_schedule()`**:** Creates a monitoring schedule to monitor an Amazon SageMaker Endpoint. If constraints and statistics are provided, or if they are able to be retrieved from a previous baselining job associated with this monitor, those will be used. If constraints and statistics cannot be automatically retrieved, baseline_inputs will be required in order to kick off a baselining job.

Usage:

```
DefaultModelMonitor$create_monitoring_schedule(
  endpoint_input,
  record_preprocessor_script = NULL,
  post_analytics_processor_script = NULL,
  output_s3_uri = NULL,
  constraints = NULL,
  statistics = NULL,
  monitor_schedule_name = NULL,
  schedule_cron_expression = NULL,
  enable_cloudwatch_metrics = TRUE
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`record_preprocessor_script` (str): The path to the record preprocessor script. This can be a local path or an S3 uri.

`post_analytics_processor_script` (str): The path to the record post-analytics processor script. This can be a local path or an S3 uri.

`output_s3_uri` (str): Desired S3 destination of the constraint_violations and statistics json files. Default: "s3://<default_session_bucket>/<job_name>/output"

`constraints` (sagemaker.model_monitor.Constraints or str): If provided alongside statistics, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an s3_uri pointing to a constraints JSON file.

`statistics` (sagemaker.model_monitor.Statistic or str): If provided alongside constraints, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an s3_uri pointing to a constraints JSON file.

`monitor_schedule_name` (str): Schedule name. If not specified, the processor generates a default job name, based on the image name and current timestamp.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job run. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

Method `update_monitoring_schedule()`**:** Updates the existing monitoring schedule.

Usage:

```
DefaultModelMonitor$update_monitoring_schedule(
  endpoint_input = NULL,
  record_preprocessor_script = NULL,
  post_analytics_processor_script = NULL,
  output_s3_uri = NULL,
  statistics = NULL,
  constraints = NULL,
  schedule_cron_expression = NULL,
  instance_count = NULL,
  instance_type = NULL,
  volume_size_in_gb = NULL,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  env = NULL,
  network_config = NULL,
  enable_cloudwatch_metrics = NULL,
  role = NULL
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`record_preprocessor_script` (str): The path to the record preprocessor script. This can be a local path or an S3 uri.

`post_analytics_processor_script` (str): The path to the record post-analytics processor script. This can be a local path or an S3 uri.

`output_s3_uri` (str): Desired S3 destination of the constraint_violations and statistics json files.

`statistics` (sagemaker.model_monitor.Statistic or str): If provided alongside constraints, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an S3 uri pointing to a constraints JSON file.

`constraints` (sagemaker.model_monitor.Constraints or str): If provided alongside statistics, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an S3 uri pointing to a constraints JSON file.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job runs at. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

`env` (dict): Environment variables to be passed to the job.

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

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

`role` (str): An AWS IAM role name or ARN. The Amazon SageMaker jobs use this role.

Method `delete_monitoring_schedule()`**:** Deletes the monitoring schedule and its job definition.

Usage:

```
DefaultModelMonitor$delete_monitoring_schedule()
```

Method `run_baseline()`**:** 'run_baseline()' is only allowed for ModelMonitor objects. Please use suggest_baseline for DefaultModelMonitor objects, instead.

Usage:

```
DefaultModelMonitor$run_baseline()
```

Method `attach()`**:** Sets this object's schedule name to point to the Amazon Sagemaker Monitoring Schedule name provided. This allows subsequent describe_schedule or list_executions calls to point to the given schedule.

Usage:

`DefaultModelMonitor$attach(monitor_schedule_name, sagemaker_session = NULL)`

Arguments:

`monitor_schedule_name` (str): The name of the schedule to attach to.

`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 `latest_monitoring_statistics()`**:** Returns the sagemaker.model_monitor.Statistics generated by the latest monitoring execution.

Usage:

`DefaultModelMonitor$latest_monitoring_statistics()`

Returns: sagemaker.model_monitoring.Statistics: The Statistics object representing the file generated by the latest monitoring execution.

Method `latest_monitoring_constraint_violations()`**:** Returns the sagemaker.model_monitor.ConstraintViolations generated by the latest monitoring execution.

Usage:

`DefaultModelMonitor$latest_monitoring_constraint_violations()`

Returns: sagemaker.model_monitoring.ConstraintViolations: The ConstraintViolations object representing the file generated by the latest monitoring execution.

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

Usage:

`DefaultModelMonitor$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

`EndpointInput` *Accepts parameters that specify an endpoint input for monitoring execution.*

Description

It also provides a method to turn those parameters into a dictionary.

Methods

Public methods:

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

Method `new()`**:** Initialize an “EndpointInput“ instance. EndpointInput accepts parameters that specify an endpoint input for a monitoring job and provides a method to turn those parameters into a dictionary.

Usage:

```
EndpointInput$new(
  endpoint_name,
  destination,
  s3_input_mode = c("File", "Pipe"),
  s3_data_distribution_type = c("FullyReplicated", "ShardedByS3Key"),
  start_time_offset = NULL,
  end_time_offset = NULL,
  features_attribute = NULL,
  inference_attribute = NULL,
  probability_attribute = NULL,
  probability_threshold_attribute = NULL
)
```

Arguments:

`endpoint_name` (str): The name of the endpoint.

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

`s3_input_mode` (str): The S3 input mode. Can be one of: "File", "Pipe. Default: "File".

`s3_data_distribution_type` (str): The S3 Data Distribution Type. Can be one of: "FullyReplicated", "ShardedByS3Key"

`start_time_offset` (str): Monitoring start time offset, e.g. "-PT1H"

`end_time_offset` (str): Monitoring end time offset, e.g. "-PT0H".

`features_attribute` (str): JSONpath to locate features in JSONlines dataset. Only used for ModelBiasMonitor and ModelExplainabilityMonitor

`inference_attribute` (str): Index or JSONpath to locate predicted label(s). Only used for ModelQualityMonitor, ModelBiasMonitor, and ModelExplainabilityMonitor

`probability_attribute` (str or int): Index or JSONpath to locate probabilities. Only used for ModelQualityMonitor, ModelBiasMonitor and ModelExplainabilityMonitor

`probability_threshold_attribute` (float): threshold to convert probabilities to binaries Only used for ModelQualityMonitor, ModelBiasMonitor and ModelExplainabilityMonitor

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

Usage:

```
EndpointInput$to_request_list()
```

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

Usage:

```
EndpointInput$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`EndpointOutput` *Accepts parameters that specify an S3 output for a monitoring job.*

Description

It also provides a method to turn those parameters into a dictionary.

Methods

Public methods:

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

Method `new()`**:** Initialize a “MonitoringOutput“ instance. MonitoringOutput accepts parameters that specify an S3 output for a monitoring job and provides a method to turn those parameters into a dictionary.

Usage:

```
EndpointOutput$new(source, destination = NULL, s3_upload_mode = "Continuous")
```

Arguments:

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

`destination` (str): The destination of the output. Optional. Default: s3://<default-session-bucket/schedule_name/output

`s3_upload_mode` (str): The S3 upload mode.

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

Usage:

```
EndpointOutput$to_request_list()
```

Returns: dict: The request dictionary.

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

Usage:

```
EndpointOutput$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Estimator *Sagemaker Estimator Class*

Description

A generic Estimator to train using any supplied algorithm. This class is designed for use with algorithms that don't have their own, custom class.

Super class

sagemaker.mlcore::EstimatorBase -> Estimator

Public fields

.module mimic python module

Methods

Public methods:

- Estimator$new()
- Estimator$training_image_uri()
- Estimator$set_hyperparameters()
- Estimator$hyperparameters()
- Estimator$create_model()
- Estimator$clone()

Method new()**:** Initialize an "Estimator" instance.

Usage:

```
Estimator$new(
  image_uri,
  role,
  instance_count = NULL,
  instance_type = NULL,
  volume_size = 30,
  volume_kms_key = NULL,
  max_run = 24 * 60 * 60,
  input_mode = "File",
  output_path = NULL,
  output_kms_key = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  hyperparameters = NULL,
  tags = NULL,
  subnets = NULL,
  security_group_ids = NULL,
  model_uri = NULL,
  model_channel_name = "model",
  metric_definitions = NULL,
  encrypt_inter_container_traffic = FALSE,
  use_spot_instances = FALSE,
  max_wait = NULL,
  checkpoint_s3_uri = NULL,
  checkpoint_local_path = NULL,
  enable_network_isolation = FALSE,
  rules = NULL,
  debugger_hook_config = NULL,
  tensorboard_output_config = NULL,
  enable_sagemaker_metrics = NULL,
  profiler_config = NULL,
  disable_profiler = FALSE,
  environment = NULL,
  max_retry_attempts = NULL,
  ...
)
```

Arguments:

`image_uri` (str): The container image to use for training.

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.

`instance_count` (int): Number of Amazon EC2 instances to use for training.

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

`volume_size` (int): Size in GB of the EBS volume to use for storing input data during training (default: 30). Must be large enough to store training data if File Mode is used (which is the default).

`volume_kms_key` (str): Optional. KMS key ID for encrypting EBS volume attached to the training instance (default: NULL).

`max_run` (int): Timeout in seconds for training (default: 24 * 60 * 60). After this amount of time Amazon SageMaker terminates the job regardless of its current status.

`input_mode` (str): The input mode that the algorithm supports (default: 'File'). Valid modes: * 'File' - Amazon SageMaker copies the training dataset from the S3 location to a local directory. * 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a Unix-named pipe. This argument can be overriden on a per-channel basis using "TrainingInput.input_mode".

`output_path` (str): S3 location for saving the training result (model artifacts and output files). If not specified, results are stored to a default bucket. If the bucket with the specific name does not exist, the estimator creates the bucket during the :meth:'~sagemaker.estimator.EstimatorBase.fit' method execution.

`output_kms_key` (str): Optional. KMS key ID for encrypting the training output (default: NULL).

`base_job_name` (str): Prefix for training job name when the :meth:'~sagemaker.estimator.EstimatorBase.fit' method launches. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

`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.

`hyperparameters` (dict): Dictionary containing the hyperparameters to initialize this estimator with.

`tags` (list[dict]): List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/

`subnets` (list[str]): List of subnet ids. If not specified training job will be created without VPC config.

`security_group_ids` (list[str]): List of security group ids. If not specified training job will be created without VPC config.

`model_uri` (str): URI where a pre-trained model is stored, either locally or in S3 (default: NULL). If specified, the estimator can download it. This model can be a 'model.tar.gz' from a previous training job, or other artifacts coming from a different source. In local mode, this should point to the path in which the model is located and not the file itself, as local Docker containers will try to mount the URI as a volume. More information: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html#td-deserialization

`model_channel_name` (str): Name of the channel where 'model_uri' will be downloaded (default: 'model').

`metric_definitions` (list[dict]): A list of dictionaries that defines the metric(s) used to evaluate the training jobs. Each dictionary contains two keys: 'Name' for the name of the metric, and 'Regex' for the regular expression used to extract the metric from the logs. This should be defined only for jobs that don't use an Amazon algorithm.

`encrypt_inter_container_traffic` (bool): Specifies whether traffic between training containers is encrypted for the training job (default: "False").

`use_spot_instances` (bool): Specifies whether to use SageMaker Managed Spot instances for training. If enabled then the 'max_wait' arg should also be set. More information: https://docs.aws.amazon.com/sagemaker/latest/dg/model-managed-spot-training.html (default: "False").

`max_wait` (int): Timeout in seconds waiting for spot training instances (default: NULL). After this amount of time Amazon SageMaker will stop waiting for Spot instances to become available (default: "NULL").

`checkpoint_s3_uri` (str): The S3 URI in which to persist checkpoints that the algorithm persists (if any) during training. (default: "NULL").

`checkpoint_local_path` (str): The local path that the algorithm writes its checkpoints to. SageMaker will persist all files under this path to 'checkpoint_s3_uri' continually during training. On job startup the reverse happens - data from the s3 location is downloaded to this path before the algorithm is started. If the path is unset then SageMaker assumes the checkpoints will be provided under '/opt/ml/checkpoints/'. (default: "NULL").

`enable_network_isolation` (bool): Specifies whether container will run in network isolation mode (default: "False"). Network isolation mode restricts the container access to outside networks (such as the Internet). The container does not make any inbound or outbound network calls. Also known as Internet-free mode.

`rules` (list[:class:'~sagemaker.debugger.Rule']): A list of :class:'~sagemaker.debugger.Rule' objects used to define rules for continuous analysis with SageMaker Debugger (default: "NULL"). For more, see https://sagemaker.readthedocs.io/en/stable/amazon_sagemaker_debugger.html#continuous-analyses-through-rules `debugger_hook_config` (:class:‘~sagemaker.debugger.DebuggerHookConfig‘ or bool): Configuration for how debugging information is emitted with SageMaker Debugger. If not specified, a default one is created using the estimator’s “output_path“, unless the region does not support SageMaker Debugger. To disable SageMaker Debugger, set this parameter to “False“. For more, see https://sagemaker.readthedocs.io/en/stable/amazon_sagemaker_debugger.html

`tensorboard_output_config` (:class:‘~sagemaker.debugger.TensorBoardOutputConfig‘): Configuration for customizing debugging visualization using TensorBoard (default: “NULL“). For more, see https://sagemaker.readthedocs.io/en/stable/amazon_sagemaker_debugger.html#capture-real-time-tensorboard-data-from-the-debugging-hook

`enable_sagemaker_metrics` (bool): enable SageMaker Metrics Time Series. For more information see: https://docs.aws.amazon.com/sagemaker/latest/dg/API_AlgorithmSpecification.html#SageMaker-Type-AlgorithmSpecification-EnableSageMakerMetricsTimeSeries (default: “NULL“).

`profiler_config` (:class:‘~sagemaker.debugger.ProfilerConfig‘): Configuration for how SageMaker Debugger collects monitoring and profiling information from your training job. If not specified, Debugger will be configured with a default configuration and will save system and framework metrics the estimator’s default “output_path“ in Amazon S3. Use :class:‘~sagemaker.debugger.ProfilerConfig‘ to configure this parameter. To disable SageMaker Debugger monitoring and profiling, set the “disable_profiler“ parameter to “True“.

`disable_profiler` (bool): Specifies whether Debugger monitoring and profiling will be disabled (default: “False“).

`environment` (dict[str, str]) : Environment variables to be set for use during training job (default: “None“)

`max_retry_attempts` (int): The number of times to move a job to the STARTING status. You can specify between 1 and 30 attempts. If the value of attempts is greater than zero, the job is retried on InternalServerFailure the same number of attempts as the value. You can cap the total duration for your job by setting “max_wait“ and “max_run“ (default: “None“)

`...` : additional arguements for parent class ‘EstimatorBase‘.

Method `training_image_uri()`**:** Returns the docker image to use for training. The fit() method, that does the model training, calls this method to find the image to use for model training.

Usage:

```
Estimator$training_image_uri()
```

Method `set_hyperparameters()`**:** formats hyperparameters for model tunning

Usage:

```
Estimator$set_hyperparameters(...)
```

Arguments:

`...` model hyperparameters

Method `hyperparameters()`**:** Returns the hyperparameters as a dictionary to use for training. The fit() method, that does the model training, calls this method to find the hyperparameters you specified.

Usage:

```
Estimator$hyperparameters()
```

Method `create_model()`**:** Create a model to deploy. The serializer, deserializer, content_type, and accept arguments are only used to define a default Predictor. They are ignored if an explicit predictor class is passed in. Other arguments are passed through to the Model class.

Usage:

```
Estimator$create_model(
  role = NULL,
  image_uri = NULL,
  predictor_cls = NULL,
  vpc_config_override = "VPC_CONFIG_DEFAULT",
  ...
)
```

Arguments:

`role` (str): The “ExecutionRoleArn“ IAM Role ARN for the “Model“, which is also used during transform jobs. If not specified, the role from the Estimator will be used.

`image_uri` (str): An container image to use for deploying the model. Defaults to the image used for training.

`predictor_cls` (Predictor): The predictor class to use when deploying the model.

`vpc_config_override` (dict[str, list[str]]): Optional override for VpcConfig set on the model. Default: use subnets and security groups from this Estimator. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

`...` : Additional parameters passed to :class:‘~sagemaker.model.Model‘ .. tip:: You can find additional parameters for using this method at :class:‘~sagemaker.model.Model‘.

Returns: (sagemaker.model.Model) a Model ready for deployment.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`EstimatorBase` *Handle end-to-end Amazon SageMaker training and deployment tasks.*

Description

For introduction to model training and deployment, see http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html Subclasses must define a way to determine what image to use for training, what hyperparameters to use, and how to create an appropriate predictor instance.

Active bindings

`model_data` The model location in S3. Only set if Estimator has been “fit()“.

`training_job_analytics` Return a “TrainingJobAnalytics“ object for the current training job.

Methods

Public methods:

- EstimatorBase$new()
- EstimatorBase$help()
- EstimatorBase$training_image_uri()
- EstimatorBase$hyperparameters()
- EstimatorBase$enable_network_isolation()
- EstimatorBase$prepare_workflow_for_training()
- EstimatorBase$latest_job_debugger_artifacts_path()
- EstimatorBase$latest_job_tensorboard_artifacts_path()
- EstimatorBase$latest_job_profiler_artifacts_path()
- EstimatorBase$fit()
- EstimatorBase$wait()
- EstimatorBase$describe()
- EstimatorBase$rule_job_summary()
- EstimatorBase$compile_model()
- EstimatorBase$attach()
- EstimatorBase$logs()
- EstimatorBase$deploy()
- EstimatorBase$register()
- EstimatorBase$create_model()
- EstimatorBase$delete_endpoint()
- EstimatorBase$transformer()
- EstimatorBase$get_vpc_config()
- EstimatorBase$.prepare_for_training()
- EstimatorBase$enable_default_profiling()
- EstimatorBase$disable_profiling()
- EstimatorBase$update_profiler()
- EstimatorBase$format()
- EstimatorBase$clone()

Method `new()`**:** Initialize an “EstimatorBase“ instance.

Usage:

```
EstimatorBase$new(
  role,
  instance_count = NULL,
  instance_type = NULL,
  volume_size = 30,
  volume_kms_key = NULL,
  max_run = 24 * 60 * 60,
  input_mode = "File",
  output_path = NULL,
  output_kms_key = NULL,
  base_job_name = NULL,
  sagemaker_session = NULL,
  tags = NULL,
  subnets = NULL,
  security_group_ids = NULL,
  model_uri = NULL,
  model_channel_name = "model",
  metric_definitions = NULL,
  encrypt_inter_container_traffic = FALSE,
  use_spot_instances = FALSE,
  max_wait = NULL,
  checkpoint_s3_uri = NULL,
  checkpoint_local_path = NULL,
  rules = NULL,
  debugger_hook_config = NULL,
  tensorboard_output_config = NULL,
  enable_sagemaker_metrics = NULL,
  enable_network_isolation = FALSE,
  profiler_config = NULL,
  disable_profiler = FALSE,
  environment = NULL,
  max_retry_attempts = NULL,
  source_dir = NULL,
  git_config = NULL,
  hyperparameters = NULL,
  container_log_level = "INFO",
  code_location = NULL,
  entry_point = NULL,
  dependencies = NULL,
  ...
)
```

Arguments:

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.

`instance_count` (int): Number of Amazon EC2 instances to use for training.

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

`volume_size` (int): Size in GB of the EBS volume to use for storing input data during training (default: 30). Must be large enough to store training data if File Mode is used (which is the default).

`volume_kms_key` (str): Optional. KMS key ID for encrypting EBS volume attached to the training instance (default: NULL).

`max_run` (int): Timeout in seconds for training (default: 24 * 60 * 60). After this amount of time Amazon SageMaker terminates the job regardless of its current status.

`input_mode` (str): The input mode that the algorithm supports (default: 'File'). Valid modes: 'File' - Amazon SageMaker copies the training dataset from the S3 location to a local directory. 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a

Unix-named pipe. This argument can be overriden on a per-channel basis using "Training-Input.input_mode".

`output_path` (str): S3 location for saving the training result (model artifacts and output files). If not specified, results are stored to a default bucket. If the bucket with the specific name does not exist, the estimator creates the bucket during the :meth:'~sagemaker.estimator.EstimatorBase.fit' method execution. file:// urls are used for local mode. For example: 'file://model/' will save to the model folder in the current directory.

`output_kms_key` (str): Optional. KMS key ID for encrypting the training output (default: NULL).

`base_job_name` (str): Prefix for training job name when the :meth:'~sagemaker.estimator.EstimatorBase.fit' method launches. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

`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.

`tags` (list[dict]): List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

`subnets` (list[str]): List of subnet ids. If not specified training job will be created without VPC config.

`security_group_ids` (list[str]): List of security group ids. If not specified training job will be created without VPC config.

`model_uri` (str): URI where a pre-trained model is stored, either locally or in S3 (default: NULL). If specified, the estimator will create a channel pointing to the model so the training job can download it. This model can be a 'model.tar.gz' from a previous training job, or other artifacts coming from a different source. In local mode, this should point to the path in which the model is located and not the file itself, as local Docker containers will try to mount the URI as a volume. More information: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html#td-deserialization

`model_channel_name` (str): Name of the channel where 'model_uri' will be downloaded (default: 'model').

`metric_definitions` (list[dict]): A list of dictionaries that defines the metric(s) used to evaluate the training jobs. Each dictionary contains two keys: 'Name' for the name of the metric, and 'Regex' for the regular expression used to extract the metric from the logs. This should be defined only for jobs that don't use an Amazon algorithm.

`encrypt_inter_container_traffic` (bool): Specifies whether traffic between training containers is encrypted for the training job (default: "False").

`use_spot_instances` (bool): Specifies whether to use SageMaker Managed Spot instances for training. If enabled then the 'max_wait' arg should also be set. More information: https://docs.aws.amazon.com/sagemaker/latest/dg/model-managed-spot-training.html (default: "False").

`max_wait` (int): Timeout in seconds waiting for spot training instances (default: NULL). After this amount of time Amazon SageMaker will stop waiting for Spot instances to become available (default: "NULL").

`checkpoint_s3_uri` (str): The S3 URI in which to persist checkpoints that the algorithm persists (if any) during training. (default: "NULL").

`checkpoint_local_path` (str): The local path that the algorithm writes its checkpoints to. SageMaker will persist all files under this path to 'checkpoint_s3_uri' continually during training. On job startup the reverse happens - data from the s3 location is downloaded to this path before the algorithm is started. If the path is unset then SageMaker assumes the checkpoints will be provided under '/opt/ml/checkpoints/'. (default: "NULL").

`rules` (list[:class:'~sagemaker.debugger.Rule']): A list of :class:'~sagemaker.debugger.Rule' objects used to define rules for continuous analysis with SageMaker Debugger (default: "NULL"). For more, see `https://sagemaker.readthedocs.io/en/stable/amazon_sagemaker_debugger.html#continuous-analyses-through-rules`

`debugger_hook_config` (:class:'~sagemaker.debugger.DebuggerHookConfig' or bool): Configuration for how debugging information is emitted with SageMaker Debugger. If not specified, a default one is created using the estimator's "output_path", unless the region does not support SageMaker Debugger. To disable SageMaker Debugger, set this parameter to "False". For more, see https://sagemaker.readthedocs.io/en/stable/amazon_sagemaker_debugger.html

`tensorboard_output_config` (:class:'~sagemaker.debugger.TensorBoardOutputConfig'): Configuration for customizing debugging visualization using TensorBoard (default: "NULL"). For more, see `https://sagemaker.readthedocs.io/en/stable/amazon_sagemaker_debugger.html#capture-real-time-tensorboard-data-from-the-debugging-hook`

`enable_sagemaker_metrics` (bool): Enables SageMaker Metrics Time Series. For more information see: `https://docs.aws.amazon.com/sagemaker/latest/dg/API_AlgorithmSpecification.html#SageMaker-Type-AlgorithmSpecification-EnableSageMakerMetricsTimeSeries` (default: "NULL").

`enable_network_isolation` (bool): Specifies whether container will run in network isolation mode (default: "False"). Network isolation mode restricts the container access to outside networks (such as the Internet). The container does not make any inbound or outbound network calls. Also known as Internet-free mode.

`profiler_config` (:class:'~sagemaker.debugger.ProfilerConfig'): Configuration for how SageMaker Debugger collects monitoring and profiling information from your training job. If not specified, a default configuration is created using the estimator's "output_path", unless the region does not support SageMaker Debugger. To disable SageMaker Debugger monitoring and profiling, set the "disable_profiler" parameter to "True".

`disable_profiler` (bool): Specifies whether Debugger monitoring and profiling will be disabled (default: "False").

`environment` (dict[str, str]) : Environment variables to be set for use during training job (default: "None")

`max_retry_attempts` (int): The number of times to move a job to the STARTING status. You can specify between 1 and 30 attempts. If the value of attempts is greater than zero, the job is retried on InternalServerFailure the same number of attempts as the value. You can cap the total duration for your job by setting "max_wait" and "max_run" (default: "None")

`source_dir` (str): The absolute, relative, or S3 URI Path to a directory with any other training 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. The structure within this directory is preserved when training on Amazon SageMaker. If 'git_config' is provided, 'source_dir' should be a relative location to a directory in the Git repo. With the following GitHub repo directory structure, if you need 'train.py' as the entry point and 'test.py' as the training source code, you can assign entry_point='train.py' and source_dir='src'.

`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 checking out the 'master' branch, and checking out 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 the SSH passphrase when you run the 'git clone' command with SSH URLs. When HTTPS URLs are provided, if 2FA is disabled, then either "token" or "username" and "password" are be used for authentication if provided. "Token" is prioritized. If 2FA is enabled, only "token" is used for authentication if provided. If required authentication info is not provided, the SageMaker Python SDK attempts to use local credentials to authenticate. If that fails, an error message is thrown. For CodeCommit repos, 2FA is not supported, so '2FA_enabled' should not be provided. There is no token in CodeCommit, so "token" should also not be provided. When "repo" is an SSH URL, the requirements are the same as GitHub repos. When "repo" is an HTTPS URL, "username" and "password" are used for authentication if they are provided. If they are not provided, the SageMaker Python SDK attempts to use either the CodeCommit credential helper or local credential storage for authentication.

`hyperparameters` (dict): A dictionary containing the hyperparameters to initialize this estimator with. (Default: None).

`container_log_level` (str): The log level to use within the container (default: logging.INFO). Valid values are defined in the Python logging module.

`code_location` (str): The S3 prefix URI where custom code is uploaded (default: None). You must not include a trailing slash because a string prepended with a "/" is appended to "code_location". The code file uploaded to S3 is 'code_location/job-name/source/sourcedir.tar.gz'. If not specified, the default "code location" is 's3://output_bucket/job-name/'.

`entry_point` (str): The absolute or relative path to the local Python source file that should be executed as the entry point to training. (Default: None). If "source_dir" is specified, then "entry_point" must point to a file located at the root of "source_dir". If 'git_config' is provided, 'entry_point' should be a relative location to the Python source file in the Git repo. You can assign entry_point='src/train.py'.

`dependencies` (list[str]): A list of absolute or relative paths to directories with any additional libraries that should be exported to the container (default: []). The library folders are 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. This is not supported with "local code" in Local Mode.

`...` : update any deprecated parameters passed into class.

Method `help()`**:** Return class documentation

Usage:

```
EstimatorBase$help()
```

Method `training_image_uri()`**:** Return the Docker image to use for training. The :meth:'~sagemaker.estimator.Estimato
method, which does the model training, calls this method to find the image to use for model training.

Usage:

```
EstimatorBase$training_image_uri()
```

Returns: str: The URI of the Docker image.

Method `hyperparameters()`**:** Return the hyperparameters as a dictionary to use for training. The :meth:‘~sagemaker.estimator.EstimatorBase.fit‘ method, which trains the model, calls this method to find the hyperparameters.

Usage:

```
EstimatorBase$hyperparameters()
```

Returns: dict[str, str]: The hyperparameters.

Method `enable_network_isolation()`**:** Return True if this Estimator will need network isolation to run.

Usage:

```
EstimatorBase$enable_network_isolation()
```

Returns: bool: Whether this Estimator needs network isolation or not.

Method `prepare_workflow_for_training()`**:** Calls _prepare_for_training. Used when setting up a workflow.

Usage:

```
EstimatorBase$prepare_workflow_for_training(job_name = NULL)
```

Arguments:

`job_name` (str): Name of the training job to be created. If not specified, one is generated, using the base name given to the constructor if applicable.

Method `latest_job_debugger_artifacts_path()`**:** Gets the path to the DebuggerHookConfig output artifacts.

Usage:

```
EstimatorBase$latest_job_debugger_artifacts_path()
```

Returns: str: An S3 path to the output artifacts.

Method `latest_job_tensorboard_artifacts_path()`**:** Gets the path to the TensorBoardOutputConfig output artifacts.

Usage:

```
EstimatorBase$latest_job_tensorboard_artifacts_path()
```

Returns: str: An S3 path to the output artifacts.

Method `latest_job_profiler_artifacts_path()`**:** Gets the path to the profiling output artifacts.

Usage:

```
EstimatorBase$latest_job_profiler_artifacts_path()
```

Returns: str: An S3 path to the output artifacts.

Method `fit()`**:** Train a model using the input training dataset. The API calls the Amazon SageMaker CreateTrainingJob API to start model training. The API uses configuration you provided to create the estimator and the specified input training data to send the CreatingTrainingJob request to Amazon SageMaker. This is a synchronous operation. After the model training successfully completes, you can call the “deploy()“ method to host the model using the Amazon SageMaker hosting services.

Usage:

```
EstimatorBase$fit(
  inputs = NULL,
  wait = TRUE,
  logs = "All",
  job_name = NULL,
  experiment_config = NULL
)
```

Arguments:

`inputs` (str or dict or TrainingInput): Information about the training data. This can be one of three types:

- **(str)** the S3 location where training data is saved, or a file:// path in local mode.
- **(dict[str, str]** or dict[str, TrainingInput]) If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:‘~TrainingInput‘ objects.
- **(TrainingInput)** - channel configuration for S3 data sources that can provide additional information as well as the path to the training dataset. See :func:‘TrainingInput‘ for full details.
- **(sagemaker.session.FileSystemInput)** - channel configuration for a file system data source that can provide additional information as well as the path to the training dataset.

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

`logs` ([str]): A list of strings specifying which logs to print. Acceptable strings are "All", "NULL", "Training", or "Rules". To maintain backwards compatibility, boolean values are also accepted and converted to strings. Only meaningful when wait is True.

`job_name` (str): Training job name. If not specified, the estimator generates a default job name, based on the training image name and current timestamp.

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

Method `wait()`**:** Wait for an Amazon SageMaker job to complete.

Usage:

```
EstimatorBase$wait(logs = "All")
```

Arguments:

`logs` ([str]): A list of strings specifying which logs to print. Acceptable strings are "All", "NULL", "Training", or "Rules". To maintain backwards compatibility, boolean values are also accepted and converted to strings.

Method `describe()`**:** Returns a response from the DescribeTrainingJob API call.

Usage:

```
EstimatorBase$describe()
```

Method `rule_job_summary()`**:** Calls describe_training_job and returns two dictionaries.

Usage:

```
EstimatorBase$rule_job_summary()
```

Returns: list[dict]: A list of DebugRuleEvaluationStatuses and ProfilerRuleEvaluationStatuses dictionary.

Method `compile_model()`**:** Compile a Neo model using the input model.

Usage:

```
EstimatorBase$compile_model(
  target_instance_family,
  input_shape,
  output_path,
  framework = NULL,
  framework_version = NULL,
  compile_max_run = 15 * 60,
  tags = NULL,
  target_platform_os = NULL,
  target_platform_arch = NULL,
  target_platform_accelerator = NULL,
  compiler_options = NULL,
  ...
)
```

Arguments:

`target_instance_family` (str): Identifies the device that you want to run your model after compilation, for example: ml_c5. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Ou

`input_shape` (dict): Specifies the name and shape of the expected inputs for your trained model in json dictionary form, for example: 'data':[1,3,1024,1024], or 'var1': [1,1,28,28], 'var2':[1,1,28,28]

`output_path` (str): Specifies where to store the compiled model

`framework` (str): The framework that is used to train the original model. Allowed values: 'mxnet', 'tensorflow', 'keras', 'pytorch', 'onnx', 'xgboost'

`framework_version` (str): The version of the framework

`compile_max_run` (int): Timeout in seconds for compilation (default: 3 * 60). After this amount of time Amazon SageMaker Neo terminates the compilation job regardless of its current status.

`tags` (list[dict]): List of tags for labeling a compilation job. For more, see https://docs.aws.amazon.com/sagemaker/latest/

`target_platform_os` (str): Target Platform OS, for example: 'LINUX'. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_OutputConfig.html. It can be used instead of target_instance_family.

`target_platform_arch` (str): Target Platform Architecture, for example: 'X86_64'. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_OutputConfig.html. It can be used instead of target_instance_family.

`target_platform_accelerator` (str, optional): Target Platform Accelerator, for example: 'NVIDIA'. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_OutputConfig.html. It can be used instead of target_instance_family.

`compiler_options` (dict, optional): Additional parameters for compiler. Compiler Options are TargetPlatform / target_instance_family specific. See https://docs.aws.amazon.com/sagemaker/latest/dg/API_Output for details.

`...` : Passed to invocation of “create_model()“. Implementations may customize “create_model()“ to accept “**kwargs“ to customize model creation during deploy. For more, see the implementation docs.

Returns: sagemaker.model.Model: A SageMaker “Model“ object. See :func:‘~sagemaker.model.Model‘ for full details.

Method `attach()`**:** Attach to an existing training job. Create an Estimator bound to an existing training job, each subclass is responsible to implement “_prepare_init_params_from_job_description()“ as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be “deploy()“ ed to create a SageMaker Endpoint and return a “Predictor“. If the training job is in progress, attach will block and display log messages from the training job, until the training job completes. Examples: »> my_estimator.fit(wait=False) »> training_job_name = my_estimator.latest_training_job.name Later on: »> attached_estimator = Estimator.attach(training_job_name) »> attached_estimator.deploy()

Usage:

```
EstimatorBase$attach(
  training_job_name,
  sagemaker_session = NULL,
  model_channel_name = "model"
)
```

Arguments:

`training_job_name` (str): The name of the training job to attach to.

`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.

`model_channel_name` (str): Name of the channel where pre-trained model data will be downloaded (default: ’model’). If no channel with the same name exists in the training job, this option will be ignored.

Returns: Instance of the calling “Estimator“ Class with the attached training job.

Method `logs()`**:** Display the logs for Estimator’s training job. If the output is a tty or a Jupyter cell, it will be color-coded based on which instance the log entry is from.

Usage:

```
EstimatorBase$logs()
```

Method `deploy()`**:** Deploy the trained model to an Amazon SageMaker endpoint and return a “sagemaker.Predictor“ object. More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Usage:

```
EstimatorBase$deploy(
  initial_instance_count,
  instance_type,
  serializer = NULL,
  deserializer = NULL,
  accelerator_type = NULL,
  endpoint_name = NULL,
  use_compiled_model = FALSE,
  wait = TRUE,
  model_name = NULL,
  kms_key = NULL,
  data_capture_config = NULL,
  tags = NULL,
  ...
)
```

Arguments:

`initial_instance_count` (int): Minimum number of EC2 instances to deploy to an endpoint for prediction.

`instance_type` (str): Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.

`serializer` (:class:‘~sagemaker.serializers.BaseSerializer‘): A serializer object, used to encode data for an inference endpoint (default: None). If “serializer“ is not None, then “serializer“ will override the default serializer. The default serializer is set by the “predictor_cls“.

`deserializer` (:class:‘~sagemaker.deserializers.BaseDeserializer‘): A deserializer object, used to decode data from an inference endpoint (default: None). If “deserializer“ is not None, then “deserializer“ will override the default deserializer. The default deserializer is set by the “predictor_cls“.

`accelerator_type` (str): Type of Elastic Inference accelerator to attach to an endpoint for model loading and inference, for example, 'ml.eia1.medium'. If not specified, no Elastic Inference accelerator will be attached to the endpoint. For more information: https://docs.aws.amazon.com/sagemaker/l

`endpoint_name` (str): Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.

`use_compiled_model` (bool): Flag to select whether to use compiled (optimized) model. Default: False.

`wait` (bool): Whether the call should wait until the deployment of model completes (default: True).

`model_name` (str): Name to use for creating an Amazon SageMaker model. If not specified, the estimator generates a default job name based on the training image name and current timestamp.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the data on the storage volume attached to the instance hosting the endpoint.

`data_capture_config` (sagemaker.model_monitor.DataCaptureConfig): Specifies configuration related to Endpoint data capture for use with Amazon SageMaker Model Monitoring. Default: None.

`tags` (List[dict[str, str]]): Optional. The list of tags to attach to this specific endpoint. Example: »> tags = ['Key': 'tagname', 'Value': 'tagvalue'] For more information about tags, see https://boto3.amazonaws.com/v1/documentation\ /api/latest/reference/services/sagemaker.html#SageMaker.Client.ad

`...` : Passed to invocation of “create_model()“. Implementations may customize “create_model()“ to accept “**kwargs“ to customize model creation during deploy. For more, see the implementation docs.

Returns: sagemaker.predictor.Predictor: A predictor that provides a “predict()“ method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Method `register()`**:** Creates a model package for creating SageMaker models or listing on Marketplace.

Usage:

```
EstimatorBase$register(
  content_types,
  response_types,
  inference_instances,
  transform_instances,
  image_uri = NULL,
  model_package_name = NULL,
  model_package_group_name = NULL,
  model_metrics = NULL,
  metadata_properties = NULL,
  marketplace_cert = FALSE,
  approval_status = NULL,
  description = NULL,
  compile_model_family = NULL,
  model_name = NULL,
  drift_check_baselines = NULL,
  ...
)
```

Arguments:

`content_types` (list): The supported MIME types for the input data.

`response_types` (list): The supported MIME types for the output data.

`inference_instances` (list): A list of the instance types that are used to generate inferences in real-time.

`transform_instances` (list): A list of the instance types on which a transformation job can be run or on which an endpoint can be deployed.

`image_uri` (str): The container image uri for Model Package, if not specified, Estimator’s training container image will be used (default: None).

`model_package_name` (str): Model Package name, exclusive to ‘model_package_group_name‘, using ‘model_package_name‘ makes the Model Package un-versioned (default: None).

`model_package_group_name` (str): Model Package Group name, exclusive to ‘model_package_name‘, using ‘model_package_group_name‘ makes the Model Package versioned (default: None).

`model_metrics` (ModelMetrics): ModelMetrics object (default: None).

`metadata_properties` (MetadataProperties): MetadataProperties (default: None).

`marketplace_cert` (bool): A boolean value indicating if the Model Package is certified for AWS Marketplace (default: False).

`approval_status` (str): Model Approval Status, values can be "Approved", "Rejected", or "PendingManualApproval" (default: "PendingManualApproval").

`description` (str): Model Package description (default: None).

`compile_model_family` (str): Instance family for compiled model, if specified, a compiled model will be used (default: None).

`model_name` (str): User defined model name (default: None).

`drift_check_baselines` (DriftCheckBaselines): DriftCheckBaselines object (default: None).

`...` : Passed to invocation of “create_model()“. Implementations may customize “create_model()“ to accept “**kwargs“ to customize model creation during deploy. For more, see the implementation docs.

Returns: str: A string of SageMaker Model Package ARN.

Method `create_model()`**:** Create a SageMaker “Model“ object that can be deployed to an “Endpoint“.

Usage:

```
EstimatorBase$create_model(...)
```

Arguments:

`...` : Keyword arguments used by the implemented method for creating the “Model“.

Returns: sagemaker.model.Model: A SageMaker “Model“ object. See :func:‘~sagemaker.model.Model‘ for full details.

Method `delete_endpoint()`**:** Delete an Amazon SageMaker “Endpoint“.

Usage:

```
EstimatorBase$delete_endpoint()
```

Method `transformer()`**:** Return a “Transformer“ that uses a SageMaker Model based on the training job. It reuses the SageMaker Session and base job name used by the Estimator.

Usage:

```
EstimatorBase$transformer(
  instance_count,
  instance_type,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  env = NULL,
  max_concurrent_transforms = NULL,
  max_payload = NULL,
  tags = NULL,
  role = NULL,
  volume_kms_key = NULL,
  vpc_config_override = "VPC_CONFIG_DEFAULT",
  enable_network_isolation = NULL,
  model_name = NULL
)
```

Arguments:

`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: NULL). Valid values: ’MultiRecord’ and ’SingleRecord’.

`assemble_with` (str): How the output is assembled (default: NULL). Valid values: 'Line' or 'NULL'.

`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: NULL).

`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.

`env` (dict): Environment variables to be set for use during the transform job (default: NULL).

`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. If NULL specified, then the tags used for the training job are used for the transform job.

`role` (str): The "ExecutionRoleArn" IAM Role ARN for the "Model", which is also used during transform jobs. If not specified, the role from the Estimator will be used.

`volume_kms_key` (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: NULL).

`vpc_config_override` (dict[str, list[str]]): Optional override for the VpcConfig set on the model. Default: use subnets and security groups from this Estimator. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

`enable_network_isolation` (bool): Specifies whether container will run in network isolation mode. Network isolation mode restricts the container access to outside networks (such as the internet). The container does not make any inbound or outbound network calls. If True, a channel named "code" will be created for any user entry script for inference. Also known as Internet-free mode. If not specified, this setting is taken from the estimator's current configuration.

`model_name` (str): Name to use for creating an Amazon SageMaker model. If not specified, the name of the training job is used.

Method `get_vpc_config()`**:** Returns VpcConfig dict either from this Estimator's subnets and security groups, or else validate and return an optional override value.

Usage:

```
EstimatorBase$get_vpc_config(vpc_config_override = "VPC_CONFIG_DEFAULT")
```

Arguments:

`vpc_config_override` :

Method `.prepare_for_training()`**:** Set any values in the estimator that need to be set before training.

Usage:

```
EstimatorBase$.prepare_for_training(job_name = NULL)
```

Arguments:

`job_name` (str): Name of the training job to be created. If not specified, one is generated, using the base name given to the constructor if applicable.

Method `enable_default_profiling():` Update training job to enable Debugger monitoring. This method enables Debugger monitoring with the default “profiler_config“ parameter to collect system metrics and the default built-in “profiler_report“ rule. Framework metrics won’t be saved. To update training job to emit framework metrics, you can use :class:‘~sagemaker.estimator.Estimator.update_profiler‘ method and specify the framework metrics you want to enable. This method is callable when the training job is in progress while Debugger monitoring is disabled.

Usage:

```
EstimatorBase$enable_default_profiling()
```

Method `disable_profiling():` Update the current training job in progress to disable profiling. Debugger stops collecting the system and framework metrics and turns off the Debugger built-in monitoring and profiling rules.

Usage:

```
EstimatorBase$disable_profiling()
```

Method `update_profiler():` Update training jobs to enable profiling. This method updates the “profiler_config“ parameter and initiates Debugger built-in rules for profiling.

Usage:

```
EstimatorBase$update_profiler(
  rules = NULL,
  system_monitor_interval_millis = NULL,
  s3_output_path = NULL,
  framework_profile_params = NULL,
  disable_framework_metrics = FALSE
)
```

Arguments:

`rules` (list[:class:‘~sagemaker.debugger.ProfilerRule‘]): A list of :class:‘~sagemaker.debugger.ProfilerRule‘ objects to define rules for continuous analysis with SageMaker Debugger. Currently, you can only add new profiler rules during the training job. (default: “None“)

`system_monitor_interval_millis` (int): How often profiling system metrics are collected; Unit: Milliseconds (default: “None“)

`s3_output_path` (str): The location in S3 to store the output. If profiler is enabled once, s3_output_path cannot be changed. (default: “None“)

`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‘. (default: “None“)

`disable_framework_metrics` (bool): Specify whether to disable all the framework metrics. This won’t update system metrics and the Debugger built-in rules for monitoring. To stop both monitoring and profiling, use the :class:‘~sagemaker.estimator.Estimator.desable_profiling‘ method. (default: “False“)

Method `format():` format class

Usage:

```
EstimatorBase$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Note

Updating the profiling configuration for TensorFlow dataloader profiling is currently not available. If you started a TensorFlow training job only with monitoring and want to enable profiling while the training job is running, the dataloader profiling cannot be updated.

`ExplainabilityAnalysisConfig` *ExplainabilityAnalysisConfig class*

Description

Analysis configuration for ModelExplainabilityMonitor.

Public fields

`analysis_config` Analysis config dictionary

Methods

Public methods:

- `ExplainabilityAnalysisConfig$new()`
- `ExplainabilityAnalysisConfig$to_list()`
- `ExplainabilityAnalysisConfig$format()`
- `ExplainabilityAnalysisConfig$clone()`

Method `new()`**:** Creates an analysis config dictionary.

Usage:

```
ExplainabilityAnalysisConfig$new(
  explainability_config,
  model_config,
  headers = NULL
)
```

Arguments:

`explainability_config` (sagemaker.clarify.ExplainabilityConfig): Config object related to explainability configurations.

`model_config` (sagemaker.clarify.ModelConfig): Config object related to bias configurations.

`headers` (list[str]): A list of feature names (without label) of model/endpint input.

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

Usage:

```
ExplainabilityAnalysisConfig$to_list()
```

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

Usage:

```
ExplainabilityAnalysisConfig$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`FileSource` *FileSource*

Description

Accepts file source parameters for conversion to request dict.

Methods

Public methods:

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

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

Usage:

```
FileSource$new(s3_uri, content_digest = NULL, content_type = NULL)
```

Arguments:

`s3_uri` (str): The S3 URI of the metric

`content_digest` (str): The digest of the metric (default: None)

`content_type` (str): Specifies the type of content in S3 URI (default: None)

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

Usage:

```
FileSource$to_request_list()
```

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

Usage:

```
FileSource$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`FileSystemRecordSet` *FileSystemRecordSet Class*

Description

Amazon SageMaker channel configuration for a file system data source for Amazon algorithms.

Public fields

`feature_dim` The dimensionality of "values" arrays in the Record features

`num_records` The number of records in the set

`channel` The SageMaker Training Job channel this RecordSet should be bound to

Methods

Public methods:

- `FileSystemRecordSet$new()`
- `FileSystemRecordSet$print()`
- `FileSystemRecordSet$data_channel()`
- `FileSystemRecordSet$clone()`

Method `new()`**:** Initialize a “FileSystemRecordSet“ object.

Usage:

```
FileSystemRecordSet$new(
  file_system_id,
  file_system_type,
  directory_path,
  num_records,
  feature_dim,
  file_system_access_mode = "ro",
  channel = "train"
)
```

Arguments:

`file_system_id` (str): An Amazon file system ID starting with ’fs-’.

`file_system_type` (str): The type of file system used for the input. Valid values: 'EFS', 'FSxLustre'.

`directory_path` (str): Absolute or normalized path to the root directory (mount point) in the file system. Reference: https://docs.aws.amazon.com/efs/latest/ug/mounting-fs.html and https://docs.aws.amazon.com/efs/latest/ug/wt1-test.html

`num_records` (int): The number of records in the set.

`feature_dim` (int): The dimensionality of "values" arrays in the Record features, and label (if each Record is labeled).

`file_system_access_mode` (str): Permissions for read and write. Valid values: 'ro' or 'rw'. Defaults to 'ro'.

`channel` (str): The SageMaker Training Job channel this RecordSet should be bound to

Method `print()`**:** Return an unambiguous representation of this RecordSet

Return an unambiguous representation of this RecordSet

Usage:

```
FileSystemRecordSet$print()
```

Method `data_channel()`**:** Return a dictionary to represent the training data in a channel for use with “fit()“

Usage:

```
FileSystemRecordSet$data_channel()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Framework` *FrameWork Class*

Description

Base class that cannot be instantiated directly. Subclasses define functionality pertaining to specific ML frameworks, such as training/deployment images and predictor instances.

Super class

`sagemaker.mlcore::EstimatorBase` -> `Framework`

Public fields

`LAUNCH_PS_ENV_NAME` class metadata

`LAUNCH_MPI_ENV_NAME` class metadata

`LAUNCH_SM_DDP_ENV_NAME` class metadata

`INSTANCE_TYPE` class metadata

`MPI_NUM_PROCESSES_PER_HOST` class metadata

`MPI_CUSTOM_MPI_OPTIONS` class metadata

`SM_DDP_CUSTOM_MPI_OPTIONS` class metadata

`CONTAINER_CODE_CHANNEL_SOURCEDIR_PATH` class metadata

`.module` mimic python module

Methods

Public methods:

- `Framework$new()`
- `Framework$.prepare_for_training()`
- `Framework$hyperparameters()`
- `Framework$training_image_uri()`
- `Framework$attach()`
- `Framework$transformer()`
- `Framework$clone()`

Method `new()`**:** Base class initializer. Subclasses which override "__init__" should invoke "super()"

Usage:

```
Framework$new(
  entry_point,
  source_dir = NULL,
  hyperparameters = NULL,
  container_log_level = "INFO",
  code_location = NULL,
  image_uri = NULL,
  dependencies = NULL,
  enable_network_isolation = FALSE,
  git_config = NULL,
  checkpoint_s3_uri = NULL,
  checkpoint_local_path = NULL,
  enable_sagemaker_metrics = NULL,
  ...
)
```

Arguments:

`entry_point` (str): Path (absolute or relative) to the local Python source file which should be executed as the entry point to training. If “source_dir“ is specified, then “entry_point“ must point to a file located at the root of “source_dir“. If 'git_config' is provided, 'entry_point' should be a relative location to the Python source file in the Git repo. Example: With the following GitHub repo directory structure: »> |—– README.md »> |—– src »> |—– train.py »> |—– test.py You can assign entry_point='src/train.py'.

`source_dir` (str): Path (absolute, relative or an S3 URI) to a directory with any other training 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 training on Amazon SageMaker. If 'git_config' is provided, 'source_dir' should be a relative location to a directory in the Git repo. .. admonition:: Example With the following GitHub repo directory structure: »> |—– README.md »> |—– src »> |—– train.py »> |—– test.py and you need 'train.py' as entry point and 'test.py' as training source code as well, you can assign entry_point='train.py', source_dir='src'.

`hyperparameters` (dict): Hyperparameters that will be used for training (default: None). The hyperparameters are made accessible as a dict[str, str] to the training code on SageMaker. For convenience, this accepts other types for keys and values, but “str()“ will be called to convert them before training.

`container_log_level` (str): Log level to use within the container (default: "INFO")

`code_location` (str): The S3 prefix URI where custom code will be uploaded (default: None) - don't include a trailing slash since a string prepended with a "/" is appended to “code_location“. The code file uploaded to S3 is 'code_location/job-name/source/sourcedir.tar.gz'. If not specified, the default “code location“ is s3://output_bucket/job-name/.

`image_uri` (str): An alternate image name to use instead of the official Sagemaker image for the framework. This is useful to run one of the Sagemaker supported frameworks with an image containing custom dependencies.

`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. .. admonition:: Example The following call »> Estimator(entry_point='train.py', ... dependencies=['my/libs/common', 'virtual-env']) results in the following inside the container: »> $ ls »> opt/ml/code »> |—— train.py »> |—— common »> |—— virtual-env This is not supported with "local code" in Local Mode.

`enable_network_isolation` (bool): Specifies whether container will run in network isolation mode. Network isolation mode restricts the container access to outside networks (such as the internet). The container does not make any inbound or outbound network calls. If True, a channel named "code" will be created for any user entry script for training. The user entry script, files in source_dir (if specified), and dependencies will be uploaded in a tar to S3. Also known as internet-free mode (default: ‘False‘).

`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. .. admonition:: Example The following config: »> git_config = 'repo': 'https://github.com/aws/sagemaker-python-sdk.git', »> 'branch': 'test-branch-git-config', »> 'commit': '329bfcf884482002c05ff7f44f62599ebc9f445a' 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.

`checkpoint_s3_uri` (str): The S3 URI in which to persist checkpoints that the algorithm persists (if any) during training. (default: "None").

`checkpoint_local_path` (str): The local path that the algorithm writes its checkpoints to. SageMaker will persist all files under this path to 'checkpoint_s3_uri' continually during training. On job startup the reverse happens - data from the s3 location is downloaded to this path before the algorithm is started. If the path is unset then SageMaker assumes the checkpoints will be provided under '/opt/ml/checkpoints/'. (default: "None").

`enable_sagemaker_metrics` (bool): enable SageMaker Metrics Time Series. For more information see: https://docs.aws.amazon.com/sagemaker/latest/dg/API_AlgorithmSpecification.html#SageMaker-Type-AlgorithmSpecification-EnableSageMakerMetricsTimeSeries (default: "None").

`...` : Additional kwargs passed to the "EstimatorBase" constructor. .. tip:: You can find additional parameters for initializing this class at :class:'~sagemaker.estimator.EstimatorBase'.

Method `.prepare_for_training():` Set hyperparameters needed for training. This method will also validate "source_dir".

Usage:

```
Framework$.prepare_for_training(job_name = NULL)
```

Arguments:

`job_name` (str): Name of the training job to be created. If not specified, one is generated, using the base name given to the constructor if applicable.

Method `hyperparameters():` Return the hyperparameters as a dictionary to use for training. The :meth:'~sagemaker.estimator.EstimatorBase.fit' method, which trains the model, calls this method to find the hyperparameters.

Usage:

```
Framework$hyperparameters()
```

Returns: dict[str, str]: The hyperparameters.

Method `training_image_uri()`: Return the Docker image to use for training. The :meth:‘~sagemaker.estimator.Estimato
method, which does the model training, calls this method to find the image to use for model training.

Usage:

```
Framework$training_image_uri()
```

Returns: str: The URI of the Docker image.

Method `attach()`: Attach to an existing training job. Create an Estimator bound to an existing training job, each subclass is responsible to implement “_prepare_init_params_from_job_description()“ as this method delegates the actual conversion of a training job description to the arguments that the class constructor expects. After attaching, if the training job has a Complete status, it can be “deploy()“ ed to create a SageMaker Endpoint and return a “Predictor“. If the training job is in progress, attach will block and display log messages from the training job, until the training job completes. Examples: »> my_estimator.fit(wait=False) »> training_job_name = my_estimator.latest_training_job.name Later on: »> attached_estimator = Estimator.attach(training_job_name) »> attached_estimator.deploy()

Usage:

```
Framework$attach(
  training_job_name,
  sagemaker_session = NULL,
  model_channel_name = "model"
)
```

Arguments:

`training_job_name` (str): The name of the training job to attach to.

`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.

`model_channel_name` (str): Name of the channel where pre-trained model data will be downloaded (default: ’model’). If no channel with the same name exists in the training job, this option will be ignored.

Returns: Instance of the calling “Estimator“ Class with the attached training job.

Method `transformer()`: Return a “Transformer“ that uses a SageMaker Model based on the training job. It reuses the SageMaker Session and base job name used by the Estimator.

Usage:

```
Framework$transformer(
  instance_count,
  instance_type,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  env = NULL,
  max_concurrent_transforms = NULL,
  max_payload = NULL,
  tags = NULL,
  role = NULL,
  model_server_workers = NULL,
  volume_kms_key = NULL,
  entry_point = NULL,
  vpc_config_override = "VPC_CONFIG_DEFAULT",
  enable_network_isolation = NULL,
  model_name = NULL
)
```

Arguments:

`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.

`env` (dict): Environment variables to be set for use during the transform job (default: None).

`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. If none specified, then the tags used for the training job are used for the transform job.

`role` (str): The "ExecutionRoleArn" IAM Role ARN for the "Model", which is also used during transform jobs. If not specified, the role from the Estimator will be used.

`model_server_workers` (int): Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.

`volume_kms_key` (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

`entry_point` (str): Path (absolute or relative) to the local Python source file which should be executed as the entry point to training. If "source_dir" is specified, then "entry_point" must point to a file located at the root of "source_dir". If not specified, the training entry point is used.

`vpc_config_override` (dict[str, list[str]]): Optional override for the VpcConfig set on the model. Default: use subnets and security groups from this Estimator. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

`enable_network_isolation` (bool): Specifies whether container will run in network isolation mode. Network isolation mode restricts the container access to outside networks (such as the internet). The container does not make any inbound or outbound network calls. If True, a channel named "code" will be created for any user entry script for inference. Also known as Internet-free mode. If not specified, this setting is taken from the estimator's current configuration.

`model_name` (str): Name to use for creating an Amazon SageMaker model. If not specified, the name of the training job is used.

Returns: sagemaker.transformer.Transformer: a "Transformer" object that can be used to start a SageMaker Batch Transform job.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`FrameworkModel` *A Model for working with an SageMaker "Framework".*

Description

This class hosts user-defined code in S3 and sets code location and configuration in model environment variables.

Super classes

`sagemaker.mlcore::ModelBase` -> `sagemaker.mlcore::Model` -> `FrameworkModel`

Methods

Public methods:

- `FrameworkModel$new()`
- `FrameworkModel$prepare_container_def()`
- `FrameworkModel$clone()`

Method `new()`**:** Initialize a "FrameworkModel".

Usage:

```
FrameworkModel$new(
  model_data,
  image_uri,
  role,
  entry_point,
  source_dir = NULL,
  predictor_cls = NULL,
  env = NULL,
  name = NULL,
  container_log_level = "INFO",
  code_location = NULL,
  sagemaker_session = NULL,
  dependencies = NULL,
  git_config = NULL,
  ...
)
```

Arguments:

`model_data` (str): The S3 location of a SageMaker model data “.tar.gz“ file.

`image_uri` (str): A Docker image URI.

`role` (str): An IAM role name or ARN for SageMaker to access AWS resources on your behalf.

`entry_point` (str): Path (absolute or relative) to the Python source file which should be executed as the entry point to model hosting. This should be compatible with either Python 2.7 or Python 3.5. If 'git_config' is provided, 'entry_point' should be a relative location to the Python source file in the Git repo. Example With the following GitHub repo directory structure: »> |—— README.md »> |—— src »> |—— inference.py »> |—— test.py You can assign entry_point='src/inference.py'.

`source_dir` (str): Path (absolute, relative or an S3 URI) to a directory with any other training 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 training on Amazon SageMaker. If 'git_config' is provided, 'source_dir' should be a relative location to a directory in the Git repo. If the directory points to S3, no code will be uploaded and the S3 location will be used instead. .. admonition:: Example With the following GitHub repo directory structure: »> |—— README.md »> |—— src »> |—— inference.py »> |—— test.py You can assign entry_point='inference.py', source_dir='src'.

`predictor_cls` (callable[string, sagemaker.session.Session]): A function to call to create a predictor (default: None). If not None, “deploy“ will return the result of invoking this function on the created endpoint name.

`env` (dict[str, str]): Environment variables to run with “image“ when hosted in SageMaker (default: None).

`name` (str): The model name. If None, a default model name will be selected on each “deploy“.

`container_log_level` (str): Log level to use within the container (default: "INFO").

`code_location` (str): Name of the S3 bucket where custom code is uploaded (default: None). If not specified, default bucket created by “sagemaker.session.Session“ is used.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

`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. If the “'source_dir“' points to S3, code will be uploaded and the S3 location will be used instead. .. admonition:: Example The following call »> Estimator(entry_point='inference.py', dependencies=['my/libs/common', 'virtual-env']) results in the following inside the container: »> $ ls »> opt/ml/code »> |—— inference.py »> |—— common »> |—— virtual-env

`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. .. admonition:: Example The following config: »> git_config = 'repo': 'https://github.com/aws/sagemaker-python-sdk.git', »> 'branch': 'test-branch-git-config', »> 'commit': '329bfcf884482002c05ff7f44f62599ebc9f445a' 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.

`...` : Keyword arguments passed to the "Model" initializer.

Method `prepare_container_def():` Return a container definition with framework configuration set in model environment variables. This also uploads user-supplied code to S3.

Usage:

```
FrameworkModel$prepare_container_def(
  instance_type = NULL,
  accelerator_type = NULL
)
```

Arguments:

`instance_type` (str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.

`accelerator_type` (str): The Elastic Inference accelerator type to deploy to the instance for loading and making inferences to the model. For example, 'ml.eia1.medium'.

Returns: dict[str, str]: A container definition object usable with the CreateModel API.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Hyperparameter` *Hyperparameter Class*

Description

An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

Public fields

`validation` validation function

`validation_message` validation message

`name` name of hyperparameter validate

`data_type` function to convert data type

`obj` parent class to alter

Active bindings

`descriptor` active binding that mimic's python descriptor class

Methods

Public methods:

- `Hyperparameter$new()`
- `Hyperparameter$validate()`
- `Hyperparameter$serialize_all()`
- `Hyperparameter$clone()`

Method `new()`:

Usage:

```
Hyperparameter$new(
  name,
  validate = function(x) TRUE,
  validation_message = "",
  data_type = as.character,
  obj
)
```

Arguments:

`name` (str): The name of this hyperparameter validate

`validate` (callable[object]->[bool]): A validation function or list of validation functions. Each function validates an object and returns False if the object value is invalid for this hyperparameter.

`validation_message` (str): A usage guide to display on validation failure.

`data_type` : function to convert data types

`obj` (R6Class): R6Class for descriptor class to modify

Method `validate()`**:** Validate value

Usage:

```
Hyperparameter$validate(value = NULL)
```

Arguments:

`value` : values to be validated

Method `serialize_all()`**:** Return all non-None “hyperparameter“ values on “obj“ as a “dict[str,str].“

Usage:

```
Hyperparameter$serialize_all(obj)
```

Arguments:

`obj` : R object to be serialized

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`HyperparameterTuner` *HyperparamerTuner*

Description

A class for creating and interacting with Amazon SageMaker hyperparameter tuning jobs, as well as deploying the resulting model(s).

Public fields

`TUNING_JOB_NAME_MAX_LENGTH` Maximumn length of sagemaker job name

`SAGEMAKER_ESTIMATOR_MODULE` Class metadata

`SAGEMAKER_ESTIMATOR_CLASS_NAME` Class metadata

`DEFAULT_ESTIMATOR_MODULE` Class metadata

`DEFAULT_ESTIMATOR_CLS_NAME` Class metadata

Active bindings

`sagemaker_session` Convenience method for accessing the :class:‘~sagemaker.session.Session‘ object associated with the estimator for the “HyperparameterTuner“.

Methods

Public methods:

- `HyperparameterTuner$new()`
- `HyperparameterTuner$fit()`
- `HyperparameterTuner$attach()`
- `HyperparameterTuner$deploy()`
- `HyperparameterTuner$stop_tunning_job()`
- `HyperparameterTuner$describe()`
- `HyperparameterTuner$wait()`
- `HyperparameterTuner$best_estimator()`
- `HyperparameterTuner$best_training_job()`
- `HyperparameterTuner$delete_endpoint()`
- `HyperparameterTuner$hyperparameter_ranges()`
- `HyperparameterTuner$hyperparameter_ranges_list()`
- `HyperparameterTuner$analytics()`
- `HyperparameterTuner$transfer_learning_tuner()`
- `HyperparameterTuner$identical_dataset_and_algorithm_tuner()`
- `HyperparameterTuner$create()`
- `HyperparameterTuner$.add_estimator()`
- `HyperparameterTuner$format()`
- `HyperparameterTuner$clone()`

Method `new()`**:** Initialize a “HyperparameterTuner“. It takes an estimator to obtain configuration information for training jobs that are created as the result of a hyperparameter tuning job.

Usage:

```
HyperparameterTuner$new(
  estimator,
  objective_metric_name,
  hyperparameter_ranges,
  metric_definitions = NULL,
  strategy = "Bayesian",
  objective_type = "Maximize",
  max_jobs = 1,
  max_parallel_jobs = 1,
  tags = NULL,
  base_tuning_job_name = NULL,
  warm_start_config = NULL,
  early_stopping_type = c("Off", "Auto"),
  estimator_name = NULL
)
```

Arguments:

`estimator` (sagemaker.estimator.EstimatorBase): An estimator object that has been initialized with the desired configuration. There does not need to be a training job associated with this instance.

`objective_metric_name` (str): Name of the metric for evaluating training jobs.

`hyperparameter_ranges` (dict[str, sagemaker.parameter.ParameterRange]): Dictionary of parameter ranges. These parameter ranges can be one of three types: Continuous, Integer, or Categorical. The keys of the dictionary are the names of the hyperparameter, and the values are the appropriate parameter range class to represent the range.

`metric_definitions` (list[dict]): A list of dictionaries that defines the metric(s) used to evaluate the training jobs (default: None). Each dictionary contains two keys: 'Name' for the name of the metric, and 'Regex' for the regular expression used to extract the metric from the logs. This should be defined only for hyperparameter tuning jobs that don't use an Amazon algorithm.

`strategy` (str): Strategy to be used for hyperparameter estimations (default: 'Bayesian').

`objective_type` (str): The type of the objective metric for evaluating training jobs. This value can be either 'Minimize' or 'Maximize' (default: 'Maximize').

`max_jobs` (int): Maximum total number of training jobs to start for the hyperparameter tuning job (default: 1).

`max_parallel_jobs` (int): Maximum number of parallel training jobs to start (default: 1).

`tags` (list[dict]): List of tags for labeling the tuning job (default: None). For more, see https://docs.aws.amazon.com/sage

`base_tuning_job_name` (str): Prefix for the hyperparameter tuning job name when the :meth:'~sagemaker.tuner.Hyperpa
method launches. If not specified, a default job name is generated, based on the training image name and current timestamp.

`warm_start_config` (sagemaker.tuner.WarmStartConfig): A "WarmStartConfig" object that has been initialized with the configuration defining the nature of warm start tuning job.

`early_stopping_type` (str): Specifies whether early stopping is enabled for the job. Can be either 'Auto' or 'Off' (default: 'Off'). If set to 'Off', early stopping will not be attempted. If set to 'Auto', early stopping of some training jobs may happen, but is not guaranteed to.

`estimator_name` (str): A unique name to identify an estimator within the hyperparameter tuning job, when more than one estimator is used with the same tuning job (default: None).

Method `fit()`: Start a hyperparameter tuning job.

Usage:

```
HyperparameterTuner$fit(
  inputs = NULL,
  job_name = NULL,
  include_cls_metadata = FALSE,
  estimator_kwargs = NULL,
  wait = TRUE,
  ...
)
```

Arguments:

`inputs` : Information about the training data. Please refer to the "fit()" method of the associated estimator, as this can take any of the following forms: * (str) - The S3 location where training data is saved. * (dict[str, str] or dict[str, TrainingInput]) - If using multiple channels for training data, you can specify a dict mapping channel names to strings or :func:'~TrainingInput' objects. * (TrainingInput) - Channel configuration for S3 data sources that can provide additional information about the training dataset. See :func:'TrainingInput' for full details. * (sagemaker.session.FileSystemInput) - channel configuration for a file system data source that can provide additional information as well as the path to the training dataset. * (sagemaker.amazon.amazon_estimator.RecordSet) - A collection of Amazon :class:~'Record' objects serialized and stored in S3. For use with an estimator for an Amazon algorithm. * (sagemaker.amazon.amazon_estimator.FileSystemRecordSet) - Amazon SageMaker channel configuration for a file system data source for Amazon algorithms. * (list[sagemaker.amazon.amazon_estimator.RecordSet]) - A list of :class:~'sagemaker.amazon.amazon_estir objects, where each instance is a different channel of training data. * (list[sagemaker.amazon.amazon_estimator.FileS - A list of :class:~'sagemaker.amazon.amazon_estimator.FileSystemRecordSet' objects, where each instance is a different channel of training data.

`job_name` (str): Tuning job name. If not specified, the tuner generates a default job name, based on the training image name and current timestamp.

`include_cls_metadata` : It can take one of the following two forms. * (bool) - Whether or not the hyperparameter tuning job should include information about the estimator class (default: False). This information is passed as a hyperparameter, so if the algorithm you are using cannot handle unknown hyperparameters (e.g. an Amazon SageMaker built-in algorithm that does not have a custom estimator in the Python SDK), then set "include_cls_metadata" to "False". * (dict[str, bool]) - This version should be used for tuners created via the factory method create(), to specify the flag for each estimator provided in the estimator_dict argument of the method. The keys would be the same estimator names as in estimator_dict. If one estimator doesn't need the flag set, then no need to include it in the dictionary.

`estimator_kwargs` (dict[str, dict]): Dictionary for other arguments needed for training. Should be used only for tuners created via the factory method create(). The keys are the estimator names for the estimator_dict argument of create() method. Each value is a dictionary for the other arguments needed for training of the corresponding estimator.

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

`...` : Other arguments needed for training. Please refer to the "fit()" method of the associated estimator to see what other arguments are needed.

Method `attach()`**:** Attach to an existing hyperparameter tuning job. Create a HyperparameterTuner bound to an existing hyperparameter tuning job. After attaching, if there exists a best training job (or any other completed training job), that can be deployed to create an Amazon SageMaker Endpoint and return a "Predictor". The "HyperparameterTuner" instance could be created in one of the following two forms. * If the 'TrainingJobDefinition' field is present in tuning job description, the tuner will be created using the default constructor with a single estimator. * If the 'TrainingJobDefinitions' field (list) is present in tuning job description, the tuner will be created using the factory method "create()" with one or several estimators. Each estimator corresponds to one item in the 'TrainingJobDefinitions' field, while the estimator names would come from the 'DefinitionName' field of items in the 'TrainingJobDefinitions' field. For more details on how tuners are created from multiple estimators, see "create()" documentation. For more details on 'TrainingJobDefinition' and 'TrainingJobDefinitions' fields in tuning job description, see https://botocore.readthedocs.io/en/latest/reference/services/sagemaker.html#SageMaker.Client.create_hyper_parameter_tu

Usage:

```
HyperparameterTuner$attach(
  tuning_job_name,
  sagemaker_session = NULL,
  job_details = NULL,
  estimator_cls = NULL
)
```

Arguments:

`tuning_job_name` (str): The name of the hyperparameter tuning job to attach to.

`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.

`job_details` (dict): The response to a “DescribeHyperParameterTuningJob“ call. If not specified, the “HyperparameterTuner“ will perform one such call with the provided hyperparameter tuning job name.

`estimator_cls` : It can take one of the following two forms. (str): The estimator class name associated with the training jobs, e.g. 'sagemaker.estimator.Estimator'. If not specified, the “HyperparameterTuner“ will try to derive the correct estimator class from training job metadata, defaulting to :class:~'Estimator' if it is unable to determine a more specific class. (dict[str, str]): This form should be used only when the 'TrainingJobDefinitions' field (list) is present in tuning job description. In this scenario training jobs could be created from different training job definitions in the 'TrainingJobDefinitions' field, each of which would be mapped to a different estimator after the “attach()“ call. The “estimator_cls“ should then be a dictionary to specify estimator class names for individual estimators as needed. The keys should be the 'DefinitionName' value of items in 'TrainingJobDefinitions', which would be used as estimator names in the resulting tuner instance. # Example #1 - assuming we have the following tuning job description, which has the # 'TrainingJobDefinition' field present using a SageMaker built-in algorithm (i.e. PCA), # and “attach()“ can derive the estimator class from the training image. # So “estimator_cls“ would not be needed. # .. code:: R list('BestTrainingJob'= 'best_training_job_name', 'TrainingJobDefinition' = list('AlgorithmSpecification' = list('TrainingImage'= '174872318107.dkr.ecr.us-west-2.amazonaws.com/pca:1))) #»> my_tuner.fit() #»> job_name = my_tuner$latest_tuning_job$name #Later on: #»> attached_tuner = HyperparameterTuner.attach(job_name) #»> attached_tuner.deploy() #Example #2 - assuming we have the following tuning job description, which has a 2-item #list for the 'TrainingJobDefinitions' field. In this case 'estimator_cls' is only #needed for the 2nd item since the 1st item uses a SageMaker built-in algorithm #(i.e. PCA). #.. code:: R list('BestTrainingJob' = 'best_training_job_name', 'TrainingJobDefinitions'= list(list('DefinitionName'= 'estimator_pca', 'AlgorithmSpecification'= list('TrainingImage'= '174872318107.dkr.ecr.us-west-2.amazonaws.com/pca:1)), list('DefinitionName'= 'estimator_byoa', 'AlgorithmSpecification' = list('TrainingImage'= '123456789012.dkr.ecr.us-west-2.amazonaws.com/byoa:latest)))) »> my_tuner.fit() »> job_name = my_tuner.latest_tuning_job.name Later on: »> attached_tuner = HyperparameterTuner.attach(»> job_name, »> estimator_cls= »> 'estimator_byoa': 'org.byoa.Estimator' »>) »> attached_tuner.deploy()

Returns: sagemaker.tuner.HyperparameterTuner: A “HyperparameterTuner“ instance with the attached hyperparameter tuning job.

Method `deploy()`**:** Deploy the best trained or user specified model to an Amazon SageMaker endpoint and return a “sagemaker.Predictor“ object. For more information: http://docs.aws.amazon.com/sagemaker/latest/d
it-works-training.html

Usage:

```
HyperparameterTuner$deploy(
  initial_instance_count,
  instance_type,
  accelerator_type = NULL,
  endpoint_name = NULL,
  wait = TRUE,
  model_name = NULL,
  kms_key = NULL,
  data_capture_config = NULL,
  ...
)
```

Arguments:

`initial_instance_count` (int): Minimum number of EC2 instances to deploy to an endpoint for prediction.

`instance_type` (str): Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.

`accelerator_type` (str): Type of Elastic Inference accelerator to attach to an endpoint for model loading and inference, for example, 'ml.eia1.medium'. If not specified, no Elastic Inference accelerator will be attached to the endpoint. For more information: https://docs.aws.amazon.com/sagemaker/l

`endpoint_name` (str): Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.

`wait` (bool): Whether the call should wait until the deployment of model completes (default: True).

`model_name` (str): Name to use for creating an Amazon SageMaker model. If not specified, the name of the training job is used.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the data on the storage volume attached to the instance hosting the endpoint.

`data_capture_config` (sagemaker.model_monitor.DataCaptureConfig): Specifies configuration related to Endpoint data capture for use with Amazon SageMaker Model Monitoring. Default: None.

`...` : Other arguments needed for deployment. Please refer to the "create_model()" method of the associated estimator to see what other arguments are needed.

Returns: sagemaker.predictor.Predictor: A predictor that provides a "predict()" method, which can be used to send requests to the Amazon SageMaker endpoint and obtain inferences.

Method `stop_tunning_job():` Stop latest running hyperparameter tuning job.

Usage:

```
HyperparameterTuner$stop_tunning_job()
```

Method `describe():` Returns a response from the DescribeHyperParameterTuningJob API call.

Usage:

```
HyperparameterTuner$describe()
```

Method `wait():` Wait for latest hyperparameter tuning job to finish.

Usage:

```
HyperparameterTuner$wait()
```

Method `best_estimator():` Return the estimator that has best training job attached. The trained model can then be deployed to an Amazon SageMaker endpoint and return a "sagemaker.Predictor" object.

Usage:

```
HyperparameterTuner$best_estimator(best_training_job = NULL)
```

Arguments:

`best_training_job` (dict): Dictionary containing "TrainingJobName" and "TrainingJobDefinitionName". Example: .. code:: R list("TrainingJobName"= "my_training_job_name", "TrainingJobDefinitionName" "my_training_job_definition_name")

Returns: sagemaker.estimator.EstimatorBase: The estimator that has the best training job attached.

Method `best_training_job()`**:** Return name of the best training job for the latest hyperparameter tuning job.

Usage:

```
HyperparameterTuner$best_training_job()
```

Method `delete_endpoint()`**:** Delete an Amazon SageMaker endpoint. If an endpoint name is not specified, this defaults to looking for an endpoint that shares a name with the best training job for deletion.

Usage:

```
HyperparameterTuner$delete_endpoint(endpoint_name = NULL)
```

Arguments:

`endpoint_name` (str): Name of the endpoint to delete

Method `hyperparameter_ranges()`**:** Return the hyperparameter ranges in a dictionary to be used as part of a request for creating a hyperparameter tuning job.

Usage:

```
HyperparameterTuner$hyperparameter_ranges()
```

Method `hyperparameter_ranges_list()`**:** Return a dictionary of hyperparameter ranges for all estimators in "estimator_dict"

Usage:

```
HyperparameterTuner$hyperparameter_ranges_list()
```

Method `analytics()`**:** An instance of HyperparameterTuningJobAnalytics for this latest tuning job of this tuner. Analytics olbject gives you access to tuning results summarized into a pandas dataframe.

Usage:

```
HyperparameterTuner$analytics()
```

Method `transfer_learning_tuner()`**:** Creates a new "HyperparameterTuner" by copying the request fields from the provided parent to the new instance of "HyperparameterTuner". Followed by addition of warm start configuration with the type as "TransferLearning" and parents as the union of provided list of "additional_parents" and the "self". Also, training image in the new tuner's estimator is updated with the provided "training_image". Examples: »> parent_tuner = HyperparameterTuner.attach(tuning_job_name="parent-job-1") »> transfer_learning_tuner = parent_tuner.transfer_learning_tuner(»> additional_parents="parent-job-2") Later On: »> transfer_learning_tuner.fit(inputs=)

Usage:

```
HyperparameterTuner$transfer_learning_tuner(
  additional_parents = NULL,
  estimator = NULL
)
```

Arguments:

`additional_parents` (setstr): Set of additional parents along with the self to be used in warm starting

`estimator` (sagemaker.estimator.EstimatorBase): An estimator object that has been initialized with the desired configuration. There does not need to be a training job associated with this instance.

Returns: sagemaker.tuner.HyperparameterTuner: “HyperparameterTuner“ instance which can be used to launch transfer learning tuning job.

Method `identical_dataset_and_algorithm_tuner():` Creates a new “HyperparameterTuner“ by copying the request fields from the provided parent to the new instance of “HyperparameterTuner“. Followed by addition of warm start configuration with the type as "IdenticalDataAndAlgorithm" and parents as the union of provided list of “additional_parents“ and the “self“ Examples: »> parent_tuner = HyperparameterTuner.attach(tuning_job_name="parent-job-1") »> identical_dataset_algo_tuner = parent_tuner.identical_dataset_and_algorithm_tuner(»> additional_parents="parent-job-2") Later On: »> identical_dataset_algo_tuner.fit(inputs=)

Usage:

```
HyperparameterTuner$identical_dataset_and_algorithm_tuner(
  additional_parents = NULL
)
```

Arguments:

`additional_parents` (setstr): Set of additional parents along with the self to be used in warm starting

Returns: sagemaker.tuner.HyperparameterTuner: HyperparameterTuner instance which can be used to launch identical dataset and algorithm tuning job.

Method `create():` Factory method to create a “HyperparameterTuner“ instance. It takes one or more estimators to obtain configuration information for training jobs that are created as the result of a hyperparameter tuning job. The estimators are provided through a dictionary (i.e. “estimator_dict“) with unique estimator names as the keys. For individual estimators separate objective metric names and hyperparameter ranges should be provided in two dictionaries, i.e. “objective_metric_name_dict“ and “hyperparameter_ranges_dict“, with the same estimator names as the keys. Optional metrics definitions could also be provided for individual estimators via another dictionary “metric_definitions_dict“.

Usage:

```
HyperparameterTuner$create(
  estimator_list,
  objective_metric_name_list,
  hyperparameter_ranges_list,
  metric_definitions_list = NULL,
  base_tuning_job_name = NULL,
  strategy = "Bayesian",
  objective_type = "Maximize",
  max_jobs = 1,
  max_parallel_jobs = 1,
  tags = NULL,
  warm_start_config = NULL,
  early_stopping_type = "Off"
)
```

Arguments:

`estimator_list` (dict[str, sagemaker.estimator.EstimatorBase]): Dictionary of estimator instances that have been initialized with the desired configuration. There does not need to be a training job associated with the estimator instances. The keys of the dictionary would be referred to as "estimator names".

`objective_metric_name_list` (dict[str, str]): Dictionary of names of the objective metric for evaluating training jobs. The keys are the same set of estimator names as in “estimator_dict“, and there must be one entry for each estimator in “estimator_dict“.

`hyperparameter_ranges_list` (dict[str, dict[str, sagemaker.parameter.ParameterRange]]): Dictionary of tunable hyperparameter ranges. The keys are the same set of estimator names as in estimator_dict, and there must be one entry for each estimator in estimator_dict. Each value is a dictionary of sagemaker.parameter.ParameterRange instance, which can be one of three types: Continuous, Integer, or Categorical. The keys of each ParameterRange dictionaries are the names of the hyperparameter, and the values are the appropriate parameter range class to represent the range.

`metric_definitions_list` (dict(str, list[dict]])): Dictionary of metric definitions. The keys are the same set or a subset of estimator names as in estimator_dict, and there must be one entry for each estimator in estimator_dict. Each value is a list of dictionaries that defines the metric(s) used to evaluate the training jobs (default: None). Each of these dictionaries contains two keys: 'Name' for the name of the metric, and 'Regex' for the regular expression used to extract the metric from the logs. This should be defined only for hyperparameter tuning jobs that don't use an Amazon algorithm.

`base_tuning_job_name` (str): Prefix for the hyperparameter tuning job name when the :meth:'~sagemaker.tuner.Hyperpa method launches. If not specified, a default job name is generated, based on the training image name and current timestamp.

`strategy` (str): Strategy to be used for hyperparameter estimations (default: 'Bayesian').

`objective_type` (str): The type of the objective metric for evaluating training jobs. This value can be either 'Minimize' or 'Maximize' (default: 'Maximize').

`max_jobs` (int): Maximum total number of training jobs to start for the hyperparameter

`max_parallel_jobs` (int): Maximum number of parallel training jobs to start (default: 1).

`tags` (list[dict]): List of tags for labeling the tuning job (default: None). For more, see https://docs.aws.amazon.com/sage

`warm_start_config` (sagemaker.tuner.WarmStartConfig): A “WarmStartConfig“ object that has been initialized with the configuration defining the nature of warm start tuning job.

`early_stopping_type` (str): Specifies whether early stopping is enabled for the job. Can be either 'Auto' or 'Off' (default: 'Off'). If set to 'Off', early stopping will not be attempted. If set to 'Auto', early stopping of some training jobs may happen, but is not guaranteed to.

`tuning` job (default: 1).

Returns: sagemaker.tuner.HyperparameterTuner: a new “HyperparameterTuner“ object that can start a hyperparameter tuning job with one or more estimators.

Method `.add_estimator()`**:** Add an estimator with corresponding objective metric name, parameter ranges and metric definitions (if applicable). This method is called by other functions and isn't required to be called directly

Usage:

```
HyperparameterTuner$.add_estimator(
  estimator_name,
  estimator,
  objective_metric_name,
  hyperparameter_ranges,
  metric_definitions = NULL
)
```

Arguments:

`estimator_name` (str): A unique name to identify an estimator within the hyperparameter tuning job, when more than one estimator is used with the same tuning job (default: None).

`estimator` (sagemaker.estimator.EstimatorBase): An estimator object that has been initialized with the desired configuration. There does not need to be a training job associated with this instance.

`objective_metric_name` (str): Name of the metric for evaluating training jobs.

`hyperparameter_ranges` (dict[str, sagemaker.parameter.ParameterRange]): Dictionary of parameter ranges. These parameter ranges can be one of three types: Continuous, Integer, or Categorical. The keys of the dictionary are the names of the hyperparameter, and the values are the appropriate parameter range class to represent the range.

`metric_definitions` (list[dict]): A list of dictionaries that defines the metric(s) used to evaluate the training jobs (default: None). Each dictionary contains two keys: 'Name' for the name of the metric, and 'Regex' for the regular expression used to extract the metric from the logs. This should be defined only for hyperparameter tuning jobs that don't use an Amazon algorithm.

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

Usage:

```
HyperparameterTuner$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`IdentitySerializer` *Serialize data by returning data without modification.*

Description

This serializer may be useful if, for example, you're sending raw bytes such as from an image file's method.

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `IdentitySerializer`

Methods

Public methods:

- `IdentitySerializer$new()`
- `IdentitySerializer$serialize()`
- `IdentitySerializer$clone()`

Method `new()`**:** Initialize an "IdentitySerializer" instance.

Usage:

```
IdentitySerializer$new(content_type = "application/octet-stream")
```

Arguments:

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "application/octet-stream").

Method `serialize()`**:** Return data without modification.

Usage:

```
IdentitySerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized.

Returns: object: The unmodified data.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`IntegerParameter` *IntegerParameter Class*

Description

A class for representing hyperparameters that have an integer range of possible values.

Super class

`sagemaker.mlcore::ParameterRange` -> `IntegerParameter`

Public fields

`.name` Helps to categorise Class

Methods

Public methods:

- `IntegerParameter$new()`
- `IntegerParameter$cast_to_type()`
- `IntegerParameter$clone()`

Method `new()`**:** Initialize a IntegerParameter

Usage:

```
IntegerParameter$new(
  min_value,
  max_value,
  scaling_type = c("Auto", "Linear", "Logarithmic", "ReverseLogarithmic")
)
```

Arguments:

`min_value` (int): The minimum value for the range.

`max_value` (int): The maximum value for the range.

`scaling_type` (str): The scale used for searching the range during tuning (default: 'Auto'). Valid values: 'Auto', 'Linear', Logarithmic' and 'ReverseLogarithmic'.

Method `cast_to_type()`**:** cast value to integer

Usage:

```
IntegerParameter$cast_to_type(value)
```

Arguments:

`value` The value to be verified.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`JSONDeserializer` *JSONDeserializer Class*

Description

Deserialize JSON data from an inference endpoint into a R object.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `JSONDeserializer`

Methods

Public methods:

- `JSONDeserializer$new()`
- `JSONDeserializer$deserialize()`
- `JSONDeserializer$clone()`

Method `new()`**:** Initialize a “JSONDeserializer“ instance.

Usage:

```
JSONDeserializer$new(accept = "application/json")
```

Arguments:

`accept` (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: "application/json").

Method `deserialize()`**:** Deserialize JSON data from an inference endpoint into a Python object.

Usage:

```
JSONDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` (botocore.response.StreamingBody): Data to be deserialized.

`content_type` (str): The MIME type of the data.

Returns: object: The JSON-formatted data deserialized into a R object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

JSONLinesDeserializer *JSONDeserializer Class*

Description

Deserialize JSON lines data from an inference endpoint.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `JSONDeserializer`

Methods

Public methods:

- `JSONLinesDeserializer$new()`
- `JSONLinesDeserializer$deserialize()`
- `JSONLinesDeserializer$clone()`

Method `new()`**:** Initialize a “JSONLinesDeserializer“ instance.

Usage:

```
JSONLinesDeserializer$new(accept = "application/json")
```

Arguments:

`accept` (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: ("text/csv","application/json")).

Method `deserialize()`**:** Deserialize JSON lines data from an inference endpoint. See https://docs.python.org/3/library/js to-json-table to understand how JSON values are converted to R objects.

Usage:

```
JSONLinesDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` (botocore.response.StreamingBody): Data to be deserialized.

`content_type` (str): The MIME type of the data.

Returns: list: A list of JSON serializable objects.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`JSONLinesSerializer` *JSONLinesSerializer Class*

Description

Serialize data to a JSON Lines formatted string.

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `IdentitySerializer`

Methods

Public methods:

- `JSONLinesSerializer$new()`
- `JSONLinesSerializer$serialize()`
- `JSONLinesSerializer$clone()`

Method `new()`**:** Initialize a “JSONLinesSerializer“ instance.

Usage:

```
JSONLinesSerializer$new(content_type = "application/jsonlines")
```

Arguments:

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "application/jsonlines").

Method `serialize()`**:** Serialize data of various formats to a JSON Lines formatted string.

Usage:

```
JSONLinesSerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized. The data can be a string, iterable of JSON serializable objects, or a file-like object.

Returns: str: The data serialized as a string containing newline-separated JSON values.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`JSONSerializer` *JSONSerializer Class*

Description

Serialize data to a JSON formatted string.

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `JSONSerializer`

Methods

Public methods:

- `JSONSerializer$serialize()`
- `JSONSerializer$clone()`

Method `serialize()`**:** Serialize data of various formats to a JSON formatted string.

Usage:

```
JSONSerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized.

Returns: (raw): The data serialized as a JSON string.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`LibSVMSerializer` *LibSVMSerializer Class*

Description

Serialize data of various formats to a LibSVM-formatted string. The data must already be in LIBSVM file format: <label> <index1>:<value1> <index2>:<value2> ... It is suitable for sparse datasets since it does not store zero-valued features.

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `LibSVMSerializer`

Methods

Public methods:

- `LibSVMSerializer$new()`
- `LibSVMSerializer$serialize()`
- `LibSVMSerializer$clone()`

Method `new()`**:** Initialize a “LibSVMSerializer“ instance.

Usage:

```
LibSVMSerializer$new(content_type = "text/libsvm")
```

Arguments:

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "text/libsvm").

Method `serialize()`**:** Serialize data of various formats to a LibSVM-formatted string.

Usage:

```
LibSVMSerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized. Can be a string, a file-like object, sparse matrix, or a list (format: list(<sparse matrix>, <label>)).

Returns: str: The data serialized as a LibSVM-formatted string.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`MetricsSource` *MetricsSource class*

Description

Accepts metrics source parameters for conversion to request dict.

Methods

Public methods:

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

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

Usage:

```
MetricsSource$new(content_type, s3_uri, content_digest = NULL)
```

Arguments:

`content_type` (str): Specifies the type of content in S3 URI

`s3_uri` (str): The S3 URI of the metric

`content_digest` (str): The digest of the metric (default: None)

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

Usage:

```
MetricsSource$to_request_list()
```

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

Usage:

```
MetricsSource$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Model` *Model Class*

Description

A SageMaker “Model“ that can be deployed to an “Endpoint“.

Super class

`sagemaker.mlcore::ModelBase` -> `Model`

Methods

Public methods:

- `Model$new()`
- `Model$register()`
- `Model$prepare_container_def()`
- `Model$enable_network_isolation()`
- `Model$check_neo_region()`
- `Model$package_for_edge()`
- `Model$compile()`
- `Model$deploy()`
- `Model$transformer()`
- `Model$delete_model()`
- `Model$.create_sagemaker_model()`
- `Model$clone()`

Method `new()`**:** Creates a new instance of this [R6][R6::R6Class] class.

Usage:

```
Model$new(
  image_uri,
  model_data = NULL,
  role = NULL,
  predictor_cls = NULL,
  env = NULL,
  name = NULL,
  vpc_config = NULL,
  sagemaker_session = NULL,
  enable_network_isolation = FALSE,
  model_kms_key = NULL,
  image_config = NULL
)
```

Arguments:

`image_uri` (str): A Docker image URI.

`model_data` (str): The S3 location of a SageMaker model data ".tar.gz" file.

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role if it needs to access some AWS resources. It can be null if this is being used to create a Model to pass to a "PipelineModel" which has its own Role field. (Default: NULL)

`predictor_cls` (callable[string, :Session]): A function to call to create a predictor (default: None). If not None, "deploy" will return the result of invoking this function on the created endpoint name.

`env` (dict[str, str]): Environment variables to run with "image" when hosted in SageMaker (Default: NULL).

`name` (str): The model name. If None, a default model name will be selected on each "deploy".

`vpc_config` (dict[str, list[str]]): The VpcConfig set on the model (Default: NULL)

- **'Subnets' (list[str])** List of subnet ids.
- **'SecurityGroupIds' (list[str]):** List of security group ids.

`sagemaker_session` (:Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

`enable_network_isolation` (Boolean): Default False. if True, enables network isolation in the endpoint, isolating the model container. No inbound or outbound network calls can be made to or from the model container.

`model_kms_key` (str): KMS key ARN used to encrypt the repacked model archive file if the model is repacked

`image_config` (dict[str, str]): Specifies whether the image of model container is pulled from ECR, or private registry in your VPC. By default it is set to pull model container image from ECR. (default: None).

Method `register()`**:** Creates a model package for creating SageMaker models or listing on Marketplace.

Usage:

```
Model$register(
  content_types,
  response_types,
  inference_instances,
  transform_instances,
  model_package_name = NULL,
  model_package_group_name = NULL,
  image_uri = NULL,
  model_metrics = NULL,
  metadata_properties = NULL,
  marketplace_cert = FALSE,
  approval_status = NULL,
  description = NULL,
  drift_check_baselines = NULL
)
```

Arguments:

`content_types` (list): The supported MIME types for the input data (default: None).

`response_types` (list): The supported MIME types for the output data (default: None).

`inference_instances` (list): A list of the instance types that are used to generate inferences in real-time (default: None).

`transform_instances` (list): A list of the instance types on which a transformation job can be run or on which an endpoint can be deployed (default: None).

`model_package_name` (str): Model Package name, exclusive to 'model_package_group_name', using 'model_package_name' makes the Model Package un-versioned (default: None).

`model_package_group_name` (str): Model Package Group name, exclusive to 'model_package_name', using 'model_package_group_name' makes the Model Package versioned (default: None).

`image_uri` (str): Inference image uri for the container. Model class' self.image will be used if it is None (default: None).

`model_metrics` (ModelMetrics): ModelMetrics object (default: None).

`metadata_properties` (MetadataProperties): MetadataProperties object (default: None).

`marketplace_cert` (bool): A boolean value indicating if the Model Package is certified for AWS Marketplace (default: False).

`approval_status` (str): Model Approval Status, values can be "Approved", "Rejected", or "PendingManualApproval" (default: "PendingManualApproval").

`description` (str): Model Package description (default: None).

`drift_check_baselines` (DriftCheckBaselines): DriftCheckBaselines object (default: None).

Returns: str: A string of SageMaker Model Package ARN.

Method `prepare_container_def():` Return a dict created by "sagemaker.container_def()" for deploying this model to a specified instance type. Subclasses can override this to provide custom container definitions for deployment to a specific instance type. Called by "deploy()".

Usage:

```
Model$prepare_container_def(instance_type, accelerator_type = NULL)
```

Arguments:

`instance_type` (str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.

`accelerator_type` (str): The Elastic Inference accelerator type to deploy to the instance for loading and making inferences to the model. For example, 'ml.eia1.medium'.

Returns: dict: A container definition object usable with the CreateModel API.

Method `enable_network_isolation():` Whether to enable network isolation when creating this Model

Usage:

```
Model$enable_network_isolation()
```

Returns: bool: If network isolation should be enabled or not.

Method `check_neo_region():` Check if this "Model" in the available region where neo support.

Usage:

```
Model$check_neo_region(region)
```

Arguments:

`region` (str): Specifies the region where want to execute compilation

Returns: bool: boolean value whether if neo is available in the specified region

Method `package_for_edge()`**:** Package this “Model“ with SageMaker Edge. Creates a new EdgePackagingJob and wait for it to finish. model_data will now point to the packaged artifacts.

Usage:

```
Model$package_for_edge(
  output_path,
  model_name,
  model_version,
  role = NULL,
  job_name = NULL,
  resource_key = NULL,
  s3_kms_key = NULL,
  tags = NULL
)
```

Arguments:

`output_path` (str): Specifies where to store the packaged model

`model_name` (str): the name to attach to the model metadata

`model_version` (str): the version to attach to the model metadata

`role` (str): Execution role

`job_name` (str): The name of the edge packaging job

`resource_key` (str): the kms key to encrypt the disk with

`s3_kms_key` (str): the kms key to encrypt the output with

`tags` (list[dict]): List of tags for labeling an edge packaging job. For more, see https://docs.aws.amazon.com/sagemaker/l

Returns: sagemaker.model.Model: A SageMaker “Model“ object. See :func:‘~sagemaker.model.Model‘ for full details.

Method `compile()`**:** Compile this “Model“ with SageMaker Neo.

Usage:

```
Model$compile(
  target_instance_family,
  input_shape,
  output_path,
  role,
  tags = NULL,
  job_name = NULL,
  compile_max_run = 5 * 60,
  framework = NULL,
  framework_version = NULL,
  target_platform_os = NULL,
  target_platform_arch = NULL,
  target_platform_accelerator = NULL,
  compiler_options = NULL
)
```

Arguments:

`target_instance_family` (str): Identifies the device that you want to run your model after compilation, for example: ml_c5. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Ou

`input_shape` (list): Specifies the name and shape of the expected inputs for your trained model in json dictionary form, for example: `list('data'= list(1,3,1024,1024)), or list('var1'= list(1,1,28,28),'var2'= list(1,1,28,28))`

`output_path` (str): Specifies where to store the compiled model

`role` (str): Execution role

`tags` (list[dict]): List of tags for labeling a compilation job. For more, see https://docs.aws.amazon.com/sagemaker/latest/

`job_name` (str): The name of the compilation job

`compile_max_run` (int): Timeout in seconds for compilation (default: 3 * 60). After this amount of time Amazon SageMaker Neo terminates the compilation job regardless of its current status.

`framework` (str): The framework that is used to train the original model. Allowed values: 'mxnet', 'tensorflow', 'keras', 'pytorch', 'onnx', 'xgboost'

`framework_version` (str):

`target_platform_os` (str): Target Platform OS, for example: 'LINUX'. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_OutputConfig.html. It can be used instead of target_instance_family.

`target_platform_arch` (str): Target Platform Architecture, for example: 'X86_64'. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_OutputConfig.html. It can be used instead of target_instance_family.

`target_platform_accelerator` (str, optional): Target Platform Accelerator, for example: 'NVIDIA'. For allowed strings see https://docs.aws.amazon.com/sagemaker/latest/dg/API_OutputConfig.html. It can be used instead of target_instance_family.

`compiler_options` (dict, optional): Additional parameters for compiler. Compiler Options are TargetPlatform / target_instance_family specific. See https://docs.aws.amazon.com/sagemaker/latest/dg/API_Output for details.

Returns: sagemaker.model.Model: A SageMaker “Model“ object. See :func:‘~sagemaker.model.Model‘ for full details.

Method `deploy()`**:** Deploy this “Model“ to an “Endpoint“ and optionally return a “Predictor“. Create a SageMaker “Model“ and “EndpointConfig“, and deploy an “Endpoint“ from this “Model“. If “self.predictor_cls“ is not None, this method returns a the result of invoking “self.predictor_cls“ on the created endpoint name. The name of the created model is accessible in the “name“ field of this “Model“ after deploy returns The name of the created endpoint is accessible in the “endpoint_name“ field of this “Model“ after deploy returns.

Usage:

```
Model$deploy(
  initial_instance_count = NULL,
  instance_type = NULL,
  serializer = NULL,
  deserializer = NULL,
  accelerator_type = NULL,
  endpoint_name = NULL,
  tags = NULL,
  kms_key = NULL,
  wait = TRUE,
  data_capture_config = NULL,
  serverless_inference_config = NULL,
  ...
)
```

Arguments:

`initial_instance_count` (int): The initial number of instances to run in the “Endpoint“ created from this “Model“.

`instance_type` (str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge', or 'local' for local mode.

`serializer` (:class:‘~sagemaker.serializers.BaseSerializer‘): A serializer object, used to encode data for an inference endpoint (default: None). If “serializer“ is not None, then “serializer“ will override the default serializer. The default serializer is set by the “predictor_cls“.

`deserializer` (:class:‘~sagemaker.deserializers.BaseDeserializer‘): A deserializer object, used to decode data from an inference endpoint (default: None). If “deserializer“ is not None, then “deserializer“ will override the default deserializer. The default deserializer is set by the “predictor_cls“.

`accelerator_type` (str): Type of Elastic Inference accelerator to deploy this model for model loading and inference, for example, 'ml.eia1.medium'. If not specified, no Elastic Inference accelerator will be attached to the endpoint. For more information: https://docs.aws.amazon.com/sagemaker/latest/dg

`endpoint_name` (str): The name of the endpoint to create (Default: NULL). If not specified, a unique endpoint name will be created.

`tags` (List[dict[str, str]]): The list of tags to attach to this specific endpoint.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the data on the storage volume attached to the instance hosting the endpoint.

`wait` (bool): Whether the call should wait until the deployment of this model completes (default: True).

`data_capture_config` (sagemaker.model_monitor.DataCaptureConfig): Specifies configuration related to Endpoint data capture for use with Amazon SageMaker Model Monitoring. Default: None.

`serverless_inference_config` (sagemaker.serverless.ServerlessInferenceConfig): Specifies configuration related to serverless endpoint. Use this configuration when trying to create serverless endpoint and make serverless inference. If empty object passed through, we will use pre-defined values in “ServerlessInferenceConfig“ class to deploy serverless endpoint (default: None)

`...` : pass deprecated parameters.

Returns: callable[string, sagemaker.session.Session] or None: Invocation of “self.predictor_cls“ on the created endpoint name, if “self.predictor_cls“ is not None. Otherwise, return None.

Method `transformer()`**:** Return a “Transformer“ that uses this Model.

Usage:

```
Model$transformer(
  instance_count,
  instance_type,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  env = NULL,
  max_concurrent_transforms = NULL,
  max_payload = NULL,
  tags = NULL,
  volume_kms_key = NULL
)
```

Arguments:

`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.

`env` (dict): Environment variables to be set for use during the transform job (default: None).

`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. If none specified, then the tags used for the training job are used for the transform job.

`volume_kms_key` (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

Method `delete_model()`**:** Delete an Amazon SageMaker Model.

Usage:

```
Model$delete_model()
```

Method `.create_sagemaker_model()`**:** Create a SageMaker Model Entity

Usage:

```
Model$.create_sagemaker_model(
  instance_type,
  accelerator_type = NULL,
  tags = NULL
)
```

Arguments:

`instance_type` (str): The EC2 instance type that this Model will be used for, this is only used to determine if the image needs GPU support or not.

`accelerator_type` (str): Type of Elastic Inference accelerator to attach to an endpoint for model loading and inference, for example, 'ml.eia1.medium'. If not specified, no Elastic Inference accelerator will be attached to the endpoint.

`tags` (List[dict[str, str]]): Optional. The list of tags to add to the model. Example: »> tags = ['Key': 'tagname', 'Value': 'tagvalue'] For more information about tags, see `https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sagemaker.html#SageMaker.Client.add_tags`.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ModelBiasMonitor` *Amazon SageMaker model monitor to monitor bias metrics of an endpoint.*

Description

Please see the 'initialize' method of its base class for how to instantiate it.

Super classes

`sagemaker.mlcore::ModelMonitor` -> `sagemaker.mlcore::ClarifyModelMonitor` -> `ModelBiasMonitor`

Public fields

`JOB_DEFINITION_BASE_NAME` Model definition base name

Methods

Public methods:

- `ModelBiasMonitor$monitoring_type()`
- `ModelBiasMonitor$suggest_baseline()`
- `ModelBiasMonitor$create_monitoring_schedule()`
- `ModelBiasMonitor$update_monitoring_schedule()`
- `ModelBiasMonitor$delete_monitoring_schedule()`
- `ModelBiasMonitor$attach()`
- `ModelBiasMonitor$clone()`

Method `monitoring_type()`**:** Type of the monitoring job.

Usage:

```
ModelBiasMonitor$monitoring_type()
```

Method `suggest_baseline()`**:** Suggests baselines for use with Amazon SageMaker Model Monitoring Schedules.

Usage:

```
ModelBiasMonitor$suggest_baseline(
  data_config,
  bias_config,
  model_config,
  model_predicted_label_config = NULL,
  wait = FALSE,
  logs = FALSE,
  job_name = NULL,
  kms_key = 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.

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

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

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

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

Returns: sagemaker.processing.ProcessingJob: The ProcessingJob object representing the baselining job.

Method `create_monitoring_schedule()`**:** Creates a monitoring schedule.

Usage:

```
ModelBiasMonitor$create_monitoring_schedule(
  endpoint_input,
  ground_truth_input,
  analysis_config = NULL,
  output_s3_uri = NULL,
  constraints = NULL,
  monitor_schedule_name = NULL,
  schedule_cron_expression = NULL,
  enable_cloudwatch_metrics = TRUE
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`ground_truth_input` (str): S3 URI to ground truth dataset.

`analysis_config` (str or BiasAnalysisConfig): URI to analysis_config for the bias job. If it is None then configuration of the latest baselining job will be reused, but if no baselining job then fail the call.

`output_s3_uri` (str): S3 destination of the constraint_violations and analysis result. Default: "s3://<default_session_bucket>/<job_name>/output"

`constraints` (sagemaker.model_monitor.Constraints or str): If provided it will be used for monitoring the endpoint. It can be a Constraints object or an S3 uri pointing to a constraints JSON file.

`monitor_schedule_name` (str): Schedule name. If not specified, the processor generates a default job name, based on the image name and current timestamp.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job run. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

Method `update_monitoring_schedule()`**:** Updates the existing monitoring schedule. If more options than schedule_cron_expression are to be updated, a new job definition will be created to hold them. The old job definition will not be deleted.

Usage:

```
ModelBiasMonitor$update_monitoring_schedule(
  endpoint_input = NULL,
  ground_truth_input = NULL,
  analysis_config = NULL,
  output_s3_uri = NULL,
  constraints = NULL,
  schedule_cron_expression = NULL,
  enable_cloudwatch_metrics = NULL,
  role = NULL,
  instance_count = NULL,
  instance_type = NULL,
  volume_size_in_gb = NULL,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  env = NULL,
  network_config = NULL
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`ground_truth_input` (str): S3 URI to ground truth dataset.

`analysis_config` (str or BiasAnalysisConfig): URI to analysis_config for the bias job. If it is None then configuration of the latest baselining job will be reused, but if no baselining job then fail the call.

`output_s3_uri` (str): S3 destination of the constraint_violations and analysis result. Default: "s3://<default_session_bucket>/<job_name>/output"

`constraints` (sagemaker.model_monitor.Constraints or str): If provided it will be used for monitoring the endpoint. It can be a Constraints object or an S3 uri pointing to a constraints JSON file.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job run. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

`role` (str): An AWS IAM role. The Amazon SageMaker jobs use this role.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

`env` (dict): Environment variables to be passed to the job.

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

Method `delete_monitoring_schedule()`**:** Deletes the monitoring schedule and its job definition.

Usage:

```
ModelBiasMonitor$delete_monitoring_schedule()
```

Method `attach()`**:** Sets this object's schedule name to the name provided. This allows subsequent describe_schedule or list_executions calls to point to the given schedule.

Usage:

```
ModelBiasMonitor$attach(monitor_schedule_name, sagemaker_session = NULL)
```

Arguments:

`monitor_schedule_name` (str): The name of the schedule to attach to.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

ModelExplainabilityMonitor

Amazon SageMaker model monitor to monitor feature attribution of an endpoint.

Description

Please see the ‘initiliaze‘ method of its base class for how to instantiate it.

Super classes

`sagemaker.mlcore::ModelMonitor` -> `sagemaker.mlcore::ClarifyModelMonitor` -> `ModelExplainabilityMonitor`

Public fields

`JOB_DEFINITION_BASE_NAME` Model definition base name

Methods

Public methods:

- `ModelExplainabilityMonitor$monitoring_type()`
- `ModelExplainabilityMonitor$suggest_baseline()`
- `ModelExplainabilityMonitor$create_monitoring_schedule()`
- `ModelExplainabilityMonitor$update_monitoring_schedule()`
- `ModelExplainabilityMonitor$delete_monitoring_schedule()`
- `ModelExplainabilityMonitor$attach()`
- `ModelExplainabilityMonitor$clone()`

Method `monitoring_type()`: Type of the monitoring job.

Usage:

```
ModelExplainabilityMonitor$monitoring_type()
```

Method `suggest_baseline()`: Suggest baselines for use with Amazon SageMaker Model Monitoring Schedules.

Usage:

```
ModelExplainabilityMonitor$suggest_baseline(
  data_config,
  explainability_config,
  model_config,
  model_scores = NULL,
  wait = FALSE,
  logs = FALSE,
  job_name = NULL,
  kms_key = NULL
)
```

Arguments:

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

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

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

`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: False).

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

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

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

Returns: sagemaker.processing.ProcessingJob: The ProcessingJob object representing the baselining job.

Method `create_monitoring_schedule()`**:** Creates a monitoring schedule.

Usage:

```
ModelExplainabilityMonitor$create_monitoring_schedule(
  endpoint_input,
  analysis_config = NULL,
  output_s3_uri = NULL,
  constraints = NULL,
  monitor_schedule_name = NULL,
  schedule_cron_expression = NULL,
  enable_cloudwatch_metrics = TRUE
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`analysis_config` (str or ExplainabilityAnalysisConfig): URI to the analysis_config for the explainability job. If it is None then configuration of the latest baselining job will be reused, but if no baselining job then fail the call.

`output_s3_uri` (str): S3 destination of the constraint_violations and analysis result. Default: "s3://<default_session_bucket>/<job_name>/output"

`constraints` (sagemaker.model_monitor.Constraints or str): If provided it will be used for monitoring the endpoint. It can be a Constraints object or an S3 uri pointing to a constraints JSON file.

`monitor_schedule_name` (str): Schedule name. If not specified, the processor generates a default job name, based on the image name and current timestamp.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job run. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

Method `update_monitoring_schedule()`**:** Updates the existing monitoring schedule. If more options than schedule_cron_expression are to be updated, a new job definition will be created to hold them. The old job definition will not be deleted.

Usage:

```
ModelExplainabilityMonitor$update_monitoring_schedule(
  endpoint_input = NULL,
  analysis_config = NULL,
  output_s3_uri = NULL,
  constraints = NULL,
  schedule_cron_expression = NULL,
  enable_cloudwatch_metrics = NULL,
  role = NULL,
  instance_count = NULL,
  instance_type = NULL,
  volume_size_in_gb = NULL,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  env = NULL,
  network_config = NULL
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`analysis_config` (str or BiasAnalysisConfig): URI to analysis_config for the bias job. If it is None then configuration of the latest baselining job will be reused, but if no baselining job then fail the call.

`output_s3_uri` (str): S3 destination of the constraint_violations and analysis result. Default: "s3://<default_session_bucket>/<job_name>/output"

`constraints` (sagemaker.model_monitor.Constraints or str): If provided it will be used for monitoring the endpoint. It can be a Constraints object or an S3 uri pointing to a constraints JSON file.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job run. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

`role` (str): An AWS IAM role. The Amazon SageMaker jobs use this role.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

`env` (dict): Environment variables to be passed to the job.

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

Method `delete_monitoring_schedule()`**:** Deletes the monitoring schedule and its job definition.

Usage:

```
ModelExplainabilityMonitor$delete_monitoring_schedule()
```

Method `attach()`**:** Sets this object's schedule name to the name provided. This allows subsequent describe_schedule or list_executions calls to point to the given schedule.

Usage:

```
ModelExplainabilityMonitor$attach(
  monitor_schedule_name,
  sagemaker_session = NULL
)
```

Arguments:

`monitor_schedule_name` (str): The name of the schedule to attach to.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ModelMetrics` *ModelMetrics class*

Description

Accepts model metrics parameters for conversion to request dict.

Methods

Public methods:

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

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

Usage:

```
ModelMetrics$new(
  model_statistics = NULL,
  model_constraints = NULL,
  model_data_statistics = NULL,
  model_data_constraints = NULL,
  bias = NULL,
  explainability = NULL,
  bias_pre_training = NULL,
  bias_post_training = NULL
)
```

Arguments:

`model_statistics` (MetricsSource): A metric source object that represents model statistics (default: None).

`model_constraints` (MetricsSource): A metric source object that represents model constraints (default: None).

`model_data_statistics` (MetricsSource): A metric source object that represents model data statistics (default: None).

`model_data_constraints` (MetricsSource): A metric source object that represents model data constraints (default: None).

`bias` (MetricsSource): A metric source object that represents bias report (default: None).

`explainability` (MetricsSource): A metric source object that represents explainability report (default: None).

`bias_pre_training` (MetricsSource): A metric source object that represents Pre-training report (default: None).

`bias_post_training` (MetricsSource): A metric source object that represents Post-training report (default: None).

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

Usage:

```
ModelMetrics$to_request_list()
```

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

Usage:

```
ModelMetrics$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ModelMonitor` *Sets up Amazon SageMaker Monitoring Schedules and baseline suggestions.*

Description

Use this class when you want to provide your own container image containing the code you'd like to run, in order to produce your own statistics and constraint validation files. For a more guided experience, consider using the DefaultModelMonitor class instead.

Methods

Public methods:

- `ModelMonitor$new()`
- `ModelMonitor$run_baseline()`
- `ModelMonitor$create_monitoring_schedule()`
- `ModelMonitor$update_monitoring_schedule()`
- `ModelMonitor$start_monitoring_schedule()`
- `ModelMonitor$stop_monitoring_schedule()`
- `ModelMonitor$delete_monitoring_schedule()`
- `ModelMonitor$baseline_statistics()`
- `ModelMonitor$suggested_constraints()`
- `ModelMonitor$latest_monitoring_statistics()`
- `ModelMonitor$latest_monitoring_constraint_violations()`
- `ModelMonitor$describe_latest_baselining_job()`
- `ModelMonitor$describe_schedule()`
- `ModelMonitor$list_executions()`
- `ModelMonitor$attach()`
- `ModelMonitor$monitoring_type()`
- `ModelMonitor$format()`
- `ModelMonitor$clone()`

Method `new()`**:** Initializes a “Monitor“ instance. The Monitor handles baselining datasets and creating Amazon SageMaker Monitoring Schedules to monitor SageMaker endpoints.

Usage:

```
ModelMonitor$new(
  role = NULL,
  image_uri = NULL,
  instance_count = 1,
  instance_type = "ml.m5.xlarge",
  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. The Amazon SageMaker jobs use this role.

`image_uri` (str): The uri of the image to use for the jobs started by the Monitor.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.xlarge'.

`entrypoint` ([str]): The entrypoint for the job.

`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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

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

`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.

`env` (dict): Environment variables to be passed to the job.

`tags` ([dict]): List of tags to be passed to the job.

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

Method `run_baseline()`**:** Run a processing job meant to baseline your dataset.

Usage:

```
ModelMonitor$run_baseline(
  baseline_inputs,
  output,
  arguments = NULL,
  wait = TRUE,
  logs = TRUE,
  job_name = NULL
)
```

Arguments:

`baseline_inputs` ([sagemaker.processing.ProcessingInput]): Input files for the processing job. These must be provided as ProcessingInput objects.

`output` (sagemaker.processing.ProcessingOutput): Destination of the constraint_violations and statistics json files.

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

`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 image name and current timestamp.

Method `create_monitoring_schedule()`**:** Creates a monitoring schedule to monitor an Amazon SageMaker Endpoint. If constraints and statistics are provided, or if they are able to be retrieved from a previous baselining job associated with this monitor, those will be used. If constraints and statistics cannot be automatically retrieved, baseline_inputs will be required in order to kick off a baselining job.

Usage:

```
ModelMonitor$create_monitoring_schedule(
  endpoint_input,
  output,
  statistics = NULL,
  constraints = NULL,
  monitor_schedule_name = NULL,
  schedule_cron_expression = NULL
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`output` (sagemaker.model_monitor.MonitoringOutput): The output of the monitoring schedule.

`statistics` (sagemaker.model_monitor.Statistic or str): If provided alongside constraints, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an S3 uri pointing to a constraints JSON file.

`constraints` (sagemaker.model_monitor.Constraints or str): If provided alongside statistics, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an S3 uri pointing to a constraints JSON file.

`monitor_schedule_name` (str): Schedule name. If not specified, the processor generates a default job name, based on the image name and current timestamp.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job runs at. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

Method `update_monitoring_schedule()`**:** Updates the existing monitoring schedule.

Usage:

```
ModelMonitor$update_monitoring_schedule(
  endpoint_input = NULL,
  output = NULL,
  statistics = NULL,
  constraints = NULL,
  schedule_cron_expression = NULL,
  instance_count = NULL,
  instance_type = NULL,
  entrypoint = NULL,
  volume_size_in_gb = NULL,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  arguments = NULL,
  max_runtime_in_seconds = NULL,
  env = NULL,
  network_config = NULL,
  role = NULL,
  image_uri = NULL
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`output` (sagemaker.model_monitor.MonitoringOutput): The output of the monitoring schedule.

`statistics` (sagemaker.model_monitor.Statistic or str): If provided alongside constraints, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an S3 uri pointing to a constraints JSON file.

`constraints` (sagemaker.model_monitor.Constraints or str): If provided alongside statistics, these will be used for monitoring the endpoint. This can be a sagemaker.model_monitor.Constraints object or an S3 uri pointing to a constraints JSON file.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job runs at. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.xlarge'.

`entrypoint` (str): The entrypoint for the job.

`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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

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

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

`env` (dict): Environment variables to be passed to the job.

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

`role` (str): An AWS IAM role name or ARN. The Amazon SageMaker jobs use this role.

`image_uri` (str): The uri of the image to use for the jobs started by the Monitor.

Method `start_monitoring_schedule()`**:** Starts the monitoring schedule.

Usage:

```
ModelMonitor$start_monitoring_schedule()
```

Method `stop_monitoring_schedule()`**:** Stops the monitoring schedule.

Usage:

```
ModelMonitor$stop_monitoring_schedule()
```

Method `delete_monitoring_schedule()`**:** Deletes the monitoring schedule.

Usage:

```
ModelMonitor$delete_monitoring_schedule()
```

Method `baseline_statistics()`**:** Returns a Statistics object representing the statistics json file generated by the latest baselining job.

Usage:

```
ModelMonitor$baseline_statistics(file_name = STATISTICS_JSON_DEFAULT_FILE_NAME)
```

Arguments:

`file_name` (str): The name of the .json statistics file

Returns: sagemaker.model_monitor.Statistics: The Statistics object representing the file that was generated by the job.

Method `suggested_constraints()`**:** Returns a Statistics object representing the constraints json file generated by the latest baselining job

Usage:

```
ModelMonitor$suggested_constraints(
  file_name = CONSTRAINTS_JSON_DEFAULT_FILE_NAME
)
```

Arguments:

`file_name` (str): The name of the .json constraints file

`sagemaker.model_monitor.Constraints`: The Constraints object representing the file that was generated by the job.

Method `latest_monitoring_statistics()`**:** Returns the sagemaker.model_monitor.Statistics generated by the latest monitoring execution.

Usage:

```
ModelMonitor$latest_monitoring_statistics(
  file_name = STATISTICS_JSON_DEFAULT_FILE_NAME
)
```

Arguments:

`file_name` (str): The name of the statistics file to be retrieved. Only override if generating a custom file name.

Returns: sagemaker.model_monitoring.Statistics: The Statistics object representing the file generated by the latest monitoring execution.

Method `latest_monitoring_constraint_violations()`**:** Returns the sagemaker.model_monitor.ConstraintViolations generated by the latest monitoring execution.

Usage:

```
ModelMonitor$latest_monitoring_constraint_violations(
  file_name = CONSTRAINT_VIOLATIONS_JSON_DEFAULT_FILE_NAME
)
```

Arguments:

`file_name` (str): The name of the constraint violdations file to be retrieved. Only override if generating a custom file name.

Returns: sagemaker.model_monitoring.ConstraintViolations: The ConstraintViolations object representing the file generated by the latest monitoring execution.

Method `describe_latest_baselining_job()`**:** Describe the latest baselining job kicked off by the suggest workflow.

Usage:

```
ModelMonitor$describe_latest_baselining_job()
```

Method `describe_schedule()`**:** Describes the schedule that this object represents.

Usage:

```
ModelMonitor$describe_schedule()
```

Returns: dict: A dictionary response with the monitoring schedule description.

Method `list_executions()`**:** Get the list of the latest monitoring executions in descending order of "ScheduledTime". Statistics or violations can be called following this example: Example: »> my_executions = my_monitor.list_executions() »> second_to_last_execution_statistics = my_executions[-1].statistics() »> second_to_last_execution_violations = my_executions[-1].constraint_violations()

Usage:

```
ModelMonitor$list_executions()
```

Returns: [sagemaker.model_monitor.MonitoringExecution]: List of MonitoringExecutions in descending order of "ScheduledTime".

Method `attach()`**:** Sets this object's schedule name to point to the Amazon Sagemaker Monitoring Schedule name provided. This allows subsequent describe_schedule or list_executions calls to point to the given schedule.

Usage:

```
ModelMonitor$attach(monitor_schedule_name, sagemaker_session = NULL)
```

Arguments:

`monitor_schedule_name` (str): The name of the schedule to attach to.

`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 `monitoring_type()`**:** Type of the monitoring job.

Usage:

```
ModelMonitor$monitoring_type()
```

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

Usage:

```
ModelMonitor$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ModelMonitoringFile` *ModelMonitoringFile Class*

Description

Represents a file with a body and an S3 uri.

Methods

Public methods:

- `ModelMonitoringFile$new()`
- `ModelMonitoringFile$save()`
- `ModelMonitoringFile$format()`
- `ModelMonitoringFile$clone()`

Method `new()`**:** Initializes a file with a body and an S3 uri.

Usage:

```
ModelMonitoringFile$new(body_dict, file_s3_uri, kms_key, sagemaker_session)
```

Arguments:

`body_dict` (str): The body of the JSON file.

`file_s3_uri` (str): The uri of the JSON file.

`kms_key` (str): The kms key to be used to decrypt the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Method `save()`**:** Save the current instance's body to s3 using the instance's s3 path. The S3 path can be overridden by providing one. This also overrides the default save location for this object.

Usage:

```
ModelMonitoringFile$save(new_save_location_s3_uri = NULL)
```

Arguments:

`new_save_location_s3_uri` (str): Optional. The S3 path to save the file to. If not provided, the file is saved in place in S3. If provided, the file's S3 path is permanently updated.

Returns: str: The s3 location to which the file was saved.

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

Usage:

```
ModelMonitoringFile$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`ModelPackage` *ModelPackage class*

Description

A SageMaker "Model" that can be deployed to an "Endpoint".

Super classes

`sagemaker.mlcore::ModelBase` -> `sagemaker.mlcore::Model` -> `ModelPackage`

Methods

Public methods:

- `ModelPackage$new()`
- `ModelPackage$enable_network_isolation()`
- `ModelPackage$.create_sagemaker_model()`
- `ModelPackage$clone()`

Method `new()`**:** Initialize a SageMaker ModelPackage.

Usage:

```
ModelPackage$new(
  role,
  model_data = NULL,
  algorithm_arn = NULL,
  model_package_arn = NULL,
  ...
)
```

Arguments:

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.

`model_data` (str): The S3 location of a SageMaker model data ".tar.gz" file. Must be provided if algorithm_arn is provided.

`algorithm_arn` (str): algorithm arn used to train the model, can be just the name if your account owns the algorithm. Must also provide "model_data".

`model_package_arn` (str): An existing SageMaker Model Package arn, can be just the name if your account owns the Model Package. "model_data" is not required.

`...` : Additional kwargs passed to the Model constructor.

Method `enable_network_isolation():` Whether to enable network isolation when creating a model out of this ModelPackage

Usage:

```
ModelPackage$enable_network_isolation()
```

Returns: bool: If network isolation should be enabled or not.

Method `.create_sagemaker_model():` Create a SageMaker Model Entity

Usage:

```
ModelPackage$.create_sagemaker_model(...)
```

Arguments:

`...` : Positional arguments coming from the caller. This class does not require any so they are ignored.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

ModelQualityMonitor — *Amazon SageMaker model monitor to monitor quality metrics for an endpoint.*

Description

Please see the 'initialize' method of its base class for how to instantiate it.

Super class

`sagemaker.mlcore::ModelMonitor` -> `ModelQualityMonitor`

Public fields

`JOB_DEFINITION_BASE_NAME` Model definition base name

Methods

Public methods:

- `ModelQualityMonitor$new()`
- `ModelQualityMonitor$monitoring_type()`
- `ModelQualityMonitor$suggest_baseline()`
- `ModelQualityMonitor$create_monitorying_schedule()`
- `ModelQualityMonitor$update_monitoring_schedule()`
- `ModelQualityMonitor$delete_monitoring_schedule()`
- `ModelQualityMonitor$attach()`
- `ModelQualityMonitor$clone()`

Method `new()`**:** Initializes a monitor instance. The monitor handles baselining datasets and creating Amazon SageMaker Monitoring Schedules to monitor SageMaker endpoints.

Usage:

```
ModelQualityMonitor$new(
  role,
  instance_count = 1,
  instance_type = "ml.m5.xlarge",
  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. The Amazon SageMaker jobs use this role.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

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

`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.

`env` (dict): Environment variables to be passed to the job.

`tags` ([dict]): List of tags to be passed to the job.

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

Method `monitoring_type()`**:** Type of the monitoring job.

Usage:

```
ModelQualityMonitor$monitoring_type()
```

Method `suggest_baseline()`**:** Suggest baselines for use with Amazon SageMaker Model Monitoring Schedules.

Usage:

```
ModelQualityMonitor$suggest_baseline(
  baseline_dataset,
  dataset_format,
  problem_type,
  inference_attribute = NULL,
  probability_attribute = NULL,
  ground_truth_attribute = NULL,
  probability_threshold_attribute = NULL,
  post_analytics_processor_script = NULL,
  output_s3_uri = NULL,
  wait = FALSE,
  logs = FALSE,
  job_name = NULL
)
```

Arguments:

`baseline_dataset` (str): The path to the baseline_dataset file. This can be a local path or an S3 uri.

`dataset_format` (dict): The format of the baseline_dataset.

`problem_type` (str): The type of problem of this model quality monitoring. Valid values are "Regression", "BinaryClassification", "MulticlassClassification".

`inference_attribute` (str): Index or JSONpath to locate predicted label(s).

`probability_attribute` (str or int): Index or JSONpath to locate probabilities.

`ground_truth_attribute` (str): Index or JSONpath to locate actual label(s).

`probability_threshold_attribute` (float): threshold to convert probabilities to binaries Only used for ModelQualityMonitor, ModelBiasMonitor and ModelExplainabilityMonitor

`post_analytics_processor_script` (str): The path to the record post-analytics processor script. This can be a local path or an S3 uri.

`output_s3_uri` (str): Desired S3 destination Destination of the constraint_violations and statistics json files. Default: "s3://<default_session_bucket>/<job_name>/output"

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

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

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

Returns: sagemaker.processing.ProcessingJob: The ProcessingJob object representing the baselining job.

Method `create_monitoryging_schedule()`**:** Creates a monitoring schedule.

Usage:

```
ModelQualityMonitor$create_monitoryging_schedule(
  endpoint_input,
  ground_truth_input,
  problem_type,
  record_preprocessor_script = NULL,
  post_analytics_processor_script = NULL,
  output_s3_uri = NULL,
  constraints = NULL,
  monitor_schedule_name = NULL,
  schedule_cron_expression = NULL,
  enable_cloudwatch_metrics = TRUE
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`ground_truth_input` (str): S3 URI to ground truth dataset.

`problem_type` (str): The type of problem of this model quality monitoring. Valid values are "Regression", "BinaryClassification", "MulticlassClassification".

`record_preprocessor_script` (str): The path to the record preprocessor script. This can be a local path or an S3 uri.

`post_analytics_processor_script` (str): The path to the record post-analytics processor script. This can be a local path or an S3 uri.

`output_s3_uri` (str): S3 destination of the constraint_violations and analysis result. Default: "s3://<default_session_bucket>/<job_name>/output"

`constraints` (sagemaker.model_monitor.Constraints or str): If provided it will be used for monitoring the endpoint. It can be a Constraints object or an S3 uri pointing to a constraints JSON file.

`monitor_schedule_name` (str): Schedule name. If not specified, the processor generates a default job name, based on the image name and current timestamp.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job run. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

Method `update_monitoring_schedule()`**:** Updates the existing monitoring schedule. If more options than schedule_cron_expression are to be updated, a new job definition will be created to hold them. The old job definition will not be deleted.

Usage:

```
ModelQualityMonitor$update_monitoring_schedule(
  endpoint_input = NULL,
  ground_truth_input = NULL,
  problem_type = NULL,
  record_preprocessor_script = NULL,
  post_analytics_processor_script = NULL,
  output_s3_uri = NULL,
  constraints = NULL,
  schedule_cron_expression = NULL,
  enable_cloudwatch_metrics = NULL,
  role = NULL,
  instance_count = NULL,
  instance_type = NULL,
  volume_size_in_gb = NULL,
  volume_kms_key = NULL,
  output_kms_key = NULL,
  max_runtime_in_seconds = NULL,
  env = NULL,
  network_config = NULL
)
```

Arguments:

`endpoint_input` (str or sagemaker.model_monitor.EndpointInput): The endpoint to monitor. This can either be the endpoint name or an EndpointInput.

`ground_truth_input` (str): S3 URI to ground truth dataset.

`problem_type` (str): The type of problem of this model quality monitoring. Valid values are "Regression", "BinaryClassification", "MulticlassClassification".

`record_preprocessor_script` (str): The path to the record preprocessor script. This can be a local path or an S3 uri.

`post_analytics_processor_script` (str): The path to the record post-analytics processor script. This can be a local path or an S3 uri.

`output_s3_uri` (str): S3 destination of the constraint_violations and analysis result. Default: "s3://<default_session_bucket>/<job_name>/output"

`constraints` (sagemaker.model_monitor.Constraints or str): If provided it will be used for monitoring the endpoint. It can be a Constraints object or an S3 uri pointing to a constraints JSON file.

`schedule_cron_expression` (str): The cron expression that dictates the frequency that this job run. See sagemaker.model_monitor.CronExpressionGenerator for valid expressions. Default: Daily.

`enable_cloudwatch_metrics` (bool): Whether to publish cloudwatch metrics as part of the baselining or monitoring jobs.

`role` (str): An AWS IAM role. The Amazon SageMaker jobs use this role.

`instance_count` (int): The number of instances to run the jobs with.

`instance_type` (str): Type of EC2 instance to use for the job, for example, 'ml.m5.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 job's volume.

`output_kms_key` (str): The KMS key id for the job's outputs.

`max_runtime_in_seconds` (int): Timeout in seconds. After this amount of time, Amazon SageMaker terminates the job regardless of its current status. Default: 3600

`env` (dict): Environment variables to be passed to the job.

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

Method `delete_monitoring_schedule()`**:** Deletes the monitoring schedule and its job definition."

Usage:

```
ModelQualityMonitor$delete_monitoring_schedule()
```

Method `attach()`**:** Sets this object's schedule name to the name provided. This allows subsequent describe_schedule or list_executions calls to point to the given schedule.

Usage:

```
ModelQualityMonitor$attach(monitor_schedule_name, sagemaker_session = NULL)
```

Arguments:

`monitor_schedule_name` (str): The name of the schedule to attach to.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`MonitoringExecution` *MonitoringExecution Class*

Description

Provides functionality to retrieve monitoring-specific files output from monitoring executions

Super classes

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

Methods

Public methods:

- `MonitoringExecution$new()`
- `MonitoringExecution$from_processing_arn()`
- `MonitoringExecution$statistics()`
- `MonitoringExecution$constraint_violations()`
- `MonitoringExecution$clone()`

Method `new()`: Initializes a MonitoringExecution job that tracks a monitoring execution kicked off by an Amazon SageMaker Model Monitoring Schedule.

Usage:

```
MonitoringExecution$new(
  sagemaker_session = NULL,
  job_name = NULL,
  inputs = NULL,
  output = NULL,
  output_kms_key = NULL
)
```

Arguments:

`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.

`job_name` (str): The name of the monitoring execution job.

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

`output` (sagemaker.Processing.ProcessingOutput): The output associated with the monitoring execution.

`output_kms_key` (str): The output kms key associated with the job. Defaults to None if not provided.

Method `from_processing_arn()`: Initializes a Baselining job from a processing arn.

Usage:

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

Arguments:

`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.

`processing_job_arn` (str): ARN of the processing job to create a MonitoringExecution out of.

Returns: sagemaker.processing.BaseliningJob: The instance of ProcessingJob created using the current job name.

Method `statistics()`: Returns a sagemaker.model_monitor.Statistics object representing the statistics JSON file generated by this monitoring execution.

Usage:

```
MonitoringExecution$statistics(
  file_name = STATISTICS_JSON_DEFAULT_FILE_NAME,
  kms_key = NULL
)
```

Arguments:

`file_name` (str): The name of the json-formatted statistics file

`kms_key` (str): The kms key to use when retrieving the file.

Returns: sagemaker.model_monitor.Statistics: The Statistics object representing the file that was generated by the execution.

Method `constraint_violations()`**:** Returns a sagemaker.model_monitor.ConstraintViolations object representing the constraint violations JSON file generated by this monitoring execution.

Usage:

```
MonitoringExecution$constraint_violations(
  file_name = CONSTRAINT_VIOLATIONS_JSON_DEFAULT_FILE_NAME,
  kms_key = NULL
)
```

Arguments:

`file_name` (str): The name of the json-formatted constraint violations file.

`kms_key` (str): The kms key to use when retrieving the file.

Returns: sagemaker.model_monitor.ConstraintViolations: The ConstraintViolations object representing the file that was generated by the monitoring execution.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`MultiDataModel` *MultiDataModel Class*

Description

A SageMaker “MultiDataModel“ that can be used to deploy multiple models to the same SageMaker “Endpoint“, and also deploy additional models to an existing SageMaker multi-model “Endpoint“

Super classes

`sagemaker.mlcore::ModelBase` -> `sagemaker.mlcore::Model` -> `MultiDataModel`

Methods

Public methods:

- `MultiDataModel$new()`
- `MultiDataModel$prepare_container_def()`
- `MultiDataModel$deploy()`
- `MultiDataModel$add_model()`
- `MultiDataModel$list_models()`
- `MultiDataModel$clone()`

Method `new()`**:** Initialize a “MultiDataModel“. In addition to these arguments, it supports all arguments supported by “Model“ constructor

Usage:

```
MultiDataModel$new(
  name,
  model_data_prefix,
  model = NULL,
  image_uri = NULL,
  role = NULL,
  sagemaker_session = NULL,
  ...
)
```

Arguments:

`name` (str): The model name.

`model_data_prefix` (str): The S3 prefix where all the models artifacts (.tar.gz) in a Multi-Model endpoint are located

`model` (sagemaker.Model): The Model object that would define the SageMaker model attributes like vpc_config, predictors, etc. If this is present, the attributes from this model are used when deploying the “MultiDataModel“. Parameters 'image_uri', 'role' and 'kwargs' are not permitted when model parameter is set.

`image_uri` (str): A Docker image_uri URI. It can be null if the 'model' parameter is passed to during “MultiDataModel“ initialization (default: None)

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role if it needs to access some AWS resources. It can be null if this is being used to create a Model to pass to a “PipelineModel“ which has its own Role field or if the 'model' parameter is passed to during “MultiDataModel“ initialization (default: None)

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

`...` : Keyword arguments passed to the :class:‘~sagemaker.model.Model‘ initializer. .. tip:: You can find additional parameters for initializing this class at :class:‘~sagemaker.model.Model‘.

Method `prepare_container_def()`**:** Return a container definition set with MultiModel mode, model data and other parameters from the model (if available). Subclasses can override this to provide custom container definitions for deployment to a specific instance type. Called by “deploy()“.

Usage:

```
MultiDataModel$prepare_container_def(
  instance_type = NULL,
  accelerator_type = NULL
)
```

Arguments:

`instance_type` (str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.

`accelerator_type` (str): The Elastic Inference accelerator type to deploy to the instance for loading and making inferences to the model. For example, 'ml.eia1.medium'.

Returns: dict[str, str]: A complete container definition object usable with the CreateModel API

Method `deploy()`**:** Deploy this "Model" to an "Endpoint" and optionally return a "Predictor". Create a SageMaker "Model" and "EndpointConfig", and deploy an "Endpoint" from this "Model". If self.model is not None, then the "Endpoint" will be deployed with parameters in self.model (like vpc_config, enable_network_isolation, etc). If self.model is None, then use the parameters in "MultiDataModel" constructor will be used. If "self.predictor_cls" is not None, this method returns a the result of invoking "self.predictor_cls" on the created endpoint name. The name of the created model is accessible in the "name" field of this "Model" after deploy returns The name of the created endpoint is accessible in the "endpoint_name" field of this "Model" after deploy returns.

Usage:

```
MultiDataModel$deploy(
  initial_instance_count,
  instance_type,
  serializer = NULL,
  deserializer = NULL,
  accelerator_type = NULL,
  endpoint_name = NULL,
  tags = NULL,
  kms_key = NULL,
  wait = TRUE,
  data_capture_config = NULL
)
```

Arguments:

`initial_instance_count` (int): The initial number of instances to run in the "Endpoint" created from this "Model".

`instance_type` (str): The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge', or 'local' for local mode.

`serializer` (:class:`~sagemaker.serializers.BaseSerializer`): A serializer object, used to encode data for an inference endpoint (default: None). If "serializer" is not None, then "serializer" will override the default serializer. The default serializer is set by the "predictor_cls".

`deserializer` (:class:`~sagemaker.deserializers.BaseDeserializer`): A deserializer object, used to decode data from an inference endpoint (default: None). If "deserializer" is not None, then "deserializer" will override the default deserializer. The default deserializer is set by the "predictor_cls".

`accelerator_type` (str): Type of Elastic Inference accelerator to deploy this model for model loading and inference, for example, 'ml.eia1.medium'. If not specified, no Elastic Inference accelerator will be attached to the endpoint. For more information: https://docs.aws.amazon.com/sagemaker/latest/dg

`endpoint_name` (str): The name of the endpoint to create (default: None). If not specified, a unique endpoint name will be created.

`tags` (List[dict[str, str]]): The list of tags to attach to this specific endpoint.

`kms_key` (str): The ARN of the KMS key that is used to encrypt the data on the storage volume attached to the instance hosting the endpoint.

`wait` (bool): Whether the call should wait until the deployment of this model completes (default: True).

`data_capture_config` (sagemaker.model_monitor.DataCaptureConfig): Specifies configuration related to Endpoint data capture for use with Amazon SageMaker Model Monitoring. Default: None.

Returns: callable[string, sagemaker.session.Session] or None: Invocation of “self.predictor_cls“ on the created endpoint name, if “self.predictor_cls“ is not None. Otherwise, return None.

Method `add_model()`**:** Adds a model to the “MultiDataModel“ by uploading or copying the model_data_source artifact to the given S3 path model_data_path relative to model_data_prefix

Usage:

```
MultiDataModel$add_model(model_data_source, model_data_path = NULL)
```

Arguments:

`model_data_source` : Valid local file path or S3 path of the trained model artifact

`model_data_path` : S3 path where the trained model artifact should be uploaded relative to “self.model_data_prefix“ path. (default: None). If None, then the model artifact is uploaded to a path relative to model_data_prefix

Returns: str: S3 uri to uploaded model artifact

Method `list_models()`**:**

Usage:

```
MultiDataModel$list_models()
```

Returns: Generates and returns relative paths to model archives stored at model_data_prefix S3 location.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`NumpyDeserializer` *NumpySerializer Class*

Description

Deserialize a stream of data in the .npy or UTF-8 CSV/JSON format. This serializer class uses python numpy package to deserialize, R objects through the use of the 'reticulate' package.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `NumpyDeserializer`

Public fields

`np` Python Numpy package

`dtype` The dtype of the data

`allow_pickle` Allow loading pickled object arrays

Methods

Public methods:

- `NumpyDeserializer$new()`
- `NumpyDeserializer$deserialize()`
- `NumpyDeserializer$clone()`

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

Usage:

```
NumpyDeserializer$new(
  dtype = NULL,
  accept = "application/x-npy",
  allow_pickle = TRUE
)
```

Arguments:

`dtype` (str): The dtype of the data (default: None).

`accept` (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: "application/x-npy").

`allow_pickle` (bool): Allow loading pickled object arrays (default: True).

Method `deserialize()`**:** Deserialize data from an inference endpoint into a NumPy array.

Usage:

```
NumpyDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` (botocore.response.StreamingBody): Data to be deserialized.

`content_type` (str): The MIME type of the data.

Returns: matrix: The data deserialized into a R matrix/array.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`NumpySerializer` *NumpySerializer Class*

Description

Serialize data of various formats to a numpy npy file format. This serializer class uses python numpy package to serialize, R objects through the use of the ‘reticulate‘ package.

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `NumpySerializer`

Public fields

`dtype` The dtype of the data

`np` Initialized python numpy package

Methods

Public methods:

- `NumpySerializer$new()`
- `NumpySerializer$serialize()`
- `NumpySerializer$clone()`

Method `new()`**:** Initialize a `NumpySerializer` instance.

Usage:

```
NumpySerializer$new(dtype = NULL, content_type = "application/x-npy")
```

Arguments:

`dtype` (str): The ‘dtype‘ of the data. ‘reticulate‘ auto maps to python, please set R class to be serialized.

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "application/x-npy").

Method `serialize()`**:** Serialize data to a buffer using the .npy format.

Usage:

```
NumpySerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized. Can be a NumPy array, list, file, or buffer.

Returns: (raw): A buffer containing data serialized in the .npy format.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`ParameterRange` *ParameterRange Class*

Description

Base class for representing parameter ranges. This is used to define what hyperparameters to tune for an Amazon SageMaker hyperparameter tuning job and to verify hyperparameters for Marketplace Algorithms.

Public fields

`.all_types` All types of child class

`min_value` The minimum value for the range

`max_value` The maximum value for the rang

`scaling_type` The scale used for searching the range during tuning

Methods

Public methods:

- `ParameterRange$new()`
- `ParameterRange$is_valid()`
- `ParameterRange$cast_to_type()`
- `ParameterRange$as_tuning_range()`
- `ParameterRange$format()`
- `ParameterRange$clone()`

Method `new()`**:** Initialize a parameter range.

Usage:

```
ParameterRange$new(
  min_value,
  max_value,
  scaling_type = c("Auto", "Linear", "Logarithmic", "ReverseLogarithmic")
)
```

Arguments:

`min_value` (float or int): The minimum value for the range.

`max_value` (float or int): The maximum value for the range.

`scaling_type` (str): The scale used for searching the range during tuning (default: 'Auto'). Valid values: 'Auto', 'Linear', Logarithmic' and 'ReverseLogarithmic'.

Method `is_valid()`**:** Determine if a value is valid within this ParameterRange.

Usage:

```
ParameterRange$is_valid(value)
```

Arguments:

`value` (float or int): The value to be verified.

Returns: bool: True if valid, False otherwise.

Method `cast_to_type()`**:** cast value to numeric

Usage:

```
ParameterRange$cast_to_type(value)
```

Arguments:

`value` The value to be verified.

Method `as_tuning_range()`**:** Represent the parameter range as a dicionary suitable for a request to create an Amazon SageMaker hyperparameter tuning job.

Usage:

```
ParameterRange$as_tuning_range(name)
```

Arguments:

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

Returns: dict[str, str]: A dictionary that contains the name and values of the hyperparameter.

Method `format()`: format class

Usage:

```
ParameterRange$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`PipelineModel` *A pipeline of SageMaker 'Model' instances.*

Description

This pipeline can be deployed as an 'Endpoint' on SageMaker.

Methods

Public methods:

- `PipelineModel$new()`
- `PipelineModel$pipeline_container_def()`
- `PipelineModel$deploy()`
- `PipelineModel$transformer()`
- `PipelineModel$delete_model()`
- `PipelineModel$format()`
- `PipelineModel$clone()`

Method `new()`: Initialize an SageMaker "Model" which can be used to build an Inference Pipeline comprising of multiple model containers.

Usage:

```
PipelineModel$new(
  models,
  role,
  predictor_cls = NULL,
  name = NULL,
  vpc_config = NULL,
  sagemaker_session = NULL,
  enable_network_isolation = FALSE
)
```

Arguments:

`models` (list[sagemaker.Model]): For using multiple containers to build an inference pipeline, you can pass a list of "sagemaker.Model" objects in the order you want the inference to happen.

`role` (str): An AWS IAM role (either name or full ARN). The Amazon SageMaker training jobs and APIs that create Amazon SageMaker endpoints use this role to access training data and model artifacts. After the endpoint is created, the inference code might use the IAM role, if it needs to access an AWS resource.

`predictor_cls` (callable[string, sagemaker.session.Session]): A function to call to create a predictor (default: None). If not None, “deploy“ will return the result of invoking this function on the created endpoint name.

`name` (str): The model name. If None, a default model name will be selected on each “deploy“.

`vpc_config` (dict[str, list[str]]): The VpcConfig set on the model (default: None) * ’Subnets’ (list[str]): List of subnet ids. * ’SecurityGroupIds’ (list[str]): List of security group ids.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

`enable_network_isolation` (bool): Default False. if True, enables network isolation in the endpoint, isolating the model container. No inbound or outbound network calls can be made to or from the model container.Boolean

Method `pipeline_container_def()`**:** Return a dict created by “sagemaker.pipeline_container_def()“ for deploying this model to a specified instance type. Subclasses can override this to provide custom container definitions for deployment to a specific instance type. Called by “deploy()“.

Usage:

```
PipelineModel$pipeline_container_def(instance_type)
```

Arguments:

`instance_type` (str): The EC2 instance type to deploy this Model to. For example, ’ml.p2.xlarge’.

Returns: list[dict[str, str]]: A list of container definition objects usable with the CreateModel API in the scenario of multiple containers (Inference Pipeline).

Method `deploy()`**:** Deploy the “Model“ to an “Endpoint“. It optionally return a “Predictor“. Create a SageMaker “Model“ and “EndpointConfig“, and deploy an “Endpoint“ from this “Model“. If “self.predictor_cls“ is not None, this method returns a the result of invoking “self.predictor_cls“ on the created endpoint name. The name of the created model is accessible in the “name“ field of this “Model“ after deploy returns The name of the created endpoint is accessible in the “endpoint_name“ field of this “Model“ after deploy returns.

Usage:

```
PipelineModel$deploy(
  initial_instance_count,
  instance_type,
  serializer = NULL,
  deserializer = NULL,
  endpoint_name = NULL,
  tags = NULL,
  wait = TRUE,
  update_endpoint = FALSE,
  data_capture_config = NULL
)
```

Arguments:

`initial_instance_count` (int): The initial number of instances to run in the “Endpoint“ created from this “Model“.

`instance_type` (str): The EC2 instance type to deploy this Model to. For example, ’ml.p2.xlarge’.

`serializer` (:class:‘~sagemaker.serializers.BaseSerializer‘): A serializer object, used to encode data for an inference endpoint (default: None). If “serializer“ is not None, then “serializer“ will override the default serializer. The default serializer is set by the “predictor_cls“.

`deserializer` (:class:‘~sagemaker.deserializers.BaseDeserializer‘): A deserializer object, used to decode data from an inference

`endpoint_name` (str): The name of the endpoint to create (default: None). If not specified, a unique endpoint name will be created.

`tags` (List[dict[str, str]]): The list of tags to attach to this specific endpoint.

`wait` (bool): Whether the call should wait until the deployment of model completes (default: True).

`update_endpoint` (bool): Flag to update the model in an existing Amazon SageMaker endpoint. If True, this will deploy a new EndpointConfig to an already existing endpoint and delete resources corresponding to the previous EndpointConfig. If False, a new endpoint will be created. Default: False

`data_capture_config` (sagemaker.model_monitor.DataCaptureConfig): Specifies configuration related to Endpoint data capture for use with Amazon SageMaker Model Monitoring. Default: None.

`endpoint` (default: None). If “deserializer“ is not None, then “deserializer“ will override the default deserializer. The default deserializer is set by the “predictor_cls“.

Returns: callable[string, sagemaker.session.Session] or None: Invocation of “self.predictor_cls“ on the created endpoint name, if “self.predictor_cls“ is not None. Otherwise, return None.

Method `transformer()`**:** Return a “Transformer“ that uses this Model.

Usage:

```
PipelineModel$transformer(
  instance_count,
  instance_type,
  strategy = NULL,
  assemble_with = NULL,
  output_path = NULL,
  output_kms_key = NULL,
  accept = NULL,
  env = NULL,
  max_concurrent_transforms = NULL,
  max_payload = NULL,
  tags = NULL,
  volume_kms_key = NULL
)
```

Arguments:

`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.

`env` (dict): Environment variables to be set for use during the transform job (default: None).

`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. If none specified, then the tags used for the training job are used for the transform job.

`volume_kms_key` (str): Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

Method `delete_model()`: Delete the SageMaker model backing this pipeline model. This does not delete the list of SageMaker models used in multiple containers to build the inference pipeline.

Usage:

```
PipelineModel$delete_model()
```

Method `format()`: format class

Usage:

```
PipelineModel$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`predict.Predictor` *S3 method that wraps Predictor Class*

Description

Predicted values returned from endpoint

Usage

```
## S3 method for class 'Predictor'
predict(object, newdata, serializer = NULL, deserializer = NULL, ...)
```

Arguments

`object`	a sagemaker model
`newdata`	data for model to predict
`serializer`	method class to serializer data to sagemaker model. Requires to be a class inherited from BaseSerializer.
`deserializer`	method class to deserializer return data streams from sagemaker model. Requires to be a class inherited from BaseDeserializer.
`...`	arguments passed to "Predictor$predict"

`Predictor` *Predictor Class*

Description

Make prediction requests to an Amazon SageMaker endpoint.

Super class

`sagemaker.mlcore::PredictorBase` -> `Predictor`

Public fields

`endpoint_name` Name of the Amazon SageMaker endpoint

`sagemaker_session` A SageMaker Session object

`serializer` Class to convert data into raw to send to endpoint

`deserializer` Class to convert raw data back from the endpoint

Active bindings

`content_type` The MIME type of the data sent to the inference endpoint.

`accept` The content type(s) that are expected from the inference endpoint.

`endpoint` Deprecated attribute. Please use endpoint_name.

Methods

Public methods:

- `Predictor$new()`
- `Predictor$predict()`
- `Predictor$update_endpoint()`
- `Predictor$delete_endpoint()`
- `Predictor$delete_predictor()`
- `Predictor$delete_model()`
- `Predictor$enable_data_capture()`

- Predictor$disable_data_capture()
- Predictor$update_data_capture_config()
- Predictor$list_monitors()
- Predictor$endpoint_context()
- Predictor$format()
- Predictor$clone()

Method `new()`**:** Initialize a “Predictor“. Behavior for serialization of input data and deserialization of result data can be configured through initializer arguments. If not specified, a sequence of bytes is expected and the API sends it in the request body without modifications. In response, the API returns the sequence of bytes from the prediction result without any modifications.

Usage:

```
Predictor$new(
  endpoint_name,
  sagemaker_session = NULL,
  serializer = IdentitySerializer$new(),
  deserializer = BytesDeserializer$new(),
  ...
)
```

Arguments:

`endpoint_name` (str): Name of the Amazon SageMaker endpoint_name to which requests are sent.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: NULL). If not specified, one is created using the default AWS configuration chain.

`serializer` (callable): Accepts a single argument, the input data, and returns a sequence of bytes. It may provide a “content_type“ attribute that defines the endpoint request content type. If not specified, a sequence of bytes is expected for the data. (default: “IdentitySerializer“)

`deserializer` (callable): Accepts two arguments, the result data and the response content type, and returns a sequence of bytes. It may provide a “content_type“ attribute that defines the endpoint response's "Accept" content type. If not specified, a sequence of bytes is expected for the data. (default: “BytesDeserializer“)

`...` Any other parameters, including and deprecate parameters from sagemaker v-1.

Method `predict()`**:** Return the inference from the specified endpoint.

Usage:

```
Predictor$predict(
  data,
  initial_args = NULL,
  target_model = NULL,
  target_variant = NULL,
  inference_id = NULL
)
```

Arguments:

data (object): Input data for which you want the model to provide inference. If a serializer was specified when creating the Predictor, the result of the serializer is sent as input data. Otherwise the data must be sequence of bytes, and the predict method then sends the bytes in the request body as is.

initial_args (list[str,str]): Optional. Default arguments for boto3 “invoke_endpoint“ call. Default is NULL (no default arguments).

target_model (str): S3 model artifact path to run an inference request on, in case of a multi model endpoint. Does not apply to endpoints hosting single model (Default: NULL)

target_variant (str): The name of the production variant to run an inference request on (Default: NULL). Note that the ProductionVariant identifies the model you want to host and the resources you want to deploy for hosting it.

inference_id (str): If you provide a value, it is added to the captured data when you enable data capture on the endpoint (Default: None).

Returns: object: Inference for the given input. If a deserializer was specified when creating the Predictor, the result of the deserializer is returned. Otherwise the response returns the sequence of bytes as is.

Method update_endpoint(): Update the existing endpoint with the provided attributes. This creates a new EndpointConfig in the process. If “initial_instance_count“, “instance_type“, “accelerator_type“, or “model_name“ is specified, then a new ProductionVariant configuration is created; values from the existing configuration are not preserved if any of those parameters are specified.

Usage:

```
Predictor$update_endpoint(
  initial_instance_count = NULL,
  instance_type = NULL,
  accelerator_type = NULL,
  model_name = NULL,
  tags = NULL,
  kms_key = NULL,
  data_capture_config_dict = NULL,
  wait = TRUE
)
```

Arguments:

initial_instance_count (int): The initial number of instances to run in the endpoint. This is required if “instance_type“, “accelerator_type“, or “model_name“ is specified. Otherwise, the values from the existing endpoint configuration’s ProductionVariants are used.

instance_type (str): The EC2 instance type to deploy the endpoint to. This is required if “initial_instance_count“ or “accelerator_type“ is specified. Otherwise, the values from the existing endpoint configuration’s “ProductionVariants“ are used.

accelerator_type (str): The type of Elastic Inference accelerator to attach to the endpoint, e.g. "ml.eia1.medium". If not specified, and “initial_instance_count“, “instance_type“, and “model_name“ are also “None“, the values from the existing endpoint configuration’s “ProductionVariants“ are used. Otherwise, no Elastic Inference accelerator is attached to the endpoint.

model_name (str): The name of the model to be associated with the endpoint. This is required if “initial_instance_count“, “instance_type“, or “accelerator_type“ is specified and if there is more than one model associated with the endpoint. Otherwise, the existing model for the endpoint is used.

`tags` (list[dict[str, str]]): The list of tags to add to the endpoint config. If not specified, the tags of the existing endpoint configuration are used. If any of the existing tags are reserved AWS ones (i.e. begin with "aws"), they are not carried over to the new endpoint configuration.

`kms_key` (str): The KMS key that is used to encrypt the data on the storage volume attached to the instance hosting the endpoint If not specified, the KMS key of the existing endpoint configuration is used.

`data_capture_config_dict` (dict): The endpoint data capture configuration for use with Amazon SageMaker Model Monitoring. If not specified, the data capture configuration of the existing endpoint configuration is used.

`wait` (boolean): Waits for endpoint to update.

Method `delete_endpoint()`: Delete the Amazon SageMaker endpoint backing this predictor. Also delete the endpoint configuration attached to it if delete_endpoint_config is True.

Usage:

```
Predictor$delete_endpoint(delete_endpoint_config = TRUE)
```

Arguments:

`delete_endpoint_config` (bool, optional): Flag to indicate whether to delete endpoint configuration together with endpoint. Defaults to True. If True, both endpoint and endpoint configuration will be deleted. If False, only endpoint will be deleted.

Method `delete_predictor()`: Delete the Amazon SageMaker endpoint backing this predictor. Also delete the endpoint configuration attached to it if delete_endpoint_config is True.

Usage:

```
Predictor$delete_predictor(delete_endpoint_config = TRUE)
```

Arguments:

`delete_endpoint_config` (bool, optional): Flag to indicate whether to delete endpoint configuration together with endpoint. Defaults to True. If True, both endpoint and endpoint configuration will be deleted. If False, only endpoint will be deleted.

Method `delete_model()`: Deletes the Amazon SageMaker models backing this predictor.

Usage:

```
Predictor$delete_model()
```

Method `enable_data_capture()`: Updates the DataCaptureConfig for the Predictor's associated Amazon SageMaker Endpoint to enable data capture. For a more customized experience, refer to update_data_capture_config, instead.

Usage:

```
Predictor$enable_data_capture()
```

Method `disable_data_capture()`: Updates the DataCaptureConfig for the Predictor's associated Amazon SageMaker Endpoint to disable data capture. For a more customized experience, refer to update_data_capture_config, instead.

Usage:

`Predictor$disable_data_capture()`

Method `update_data_capture_config()`**:** Updates the DataCaptureConfig for the Predictor's associated Amazon SageMaker Endpoint with the provided DataCaptureConfig.

Usage:

`Predictor$update_data_capture_config(data_capture_config = NULL)`

Arguments:

`data_capture_config` (sagemaker.model_monitor.DataCaptureConfig): The DataCaptureConfig to update the predictor's endpoint to use.

Method `list_monitors()`**:** Generates ModelMonitor objects (or DefaultModelMonitors). Objects are generated based on the schedule(s) associated with the endpoint that this predictor refers to.

Usage:

`Predictor$list_monitors()`

Returns: [sagemaker.model_monitor.model_monitoring.ModelMonitor]: A list of ModelMonitor (or DefaultModelMonitor) objects.

Method `endpoint_context()`**:** Retrieves the lineage context object representing the endpoint.

Usage:

`Predictor$endpoint_context()`

Returns: ContextEndpoint: The context for the endpoint.

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

Usage:

`Predictor$format()`

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

Usage:

`Predictor$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

`RecordDeserializer` *RecordDeserializer Class*

Description

Deserialize RecordIO Protobuf data from an inference endpoint.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `RecordDeserializer`

Methods

Public methods:

- RecordDeserializer$new()
- RecordDeserializer$deserializer()
- RecordDeserializer$clone()

Method `new()`**:** Intialize RecordDeserializer class

Usage:

```
RecordDeserializer$new(accept = "application/x-recordio-protobuf")
```

Arguments:

`accept` (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: "application/x-recordio-protobuf").

Method `deserializer()`**:** Deserialize RecordIO Protobuf data from an inference endpoint.

Usage:

```
RecordDeserializer$deserializer(data, content_type)
```

Arguments:

`data` (object): The protobuf message to deserialize.

`content_type` (str): The MIME type of the data.

Returns: list: A list of records.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`RecordSerializer` *RecordSerializer Class*

Description

Serialize a matrices and array for an inference request.

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `RecordSerializer`

Methods

Public methods:

- `RecordSerializer$new()`
- `RecordSerializer$serialize()`
- `RecordSerializer$clone()`

Method `new()`**:** Intialize RecordSerializer class

Usage:

```
RecordSerializer$new(content_type = "application/x-recordio-protobuf")
```

Arguments:

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "application/x-recordio-protobuf").

Method `serialize()`**:** Serialize a matrix/array into a buffer containing RecordIO records.

Usage:

```
RecordSerializer$serialize(data)
```

Arguments:

`data` (matrix): The data to serialize.

Returns: raw: A buffer containing the data serialized as records.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`RecordSet` *RecordSet Class*

Description

RecordSet Class

RecordSet Class

Methods

Public methods:

- `RecordSet$new()`
- `RecordSet$data_channel()`
- `RecordSet$records_s3_input()`
- `RecordSet$print()`
- `RecordSet$clone()`

Method `new()`**:** A collection of Amazon :class:~‘Record‘ objects serialized and stored in S3.

Usage:

```
RecordSet$new(
  s3_data,
  num_records,
  feature_dim,
  s3_data_type = "ManifestFile",
  channel = "train"
)
```

Arguments:

`s3_data` (str): The S3 location of the training data

`num_records` (int): The number of records in the set.

`feature_dim` (int): The dimensionality of "values" arrays in the Record features, and label (if each Record is labeled).

`s3_data_type` (str): Valid values: ’S3Prefix’, ’ManifestFile’. If ’S3Prefix’, “s3_data“ defines a prefix of s3 objects to train on. All objects with s3 keys beginning with “s3_data“ will be used to train. If ’ManifestFile’, then “s3_data“ defines a single s3 manifest file, listing each s3 object to train on.

`channel` (str): The SageMaker Training Job channel this RecordSet should be bound to

Method `data_channel()`**:** Return a dictionary to represent the training data in a channel for use with “fit()“

Usage:

```
RecordSet$data_channel()
```

Method `records_s3_input()`**:** Return a TrainingInput to represent the training data

Usage:

```
RecordSet$records_s3_input()
```

Method `print()`**:** Return an unambiguous representation of this RecordSet

Usage:

```
RecordSet$print()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

SimpleBaseDeserializer *Abstract base class for creation of new deserializers.*

Description

This class extends the API of BaseDeserializer with more user-friendly options for setting the ACCEPT content type header, in situations where it can be provided at init and freely updated.

Super class

sagemaker.mlcore::BaseDeserializer -> SimpleBaseDeserializer

Public fields

accept The MIME type that is expected from the inference endpoint

Active bindings

ACCEPT The tuple of possible content types that are expected from the inference endpoint.

Methods

Public methods:

- SimpleBaseDeserializer$new()
- SimpleBaseDeserializer$clone()

Method new()**:** Initialize a “SimpleBaseDeserializer“ instance.

Usage:

```
SimpleBaseDeserializer$new(accept = "*/*")
```

Arguments:

accept (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: "*/*").

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other serializer: BaseDeserializer, BaseSerializer, BytesDeserializer, CSVDeserializer, CSVSerializer, DataTableDeserializer, IdentitySerializer, JSONDeserializer, JSONLinesDeserializer, JSONLinesSerializer, JSONSerializer, LibSVMSerializer, NumpyDeserializer, NumpySerializer, SimpleBaseSerializer, SparseMatrixSerializer, StringDeserializer, TibbleDeserializer

`SimpleBaseSerializer` *Abstract base class for creation of new serializers.*

Description

This class extends the API of `BaseSerializer` with more user-friendly options for setting the Content-Type header, in situations where it can be provided at init and freely updated.

Super class

`sagemaker.mlcore::BaseSerializer` -> `SimpleBaseSerializer`

Public fields

`content_type` The data MIME type

Active bindings

`CONTENT_TYPE` The data MIME type set in the Content-Type header on prediction endpoint requests.

Methods

Public methods:

- `SimpleBaseSerializer$new()`
- `SimpleBaseSerializer$serialize()`
- `SimpleBaseSerializer$clone()`

Method `new()`**:** Initialize a `SimpleBaseSerializer` instance.

Usage:

```
SimpleBaseSerializer$new(content_type = "application/json")
```

Arguments:

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "application/json").

Method `serialize()`**:** Take data of various data formats and serialize them into CSV.

Usage:

```
SimpleBaseSerializer$serialize(data)
```

Arguments:

`data` (object): Data to be serialized

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SparseMatrixSerializer`, `StringDeserializer`, `TibbleDeserializer`

`SparseMatrixSerializer` *SparseMatrixSerializer Class*

Description

Serialize a sparse matrix to a buffer using the .npz format.

Super classes

`sagemaker.mlcore::BaseSerializer` -> `sagemaker.mlcore::SimpleBaseSerializer` -> `SparseMatrixSerializer`

Public fields

`scipy` Python scipy package

Methods

Public methods:

- `SparseMatrixSerializer$new()`
- `SparseMatrixSerializer$serialize()`
- `SparseMatrixSerializer$clone()`

Method `new()`**:** Initialize a “SparseMatrixSerializer“ instance.

Usage:

```
SparseMatrixSerializer$new(content_type = "application/x-npz")
```

Arguments:

`content_type` (str): The MIME type to signal to the inference endpoint when sending request data (default: "application/x-npz").

Method `serialize()`**:** Serialize a sparse matrix to a buffer using the .npz format. Sparse matrices can be in the “csc“, “csr“, “bsr“, “dia“ or “coo“ formats.

Usage:

```
SparseMatrixSerializer$serialize(data)
```

Arguments:

`data` (sparseMatrix): The sparse matrix to serialize.

Returns: raw: A buffer containing the serialized sparse matrix.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `StringDeserializer`, `TibbleDeserializer`

`Statistics` *Statistics Class*

Description

Represents the statistics JSON file used in Amazon SageMaker Model Monitoring.

Super class

`sagemaker.mlcore::ModelMonitoringFile` -> `Statistics`

Methods

Public methods:

- `Statistics$new()`
- `Statistics$from_s3_uri()`
- `Statistics$from_string()`
- `Statistics$from_file_path()`
- `Statistics$clone()`

Method `new()`**:** Initializes the Statistics object used in Amazon SageMaker Model Monitoring.

Usage:

```
Statistics$new(
  body_dict = NULL,
  statistics_file_s3_uri = NULL,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`body_dict` (str): The body of the statistics JSON file.

`statistics_file_s3_uri` (str): The uri of the statistics JSON file.

`kms_key` (str): The kms key to be used to decrypt the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Method `from_s3_uri()`**:** Generates a Statistics object from an s3 uri.

Usage:

```
Statistics$from_s3_uri(
  statistics_file_s3_uri,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`statistics_file_s3_uri` (str): The uri of the statistics JSON file.

`kms_key` (str): The kms key to be used to decrypt the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.Statistics: The instance of Statistics generated from the s3 uri.

Method `from_string()`**:** Generates a Statistics object from an s3 uri.

Usage:

```
Statistics$from_string(
  statistics_file_string,
  kms_key = NULL,
  file_name = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`statistics_file_string` (str): The uri of the statistics JSON file.

`kms_key` (str): The kms key to be used to encrypt the file in S3.

`file_name` (str): The file name to use when uploading to S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.Statistics: The instance of Statistics generated from the s3 uri.

Method `from_file_path()`**:** Initializes a Statistics object from a file path.

Usage:

```
Statistics$from_file_path(
  statistics_file_path,
  kms_key = NULL,
  sagemaker_session = NULL
)
```

Arguments:

`statistics_file_path` (str): The path to the statistics file.

`kms_key` (str): The kms_key to use when encrypting the file in S3.

`sagemaker_session` (sagemaker.session.Session): A SageMaker Session object, used for SageMaker interactions (default: None). If not specified, one is created using the default AWS configuration chain.

Returns: sagemaker.model_monitor.Statistics: The instance of Statistics generated from the local file path.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`StringDeserializer` *StringBaseDeserializer Class*

Description

Deserialize raw data stream into a character string

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `StringBaseDeserializer`

Public fields

`encoding` string encoding to be used

Methods

Public methods:

- `StringDeserializer$new()`
- `StringDeserializer$deserialize()`
- `StringDeserializer$clone()`

Method `new()`**:** Initialize a “StringDeserializer“ instance.

Usage:

```
StringDeserializer$new(encoding = "UTF-8", accept = "application/json")
```

Arguments:

`encoding` (str): The string encoding to use (default: UTF-8).

`accept` (str): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: "application/json").

Method `deserialize()`**:** Takes raw data stream and deserializes it.

Usage:

```
StringDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` raw data to be deserialize

`content_type` (str): The MIME type of the data.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `TibbleDeserializer`

`TibbleDeserializer` *TibbleDeserializer Class*

Description

Deserialize CSV or JSON data from an inference endpoint into a tibble.

Super classes

`sagemaker.mlcore::BaseDeserializer` -> `sagemaker.mlcore::SimpleBaseDeserializer` -> `TibbleDeserializer`

Public fields

`encoding` string encoding to be used

Methods

Public methods:

- `TibbleDeserializer$new()`
- `TibbleDeserializer$deserialize()`
- `TibbleDeserializer$clone()`

Method `new()`**:** Initialize a “TibbleDeserializer“ instance.

Usage:

```
TibbleDeserializer$new(
  encoding = "UTF-8",
  accept = c("text/csv", "application/json")
)
```

Arguments:

`encoding` (str): The string encoding to use (default: "UTF-8").

`accept` (union[str, tuple[str]]): The MIME type (or tuple of allowable MIME types) that is expected from the inference endpoint (default: ("text/csv","application/json")).

Method `deserialize()`**:** Deserialize CSV or JSON data from an inference endpoint into a data.table. If the data is JSON, the data should be formatted in the ’columns’ orient.

Usage:

```
TibbleDeserializer$deserialize(stream, content_type)
```

Arguments:

`stream` (botocore.response.StreamingBody): Data to be deserialized.

`content_type` (str): The MIME type of the data.

Returns: data.table: The data deserialized into a tibble.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other serializer: `BaseDeserializer`, `BaseSerializer`, `BytesDeserializer`, `CSVDeserializer`, `CSVSerializer`, `DataTableDeserializer`, `IdentitySerializer`, `JSONDeserializer`, `JSONLinesDeserializer`, `JSONLinesSerializer`, `JSONSerializer`, `LibSVMSerializer`, `NumpyDeserializer`, `NumpySerializer`, `SimpleBaseDeserializer`, `SimpleBaseSerializer`, `SparseMatrixSerializer`, `StringDeserializer`

`TrainingCompilerConfig` *The configuration class for accelerating SageMaker training jobs through compilation.*

Description

SageMaker Training Compiler speeds up training by optimizing the model execution graph.

Public fields

`DEBUG_PATH` Placeholder

`SUPPORTED_INSTANCE_CLASS_PREFIXES` Placeholder

`HP_ENABLE_COMPILER` Placeholder

`HP_ENABLE_DEBUG` Placeholder

Methods

Public methods:

- `TrainingCompilerConfig$new()`
- `TrainingCompilerConfig$disclaimers_and_warnings()`
- `TrainingCompilerConfig$.to_hyperparameter_list()`
- `TrainingCompilerConfig$validate()`
- `TrainingCompilerConfig$clone()`

Method `new()`**:** This class initializes a “TrainingCompilerConfig“ instance. Pass the output of it to the “compiler_config“ parameter of the :class:‘~sagemaker.huggingface.HuggingFace‘ class.

Usage:

```
TrainingCompilerConfig$new(enabled = TRUE, debug = FALSE)
```

Arguments:

`enabled` (bool): Optional. Switch to enable SageMaker Training Compiler. The default is `True`.

`debug` (bool): Optional. Whether to dump detailed logs for debugging. This comes with a potential performance slowdown. The default is `FALSE`.

Method `disclaimers_and_warnings()`**:** Disclaimers and warnings. Logs disclaimers and warnings about the requested configuration of SageMaker Training Compiler.

Usage:

```
TrainingCompilerConfig$disclaimers_and_warnings()
```

Method `.to_hyperparameter_list()`**:** Converts configuration object into hyperparameters.

Usage:

```
TrainingCompilerConfig$.to_hyperparameter_list()
```

Returns: (list): A portion of the hyperparameters passed to the training job as a list

Method `validate()`**:** Checks if SageMaker Training Compiler is configured correctly.

Usage:

```
TrainingCompilerConfig$validate(image_uri, instance_type, distribution)
```

Arguments:

`image_uri` (str): A string of a Docker image URI that's specified to :class:'~sagemaker.huggingface.HuggingFace'. If SageMaker Training Compiler is enabled, the HuggingFace estimator automatically chooses the right image URI. You cannot specify and override the image URI.

`instance_type` (str): A string of the training instance type that's specified to :class:'~sagemaker.huggingface.HuggingFa The 'validate' classmethod raises error if an instance type not in the "SUPPORTED_INSTANCE_CLASS_PREFIXES list or "local" is passed to the 'instance_type' parameter.

`distribution` (dict): A dictionary of the distributed training option that's specified to :class:'~sagemaker.huggingface.H SageMaker's distributed data parallel and model parallel libraries are currently not compatible with SageMaker Training Compiler.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`WarmStartConfig` *WarmStartConfig Class*

Description

Warm Start Configuration which defines the nature of the warm start "HyperparameterTuner", with type and parents for warm start. Examples: »> warm_start_config = WarmStartConfig(»> type=WarmStartTypes.TransferLearning, parents="p1","p2") »> warm_start_config.type "TransferLearning" »> warm_start_config.parents "p1","p2"

Public fields

`type` Supported warm start types

`parents` Set of parent tuning jobs

Methods

Public methods:

- `WarmStartConfig$new()`
- `WarmStartConfig$from_job_desc()`
- `WarmStartConfig$to_input_req()`
- `WarmStartConfig$format()`

- `WarmStartConfig$clone()`

Method `new()`: Initializes the “WarmStartConfig“ with the provided “WarmStartTypes“ and parents.

Usage:

```
WarmStartConfig$new(
  warm_start_type = c("IdenticalDataAndAlgorithm", "TransferLearning"),
  parents = NULL
)
```

Arguments:

`warm_start_type` (str): This should be one of the supported warm start types “‘("Identical-DataAndAlgorithm", "TransferLearning")“‘

`parents` (str/list): Set of parent tuning jobs which will be used to warm start the new tuning job.

Method `from_job_desc()`: Creates an instance of “WarmStartConfig“ class, from warm start configuration response from DescribeTrainingJob. Examples: »> warm_start_config = WarmStartConfig$new()$from_job_desc(warm_start_config=list(»> "WarmStartType"="TransferLearning", »> "ParentHyperParameterTuningJobs"= list(»> list('HyperParameterTuningJobName'= "p1"), »> list('HyperParameterTuningJobName'= "p2") »>) »>)) »> warm_start_config.type "TransferLearning" »> warm_start_config.parents ["p1","p2"]

Usage:

```
WarmStartConfig$from_job_desc(warm_start_config)
```

Arguments:

`warm_start_config` (dict): The expected format of the “warm_start_config“ contains two first-class

Returns: sagemaker.tuner.WarmStartConfig: De-serialized instance of WarmStartConfig containing the type and parents provided as part of “warm_start_config“.

Method `to_input_req()`: Converts the “self“ instance to the desired input request format. Examples: »> warm_start_config = WarmStartConfig$new("TransferLearning",parents="p1,p2") »> warm_start_config$to_input_req() list("WarmStartType"="TransferLearning", "ParentHyperParameterTuningJobs"= list(list('HyperParameterTuningJobName': "p1"), list('HyperParameterTuningJobName': "p2")))

Usage:

```
WarmStartConfig$to_input_req()
```

Returns: list: Containing the "WarmStartType" and "ParentHyperParameterTuningJobs" as the first class fields.

Method `format()`: format class

Usage:

```
WarmStartConfig$format()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`WarmStartTypes` *WarmStartTypes Class*

Description

Warm Start Configuration type. There can be two types of warm start jobs: * IdenticalDataAndAlgorithm: Type of warm start that allows users to reuse training results from existing tuning jobs that have the same algorithm code and datasets. * TransferLearning: Type of warm start that allows users to reuse training results from existing tuning jobs that have similar algorithm code and datasets.

Usage

```
WarmStartTypes
```

Format

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

Index