sagemaker

Release 1.17.0

Jan 10, 2019

Contents

Amazon SageMaker Python SDK is an open source library for training and deploying machine-learned models on Amazon SageMaker.

With the SDK, you can train and deploy models using popular deep learning frameworks, algorithms provided by Amazon, or your own algorithms built into SageMaker-compatible Docker images.

Here you'll find API docs for SageMaker Python SDK. The project homepage is in Github: https://github.com/aws/sagemaker-python-sdk, where you can find the SDK source, installation instructions and a general overview of the library.

CHAPTER 1

Overview

The SageMaker Python SDK consists of a few primary interfaces:

1.1 Estimators

A high level interface for SageMaker training

class sagemaker.estimator.**EstimatorBase**(*role*, *train_instance_count*, *train_instance_type*, *train_volume_size=30*, *train_volume_kms_key=None*, *train_max_run=86400*, *input_mode='File'*, *output_path=None*, *output_kms_key=None*, *base_job_name=None*, *sagemaker_session=None*, *tags=None*, *subnets=None*, *security_group_ids=None*, *model_uri=None*, *model_channel_name='model'*, *metric_definitions=None*)

Bases: object

Handle end-to-end Amazon SageMaker training and deployment tasks.

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.

Initialize an EstimatorBase instance.

Parameters

- **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.

- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **train_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).
- **train_volume_kms_key** (*str*) – Optional. KMS key ID for encrypting EBS volume attached to the training instance (default: None).
- **train_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 `sagemaker.session.s3_input.input_mode`.
- **output_path** (*str*) – S3 location for saving the trainig 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 *fit()* method execution.
- **output_kms_key** (*str*) – Optional. KMS key ID for encrypting the training output (default: None).
- **base_job_name** (*str*) – Prefix for training job name when the *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: None). 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.

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

hyperparameters()
Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

enable_network_isolation()
Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*inputs=None*, *wait=True*, *logs=True*, *job_name=None*)
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.

Parameters

- **inputs** (*str or dict or sagemaker.session.s3_input*) – Information about the training data. This can be one of three types:
  - (str) the S3 location where training data is saved.
  - **(dict[str, str] or dict[str, sagemaker.session.s3_input]) If using multiple channels for** training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.
  - **(sagemaker.session.s3_input) - channel configuration for S3 data sources that can provide** additional information as well as the path to the training dataset. See *sagemaker.session.s3_input()* for full details.
- **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.

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)
Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

model_data

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

Type str

create_model(***kwargs*)

Create a SageMaker `Model` object that can be deployed to an `Endpoint`.

Parameters ****kwargs** – Keyword arguments used by the implemented method for creating the `Model`.

Returns A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

delete_endpoint()
Delete an Amazon SageMaker Endpoint.

Raises ValueError – If the endpoint does not exist.

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

training_job_analytics
Return a TrainingJobAnalytics object for the current training job.

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

class sagemaker.estimator.**Estimator**(*image_name*, *role*, *train_instance_count*, *train_instance_type*, *train_volume_size=30*, *train_volume_kms_key=None*, *train_max_run=86400*, *input_mode='File'*, *output_path=None*, *output_kms_key=None*, *base_job_name=None*, *sagemaker_session=None*, *hyperparameters=None*, *tags=None*, *subnets=None*, *security_group_ids=None*, *model_uri=None*, *model_channel_name='model'*, *metric_definitions=None*)

Bases: *sagemaker.estimator.EstimatorBase*

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.

Initialize an `Estimator` instance.

Parameters

- **image_name** (*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.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **train_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).
- **train_volume_kms_key** (*str*) – Optional. KMS key ID for encrypting EBS volume attached to the training instance (default: None).
- **train_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 `sagemaker.session.s3_input.input_mode`.
- **output_path** (*str*) – S3 location for saving the trainig 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 *fit()* method execution.
- **output_kms_key** (*str*) – Optional. KMS key ID for encrypting the training output (default: None).

- **base_job_name** (*str*) – Prefix for training job name when the *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/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: None). 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.

train_image()
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.

set_hyperparameters(***kwargs*)

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.

create_model(*role=None*, *image=None*, *predictor_cls=None*, *serializer=None*, *deserializer=None*, *content_type=None*, *accept=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*, ***kwargs*)
Create a model to deploy.

Parameters

- **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** (*str*) – An container image to use for deploying the model. Defaults to the image used for training.

- **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 th endpoint response Accept content type.
- **content_type** (*str*) – The invocation ContentType, overriding any content_type from the serializer
- **accept** (*str*) – The invocation Accept, overriding any accept from the deserializer.
- **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.
- **serializer, deserializer, content_type, and accept arguments are only used to define a default** (*The*) –
- **They are ignored if an explicit predictor class is passed in. Other arguments** (*RealTimePredictor.*) –
- **passed through to the Model class.** (*are*) –

Returns: a Model ready for deployment.

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()

Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*inputs=None*, *wait=True*, *logs=True*, *job_name=None*)

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.

Parameters

- **inputs** (*str or dict or sagemaker.session.s3_input*) – Information about the training data. This can be one of three types:
  - (str) the S3 location where training data is saved.
  - **(dict[str, str] or dict[str, sagemaker.session.s3_input]) If using multiple channels for** training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.
  - **(sagemaker.session.s3_input) - channel configuration for S3 data sources that can provide** additional information as well as the path to the training dataset. See *sagemaker.session.s3_input()* for full details.
- **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.

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

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

Type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

class sagemaker.estimator.**Framework**(*entry_point*, *source_dir=None*, *hyperparameters=None*, *enable_cloudwatch_metrics=False*, *container_log_level=20*, *code_location=None*, *image_name=None*, *dependencies=None*, ***kwargs*)
Bases: *sagemaker.estimator.EstimatorBase*

Base class that cannot be instantiated directly.

Subclasses define functionality pertaining to specific ML frameworks, such as training/deployment images and predictor instances.

Base class initializer. Subclasses which override `__init__` should invoke `super()`

Parameters

- **entry_point** (*str*) – Path (absolute or relative) to the local Python source file which should be executed as the entry point to training. This should be compatible with either Python 2.7 or Python 3.5.
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from tne entry point file (default: None). Structure within this directory are preserved when training on Amazon SageMaker.
- **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. .. rubric:: 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
```

- **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.
- **enable_cloudwatch_metrics** (*bool*) – [DEPRECATED] Now there are cloudwatch metrics emitted by all SageMaker training jobs. This will be ignored for now and removed in a further release.
- **container_log_level** (*int*) – 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 will be uploaded (default: None). The code file uploaded in S3 is 'code_location/source/sourcedir.tar.gz'. If not specified, the default code location is s3://default_bucket/job-name/. And code file uploaded to S3 is s3://default_bucket/job-name/source/sourcedir.tar.gz
- **image_name** (*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.
- ****kwargs** – Additional kwargs passed to the `EstimatorBase` constructor.

LAUNCH_PS_ENV_NAME = 'sagemaker_parameter_server_enabled'

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)
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.

Parameters

- **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.

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()
```

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *model_server_workers=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

create_model(***kwargs*)

Create a SageMaker `Model` object that can be deployed to an `Endpoint`.

Parameters ****kwargs** – Keyword arguments used by the implemented method for creating the `Model`.

Returns A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()

Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*inputs=None*, *wait=True*, *logs=True*, *job_name=None*)
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.

Parameters

- **inputs** (*str or dict or sagemaker.session.s3_input*) – Information about the training data. This can be one of three types:
  - (str) the S3 location where training data is saved.
  - **(dict[str, str] or dict[str, sagemaker.session.s3_input]) If using multiple channels for** training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.
  - **(sagemaker.session.s3_input) - channel configuration for S3 data sources that can provide** additional information as well as the path to the training dataset. See *sagemaker.session.s3_input()* for full details.
- **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.

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

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

Type str

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

1.2 HyperparameterTuner

class sagemaker.tuner.**HyperparameterTuner**(*estimator*, *objective_metric_name*, *hyperparameter_ranges*, *metric_definitions=None*, *strategy='Bayesian'*, *objective_type='Maximize'*, *max_jobs=1*, *max_parallel_jobs=1*, *tags=None*, *base_tuning_job_name=None*, *warm_start_config=None*, *early_stopping_type='Off'*)

Bases: `object`

A class for creating and interacting with Amazon SageMaker hyperparameter tuning jobs, as well as deploying the resulting model(s).

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.

Parameters

- **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/sagemaker/latest/dg/API_Tag.html.
- **base_tuning_job_name** (*str*) – Prefix for the hyperparameter tuning job name when the *fit()* 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.

TUNING_JOB_NAME_MAX_LENGTH = 32

SAGEMAKER_ESTIMATOR_MODULE = 'sagemaker_estimator_module'

SAGEMAKER_ESTIMATOR_CLASS_NAME = 'sagemaker_estimator_class_name'

DEFAULT_ESTIMATOR_MODULE = 'sagemaker.estimator'

DEFAULT_ESTIMATOR_CLS_NAME = 'Estimator'

fit(*inputs=None*, *job_name=None*, *include_cls_metadata=False*, ***kwargs*)
Start a hyperparameter tuning job.

Parameters

- **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, sagemaker.session.s3_input]) - If using multiple channels for** training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.
  - **(sagemaker.session.s3_input) - Channel configuration for S3 data sources that can provide** additional information about the training dataset. See *sagemaker.session.s3_input()* for full details.
  - **(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.
  - **(list[sagemaker.amazon.amazon_estimator.RecordSet]) - A list of** :class:~'sagemaker.amazon.amazon_estimator.RecordSet' 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** (*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`.
- ****kwargs** – Other arguments needed for training. Please refer to the `fit()` method of the associated estimator to see what other arguments are needed.

classmethod attach(*tuning_job_name*, *sagemaker_session=None*, *job_details=None*, *estimator_cls=None*)
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 `deploy()``ed to create an Amazon SageMaker Endpoint and return a ``Predictor`.

Parameters

- **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** (*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:~'sagemaker.estimator.Estimator' if it is unable to determine a more specific class.

Examples

```
>>> my_tuner.fit()
>>> job_name = my_tuner.latest_tuning_job.name
Later on:
>>> attached_tuner = HyperparameterTuner.attach(job_name)
>>> attached_tuner.deploy()
```

Returns

A `HyperparameterTuner` instance with the attached hyperparameter tuning job.

Return type *sagemaker.tuner.HyperparameterTuner*

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, ***kwargs*)

Deploy the best trained or user specified model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

For more information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- ****kwargs** – Other arguments needed for deployment. Please refer to the `create_model()` method of the associated estimator to see what other arguments are needed.

Returns

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

Return type *sagemaker.predictor.RealTimePredictor*

stop_tuning_job()

Stop latest running hyperparameter tuning job.

wait()

Wait for latest hyperparameter tuning job to finish.

best_training_job()
Return name of the best training job for the latest hyperparameter tuning job.

Raises Exception – If there is no best training job available for the hyperparameter tuning job.

delete_endpoint(*endpoint_name=None*)
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.

Parameters **endpoint_name** (*str*) – Name of the endpoint to delete

hyperparameter_ranges()
Return the hyperparameter ranges in a dictionary to be used as part of a request for creating a hyperparameter tuning job.

sagemaker_session
~'sagemaker.session.Session' object associated with the estimator for the HyperparameterTuner.

Type Convenience method for accessing the

Type class

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.

transfer_learning_tuner(*additional_parents=None*, *estimator=None*)
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.

Parameters

- **additional_parents** (*set{str}*) – Set of additional parents along with the self to be used in warm starting
- **transfer learning tuner.** (*the*) –
- **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 HyperparameterTuner instance which can be used to launch transfer learning tuning job.

Return type *sagemaker.tuner.HyperparameterTuner*

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={})
```

identical_dataset_and_algorithm_tuner(*additional_parents=None*)

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`

Parameters

- **additional_parents** (*set{str}*) – Set of additional parents along with the self to be used in warm starting
- **identical dataset and algorithm tuner.** (*the*) –

Returns HyperparameterTuner instance which can be used to launch identical dataset and algorithm tuning job.

Return type *sagemaker.tuner.HyperparameterTuner*

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={})
```

class sagemaker.tuner.**ContinuousParameter**(*min_value*, *max_value*)

Bases: `sagemaker.parameter.ParameterRange`

A class for representing hyperparameters that have a continuous range of possible values. :param min_value: The minimum value for the range. :type min_value: float :param max_value: The maximum value for the range. :type max_value: float

Initialize a parameter range.

Parameters

- **min_value** (*float or int*) – The minimum value for the range.
- **max_value** (*float or int*) – The maximum value for the range.

classmethod cast_to_type(*value*)

class sagemaker.tuner.**IntegerParameter**(*min_value*, *max_value*)

Bases: `sagemaker.parameter.ParameterRange`

A class for representing hyperparameters that have an integer range of possible values. :param min_value: The minimum value for the range. :type min_value: int :param max_value: The maximum value for the range. :type max_value: int

Initialize a parameter range.

Parameters

- **min_value** (*float or int*) – The minimum value for the range.
- **max_value** (*float or int*) – The maximum value for the range.

classmethod cast_to_type(*value*)

class sagemaker.tuner.**CategoricalParameter**(*values*)
Bases: sagemaker.parameter.ParameterRange

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

Initialize a CategoricalParameter.

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

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

Parameters **name** (*str*) – The name of the hyperparameter.

Returns A dictionary that contains the name and values of the hyperparameter.

Return type dict[str, list[str]]

as_json_range(*name*)
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.

Parameters **name** (*str*) – The name of the hyperparameter.

Returns

A dictionary that contains the name and values of the hyperparameter, where the values are serialized as JSON.

Return type dict[str, list[str]]

is_valid(*value*)
Determine if a value is valid within this ParameterRange.

Parameters **value** (*float or int*) – The value to be verified.

Returns True if valid, False otherwise.

Return type bool

classmethod cast_to_type(*value*)

1.3 Model

class sagemaker.model.**Model**(*model_data*, *image*, *role=None*, *predictor_cls=None*, *env=None*, *name=None*, *vpc_config=None*, *sagemaker_session=None*)
Bases: object

A SageMaker Model that can be deployed to an Endpoint.

Initialize an SageMaker Model.

Parameters

- **model_data** (*str*) – The S3 location of a SageMaker model data .tar.gz file.
- **image** (*str*) – A Docker image URI.

- **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: None)
- **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`.
- **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.

prepare_container_def(*instance_type*, *accelerator_type=None*)

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()`.

Parameters

- **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 A container definition object usable with the CreateModel API.

Return type dict

enable_network_isolation()

Whether to enable network isolation when creating this Model

Returns If network isolation should be enabled or not.

Return type bool

compile(*target_instance_family*, *input_shape*, *output_path*, *role*, *tags=None*, *job_name=None*, *compile_max_run=300*, *framework=None*, *framework_version=None*)

Compile this `Model` with SageMaker Neo.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b

- **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
- **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/dg/API_Tag.html.
- **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', 'pytorch', 'onnx', 'xgboost'
- **framework_version** (*str*) –

Returns A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *tags=None*)

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.

Parameters

- **instance_type** (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.
- **initial_instance_count** (*int*) – The initial number of instances to run in the `Endpoint` created from this `Model`.
- **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/ei.html
- **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.

Returns

Invocation of `self.predictor_cls` on the created endpoint name, if `self.predictor_cls` is not None. Otherwise, return None.

Return type callable[string, *sagemaker.session.Session*] or None

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *volume_kms_key=None*)

Return a `Transformer` that uses this Model.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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 Model 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).

1.4 PipelineModel

class sagemaker.pipeline.**PipelineModel**(*models*, *role*, *predictor_cls=None*, *name=None*, *vpc_config=None*, *sagemaker_session=None*)

Bases: `object`

A pipeline of SageMaker `Model``s that can be deployed to an ``Endpoint`.

Initialize an SageMaker `Model` which can be used to build an Inference Pipeline comprising of multiple model containers.

Parameters

- **models** (*list[sagemaker.Model]*) – For using multiple containers to build an inference pipeline,
- **can pass a list of sagemaker.Model objects in the order you want the inference to happen.** (*you*) –
- **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.

pipeline_container_def(*instance_type*)

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()`.

Parameters **instance_type** (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.

Returns A list of container definition objects usable with the CreateModel API in the scenario of multiple containers (Inference Pipeline).

Return type list[dict[str, str]]

deploy(*initial_instance_count*, *instance_type*, *endpoint_name=None*, *tags=None*)

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.

Parameters

- **instance_type** (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.
- **initial_instance_count** (*int*) – The initial number of instances to run in the `Endpoint` created from this `Model`.
- **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.

Returns

Invocation of self.predictor_cls on the created endpoint name, if `self.predictor_cls` is not None. Otherwise, return None.

Return type callable[string, *sagemaker.session.Session*] or None

1.5 Predictors

Make real-time predictions against SageMaker endpoints with Python objects

class sagemaker.predictor.**RealTimePredictor**(*endpoint*, *sagemaker_session=None*, *serializer=None*, *deserializer=None*, *content_type=None*, *accept=None*)

Bases: `object`

Make prediction requests to an Amazon SageMaker endpoint.

Initialize a `RealTimePredictor`.

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.

Parameters

- **endpoint** (*str*) – Name of the Amazon SageMaker endpoint to which requests are sent.
- **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.
- **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.
- **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.
- **content_type** (*str*) – The invocation's "ContentType", overriding any `content_type` from the serializer (default: None).
- **accept** (*str*) – The invocation's "Accept", overriding any accept from the deserializer (default: None).

predict(*data*, *initial_args=None*)

Return the inference from the specified endpoint.

Parameters

- **data** (*object*) – Input data for which you want the model to provide inference. If a serializer was specified when creating the RealTimePredictor, 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** (*dict[str,str]*) – Optional. Default arguments for boto3 `invoke_endpoint` call. Default is None (no default arguments).

Returns

Inference for the given input. If a deserializer was specified when creating the RealTimePredictor, the result of the deserializer is returned. Otherwise the response returns the sequence of bytes as is.

Return type object

delete_endpoint()
Delete the Amazon SageMaker endpoint backing this predictor.

1.6 Transformer

class sagemaker.transformer.**Transformer**(*model_name*, *instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *env=None*, *base_transform_job_name=None*, *sagemaker_session=None*, *volume_kms_key=None*)

Bases: object

A class for handling creating and interacting with Amazon SageMaker transform jobs.

Initialize a `Transformer`.

Parameters

- **model_name** (*str*) – Name of the SageMaker model being used for the transform job.
- **instance_count** (*int*) – Number of EC2 instances to use.
- **instance_type** (*str*) – Type of EC2 instance to use, for example, 'ml.c4.xlarge'.
- **strategy** (*str*) – The strategy used to decide how to batch records in a single request (default: None). Valid values: 'MultiRecord' and 'SingleRecord'.
- **assemble_with** (*str*) – How the output is assembled (default: None). Valid values: 'Line' or 'None'.
- **output_path** (*str*) – S3 location for saving the transform result. If not specified, results are stored to a default bucket.
- **output_kms_key** (*str*) – Optional. KMS key ID for encrypting the transform output (default: None).
- **accept** (*str*) – The content type accepted by the endpoint deployed during the transform job.
- **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.
- **env** (*dict*) – Environment variables to be set for use during the transform job (default: None).

- **tags** (*list[dict]*) – List of tags for labeling a transform job (default: None). For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.
- **base_transform_job_name** (*str*) – Prefix for the transform job when the *transform()* method launches. If not specified, a default prefix will be generated based on the training image name that was used to train the model associated with the transform job.
- **sagemaker_session** (sagemaker.session.Session) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, the estimator creates one using the default AWS configuration chain.
- **volume_kms_key** (*str*) – Optional. KMS key ID for encrypting the volume attached to the ML compute instance (default: None).

transform(*data*, *data_type='S3Prefix'*, *content_type=None*, *compression_type=None*, *split_type=None*, *job_name=None*)

Start a new transform job.

Parameters

- **data** (*str*) – Input data location in S3.
- **data_type** (*str*) – What the S3 location defines (default: 'S3Prefix'). Valid values:
  - **'S3Prefix' - the S3 URI defines a key name prefix. All objects with this prefix will be used as** inputs for the transform job.
  - **'ManifestFile' - the S3 URI points to a single manifest file listing each S3 object to use as** an input for the transform job.
- **content_type** (*str*) – MIME type of the input data (default: None).
- **compression_type** (*str*) – Compression type of the input data, if compressed (default: None). Valid values: 'Gzip', None.
- **split_type** (*str*) – The record delimiter for the input object (default: 'None'). Valid values: 'None', 'Line', 'RecordIO', and 'TFRecord'.
- **job_name** (*str*) – job name (default: None). If not specified, one will be generated.

wait()

classmethod attach(*transform_job_name*, *sagemaker_session=None*)

Attach an existing transform job to a new Transformer instance

Parameters

- **transform_job_name** (*str*) – Name for the transform job to be attached.
- **sagemaker_session** (sagemaker.session.Session) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, one will be created using the default AWS configuration chain.

Returns The Transformer instance with the specified transform job attached.

Return type *sagemaker.transformer.Transformer*

1.7 Session

class sagemaker.session.**LogState**
Bases: object

STARTING = 1

WAIT_IN_PROGRESS = 2

TAILING = 3

JOB_COMPLETE = 4

COMPLETE = 5

class sagemaker.session.**Session**(*boto_session=None*, *sagemaker_client=None*, *sagemaker_runtime_client=None*)
Bases: object

Manage interactions with the Amazon SageMaker APIs and any other AWS services needed.

This class provides convenient methods for manipulating entities and resources that Amazon SageMaker uses, such as training jobs, endpoints, and input datasets in S3.

AWS service calls are delegated to an underlying Boto3 session, which by default is initialized using the AWS configuration chain. When you make an Amazon SageMaker API call that accesses an S3 bucket location and one is not specified, the `Session` creates a default bucket based on a naming convention which includes the current AWS account ID.

Initialize a SageMaker `Session`.

Parameters

- **boto_session** (*boto3.session.Session*) – The underlying Boto3 session which AWS service calls are delegated to (default: None). If not provided, one is created with default AWS configuration chain.
- **sagemaker_client** (*boto3.SageMaker.Client*) – Client which makes Amazon SageMaker service calls other than `InvokeEndpoint` (default: None). Estimators created using this `Session` use this client. If not provided, one will be created using this instance's `boto_session`.
- **sagemaker_runtime_client** (*boto3.SageMakerRuntime.Client*) – Client which makes `InvokeEndpoint` calls to Amazon SageMaker (default: None). Predictors created using this `Session` use this client. If not provided, one will be created using this instance's `boto_session`.

boto_region_name

upload_data(*path*, *bucket=None*, *key_prefix='data'*)
Upload local file or directory to S3.

If a single file is specified for upload, the resulting S3 object key is `{key_prefix}/{filename}` (filename does not include the local path, if any specified).

If a directory is specified for upload, the API uploads all content, recursively, preserving relative structure of subdirectories. The resulting object key names are: `{key_prefix}/{relative_subdirectory_path}/filename`.

Parameters

- **path** (*str*) – Path (absolute or relative) of local file or directory to upload.

- **bucket** (*str*) – Name of the S3 Bucket to upload to (default: None). If not specified, the default bucket of the `Session` is used (if default bucket does not exist, the `Session` creates it).
- **key_prefix** (*str*) – Optional S3 object key name prefix (default: 'data'). S3 uses the prefix to create a directory structure for the bucket content that it display in the S3 console.

Returns

The S3 URI of the uploaded file(s). If a file is specified in the path argument, the URI format is: `s3://{bucket name}/{key_prefix}/{original_file_name}`. If a directory is specified in the path argument, the URI format is `s3://{bucket name}/{key_prefix}`.

Return type str

default_bucket()

Return the name of the default bucket to use in relevant Amazon SageMaker interactions.

Returns The name of the default bucket, which is of the form: `sagemaker-{region}-{AWS account ID}`.

Return type str

train(*input_mode*, *input_config*, *role*, *job_name*, *output_config*, *resource_config*, *vpc_config*, *hyperparameters*, *stop_condition*, *tags*, *metric_definitions*, *enable_network_isolation=False*, *image=None*, *algorithm_arn=None*)

Create an Amazon SageMaker training job.

Parameters

- **input_mode** (*str*) – The input mode that the algorithm supports. Valid modes:
  - **'File' - Amazon SageMaker copies the training dataset from the S3 location to** a directory in the Docker container.
  - 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a Unix-named pipe.
- **input_config** (*list*) – A list of Channel objects. Each channel is a named input source. Please refer to the format details described: https://botocore.readthedocs.io/en/latest/reference/services/sagemaker.html#SageMaker.Client.create_training_job
- **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. You must grant sufficient permissions to this role.
- **job_name** (*str*) – Name of the training job being created.
- **output_config** (*dict*) – The S3 URI where you want to store the training results and optional KMS key ID.
- **resource_config** (*dict*) – Contains values for ResourceConfig:
  - **instance_count (int): Number of EC2 instances to use for training.** The key in resource_config is 'InstanceCount'.
  - **instance_type (str): Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.** The key in resource_config is 'InstanceType'.
- **vpc_config** (*dict*) – Contains values for VpcConfig:
  - **subnets (list[str]): List of subnet ids.** The key in vpc_config is 'Subnets'.

– **security_group_ids (list[str]): List of security group ids.** The key in vpc_config is 'SecurityGroupIds'.

- **hyperparameters** (*dict*) – Hyperparameters for model training. 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.
- **stop_condition** (*dict*) – Defines when training shall finish. Contains entries that can be understood by the service like `MaxRuntimeInSeconds`.
- **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.
- **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.
- **enable_network_isolation** (*bool*) – Whether to request for the training job to run with network isolation or not.
- **image** (*str*) – Docker image containing training code.
- **algorithm_arn** (*str*) – Algorithm Arn from Marketplace.

Returns ARN of the training job, if it is created.

Return type str

compile_model(*input_model_config*, *output_model_config*, *role*, *job_name*, *stop_condition*, *tags*)
Create an Amazon SageMaker Neo compilation job.

Parameters

- **input_model_config** (*dict*) – the trained model and the Amazon S3 location where it is stored.
- **output_model_config** (*dict*) –

  – Identifies the Amazon S3 location where you want Amazon SageMaker Neo to save the results of compilation job
- **role** (*str*) – An AWS IAM role (either name or full ARN). The Amazon SageMaker Neo compilation jobs use this role to access model artifacts. You must grant sufficient permissions to this role.
- **job_name** (*str*) – Name of the compilation job being created.
- **stop_condition** (*dict*) – Defines when compilation job shall finish. Contains entries that can be understood by the service like `MaxRuntimeInSeconds`.
- **tags** (*list[dict]*) – List of tags for labeling a compile model job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

Returns ARN of the compile model job, if it is created.

Return type str

tune(*job_name*, *strategy*, *objective_type*, *objective_metric_name*, *max_jobs*, *max_parallel_jobs*, *parameter_ranges*, *static_hyperparameters*, *input_mode*, *metric_definitions*, *role*, *input_config*, *output_config*, *resource_config*, *stop_condition*, *tags*, *warm_start_config*, *enable_network_isolation=False*, *image=None*, *algorithm_arn=None*, *early_stopping_type='Off'*)
Create an Amazon SageMaker hyperparameter tuning job

Parameters

- **job_name** (*str*) – Name of the tuning job being created.
- **strategy** (*str*) – Strategy to be used for hyperparameter estimations.
- **objective_type** (*str*) – The type of the objective metric for evaluating training jobs. This value can be either 'Minimize' or 'Maximize'.
- **objective_metric_name** (*str*) – Name of the metric for evaluating training jobs.
- **max_jobs** (*int*) – Maximum total number of training jobs to start for the hyperparameter tuning job.
- **max_parallel_jobs** (*int*) – Maximum number of parallel training jobs to start.
- **parameter_ranges** (*dict*) – Dictionary of parameter ranges. These parameter ranges can be one of three types: Continuous, Integer, or Categorical.
- **static_hyperparameters** (*dict*) – Hyperparameters for model training. These hyperparameters remain unchanged across all of the training jobs for the hyperparameter tuning job. The hyperparameters are made accessible as a dictionary for the training code on SageMaker.
- **image** (*str*) – Docker image containing training code.
- **input_mode** (*str*) – The input mode that the algorithm supports. Valid modes:
  - **'File' - Amazon SageMaker copies the training dataset from the S3 location to** a directory in the Docker container.
  - 'Pipe' - Amazon SageMaker streams data directly from S3 to the container via a Unix-named pipe.
- **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.
- **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. You must grant sufficient permissions to this role.
- **input_config** (*list*) – A list of Channel objects. Each channel is a named input source. Please refer to the format details described: https://botocore.readthedocs.io/en/latest/reference/services/sagemaker.html#SageMaker.Client.create_training_job
- **output_config** (*dict*) – The S3 URI where you want to store the training results and optional KMS key ID.
- **resource_config** (*dict*) – Contains values for ResourceConfig:
  - **instance_count (int): Number of EC2 instances to use for training.** The key in resource_config is 'InstanceCount'.
  - **instance_type (str): Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.** The key in resource_config is 'InstanceType'.
- **stop_condition** (*dict*) – When training should finish, e.g. `MaxRuntimeInSeconds`.

- **tags** (*list[dict]*) – List of tags for labeling the tuning job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.
- **warm_start_config** (*dict*) – Configuration defining the type of warm start and other required configurations.
- **early_stopping_type** (*str*) – Specifies whether early stopping is enabled for the job. Can be either 'Auto' or '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.

stop_tuning_job(*name*)

Stop the Amazon SageMaker hyperparameter tuning job with the specified name.

Parameters **name** (*str*) – Name of the Amazon SageMaker hyperparameter tuning job.

Raises `ClientError` – If an error occurs while trying to stop the hyperparameter tuning job.

transform(*job_name*, *model_name*, *strategy*, *max_concurrent_transforms*, *max_payload*, *env*, *input_config*, *output_config*, *resource_config*, *tags*)

Create an Amazon SageMaker transform job.

Parameters

- **job_name** (*str*) – Name of the transform job being created.
- **model_name** (*str*) – Name of the SageMaker model being used for the transform job.
- **strategy** (*str*) – The strategy used to decide how to batch records in a single request. Possible values are 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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.
- **env** (*dict*) – Environment variables to be set for use during the transform job.
- **input_config** (*dict*) – A dictionary describing the input data (and its location) for the job.
- **output_config** (*dict*) – A dictionary describing the output location for the job.
- **resource_config** (*dict*) – A dictionary describing the resources to complete the job.
- **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.

create_model(*name*, *role*, *container_defs*, *vpc_config=None*, *enable_network_isolation=False*, *primary_container=None*)

Create an Amazon SageMaker `Model`. Specify the S3 location of the model artifacts and Docker image containing the inference code. Amazon SageMaker uses this information to deploy the model in Amazon SageMaker. This method can also be used to create a Model for an Inference Pipeline if you pass the list of container definitions through the containers parameter. :param name: Name of the Amazon SageMaker `Model` to create. :type name: str :param role: 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. You must grant sufficient permissions to this role.

Parameters

- **container_defs** (*list[dict[str, str]] or [dict[str, str]]*) – A single container definition or a list of container definitions which will be invoked sequentially while performing the prediction. If the list contains only one container, then it'll be passed to SageMaker Hosting as the `PrimaryContainer` and otherwise, it'll be passed as `Containers`.You can also specify the return value of `sagemaker.get_container_def()` or `sagemaker.pipeline_container_def()`, which will used to create more advanced container configurations ,including model containers which need artifacts from S3.
- **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.
- **enable_network_isolation** (*bool*) – Wether the model requires network isolation or not.
- **primary_container** (*str or dict[str, str]*) – Docker image which defines the inference code. You can also specify the return value of `sagemaker.container_def()`, which is used to create more advanced container configurations, including model containers which need artifacts from S3. This field is deprecated, please use container_defs instead.

Returns Name of the Amazon SageMaker `Model` created.

Return type str

create_model_from_job(*training_job_name*, *name=None*, *role=None*, *primary_container_image=None*, *model_data_url=None*, *env=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*)

Create an Amazon SageMaker `Model` from a SageMaker Training Job.

Parameters

- **training_job_name** (*str*) – The Amazon SageMaker Training Job name.
- **name** (*str*) – The name of the SageMaker `Model` to create (default: None). If not specified, the training job name is used.
- **role** (*str*) – The `ExecutionRoleArn` IAM Role ARN for the `Model`, specified either by an IAM role name or role ARN. If None, the `RoleArn` from the SageMaker Training Job will be used.
- **primary_container_image** (*str*) – The Docker image reference (default: None). If None, it defaults to the Training Image in `training_job_name`.
- **model_data_url** (*str*) – S3 location of the model data (default: None). If None, defaults to the `ModelS3Artifacts` of `training_job_name`.
- **env** (*dict[string,string]*) – Model environment variables (default: {}).
- **vpc_config_override** (*dict[str, list[str]]*) – Optional override for VpcConfig set on the model. Default: use VpcConfig from training job. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

Returns The name of the created `Model`.

Return type str

create_model_package_from_algorithm(*name*, *description*, *algorithm_arn*, *model_data*)
Create a SageMaker Model Package from the results of training with an Algorithm Package

Parameters

- **name** (*str*) – ModelPackage name
- **description** (*str*) – Model Package description
- **algorithm_arn** (*str*) – arn or name of the algorithm used for training.
- **model_data** (*str*) – s3 URI to the model artifacts produced by training

wait_for_model_package(*model_package_name*, *poll=5*)
Wait for an Amazon SageMaker endpoint deployment to complete.

Parameters

- **endpoint** (*str*) – Name of the `Endpoint` to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeEndpoint` API.

Return type dict

create_endpoint_config(*name*, *model_name*, *initial_instance_count*, *instance_type*, *accelerator_type=None*)
Create an Amazon SageMaker endpoint configuration.

The endpoint configuration identifies the Amazon SageMaker model (created using the `CreateModel` API) and the hardware configuration on which to deploy the model. Provide this endpoint configuration to the `CreateEndpoint` API, which then launches the hardware and deploys the model.

Parameters

- **name** (*str*) – Name of the Amazon SageMaker endpoint configuration to create.
- **model_name** (*str*) – Name of the Amazon SageMaker `Model`.
- **initial_instance_count** (*int*) – Minimum number of EC2 instances to launch. The actual number of active instances for an endpoint at any given time varies due to autoscaling.
- **instance_type** (*str*) – Type of EC2 instance to launch, for example, 'ml.c4.xlarge'.
- **accelerator_type** (*str*) – Type of Elastic Inference accelerator to attach to the instance. For example, 'ml.eia1.medium'. For more information: https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html

Returns Name of the endpoint point configuration created.

Return type str

create_endpoint(*endpoint_name*, *config_name*, *wait=True*)
Create an Amazon SageMaker `Endpoint` according to the endpoint configuration specified in the request.

Once the `Endpoint` is created, client applications can send requests to obtain inferences. The endpoint configuration is created using the `CreateEndpointConfig` API.

Parameters

- **endpoint_name** (*str*) – Name of the Amazon SageMaker `Endpoint` being created.

- **config_name** (*str*) – Name of the Amazon SageMaker endpoint configuration to deploy.
- **wait** (*bool*) – Whether to wait for the endpoint deployment to complete before returning (default: True).

Returns Name of the Amazon SageMaker `Endpoint` created.

Return type str

delete_endpoint(*endpoint_name*)
Delete an Amazon SageMaker `Endpoint`.

Parameters **endpoint_name** (*str*) – Name of the Amazon SageMaker `Endpoint` to delete.

wait_for_job(*job*, *poll=5*)
Wait for an Amazon SageMaker training job to complete.

Parameters

- **job** (*str*) – Name of the training job to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeTrainingJob` API.

Return type (dict)

Raises `ValueError` – If the training job fails.

wait_for_compilation_job(*job*, *poll=5*)
Wait for an Amazon SageMaker Neo compilation job to complete.

Parameters

- **job** (*str*) – Name of the compilation job to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeCompilationJob` API.

Return type (dict)

Raises `ValueError` – If the compilation job fails.

wait_for_tuning_job(*job*, *poll=5*)
Wait for an Amazon SageMaker hyperparameter tuning job to complete.

Parameters

- **job** (*str*) – Name of the tuning job to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeHyperParameterTuningJob` API.

Return type (dict)

Raises `ValueError` – If the hyperparameter tuning job fails.

wait_for_transform_job(*job*, *poll=5*)
Wait for an Amazon SageMaker transform job to complete.

Parameters

- **job** (*str*) – Name of the transform job to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeTransformJob` API.

Return type (dict)

Raises `ValueError` – If the transform job fails.

wait_for_endpoint(*endpoint*, *poll=5*)
Wait for an Amazon SageMaker endpoint deployment to complete.

Parameters

- **endpoint** (*str*) – Name of the `Endpoint` to wait for.
- **poll** (*int*) – Polling interval in seconds (default: 5).

Returns Return value from the `DescribeEndpoint` API.

Return type dict

endpoint_from_job(*job_name*, *initial_instance_count*, *instance_type*, *deployment_image=None*, *name=None*, *role=None*, *wait=True*, *model_environment_vars=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*, *accelerator_type=None*)
Create an `Endpoint` using the results of a successful training job.

Specify the job name, Docker image containing the inference code, and hardware configuration to deploy the model. Internally the API, creates an Amazon SageMaker model (that describes the model artifacts and the Docker image containing inference code), endpoint configuration (describing the hardware to deploy for hosting the model), and creates an `Endpoint` (launches the EC2 instances and deploys the model on them). In response, the API returns the endpoint name to which you can send requests for inferences.

Parameters

- **job_name** (*str*) – Name of the training job to deploy the results of.
- **initial_instance_count** (*int*) – Minimum number of EC2 instances to launch. The actual number of active instances for an endpoint at any given time varies due to autoscaling.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, for example, 'ml.c4.xlarge'.
- **deployment_image** (*str*) – The Docker image which defines the inference code to be used as the entry point for accepting prediction requests. If not specified, uses the image used for the training job.
- **name** (*str*) – Name of the `Endpoint` to create. If not specified, uses the training job name.
- **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. You must grant sufficient permissions to this role.
- **wait** (*bool*) – Whether to wait for the endpoint deployment to complete before returning (default: True).
- **model_environment_vars** (*dict[str, str]*) – Environment variables to set on the model container (default: None).
- **vpc_config_override** (*dict[str, list[str]]*) – Overrides VpcConfig set on the model. Default: use VpcConfig from training job. * 'Subnets' (list[str]): List of subnet ids. * 'SecurityGroupIds' (list[str]): List of security group ids.

- **accelerator_type** (*str*) – Type of Elastic Inference accelerator to attach to the instance. For example, 'ml.eia1.medium'. For more information: https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html

Returns Name of the `Endpoint` that is created.

Return type str

endpoint_from_model_data(*model_s3_location*, *deployment_image*, *initial_instance_count*, *instance_type*, *name=None*, *role=None*, *wait=True*, *model_environment_vars=None*, *model_vpc_config=None*, *accelerator_type=None*)

Create and deploy to an `Endpoint` using existing model data stored in S3.

Parameters

- **model_s3_location** (*str*) – S3 URI of the model artifacts to use for the endpoint.
- **deployment_image** (*str*) – The Docker image which defines the runtime code to be used as the entry point for accepting prediction requests.
- **initial_instance_count** (*int*) – Minimum number of EC2 instances to launch. The actual number of active instances for an endpoint at any given time varies due to autoscaling.
- **instance_type** (*str*) – Type of EC2 instance to deploy to an endpoint for prediction, e.g. 'ml.c4.xlarge'.
- **name** (*str*) – Name of the `Endpoint` to create. If not specified, uses a name generated by combining the image name with a timestamp.
- **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. You must grant sufficient permissions to this role.
- **wait** (*bool*) – Whether to wait for the endpoint deployment to complete before returning (default: True).
- **model_environment_vars** (*dict[str, str]*) – Environment variables to set on the model container (default: None).
- **model_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.
- **accelerator_type** (*str*) – Type of Elastic Inference accelerator to attach to the instance. For example, 'ml.eia1.medium'. For more information: https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html

Returns Name of the `Endpoint` that is created.

Return type str

endpoint_from_production_variants(*name*, *production_variants*, *tags=None*, *wait=True*)

Create an SageMaker `Endpoint` from a list of production variants.

Parameters

- **name** (*str*) – The name of the `Endpoint` to create.
- **production_variants** (*list[dict[str, str]]*) – The list of production variants to deploy.

- **tags** (*list[dict[str, str]]*) – A list of key-value pairs for tagging the endpoint (default: None).
- **wait** (*bool*) – Whether to wait for the endpoint deployment to complete before returning (default: True).

Returns The name of the created `Endpoint`.

Return type str

expand_role(*role*)
Expand an IAM role name into an ARN.

If the role is already in the form of an ARN, then the role is simply returned. Otherwise we retrieve the full ARN and return it.

Parameters **role** (*str*) – An AWS IAM role (either name or full ARN).

Returns The corresponding AWS IAM role ARN.

Return type str

get_caller_identity_arn()
Returns the ARN user or role whose credentials are used to call the API. :returns: The ARN user or role :rtype: (str)

logs_for_job(*job_name*, *wait=False*, *poll=10*)
Display the logs for a given training job, optionally tailing them until the job is complete. If the output is a tty or a Jupyter cell, it will be color-coded based on which instance the log entry is from.

Parameters

- **job_name** (*str*) – Name of the training job to display the logs for.
- **wait** (*bool*) – Whether to keep looking for new log entries until the job completes (default: False).
- **poll** (*int*) – The interval in seconds between polling for new log entries and job completion (default: 5).

Raises `ValueError` – If waiting and the training job fails.

sagemaker.session.**container_def**(*image*, *model_data_url=None*, *env=None*)
Create a definition for executing a container as part of a SageMaker model. :param image: Docker image to run for this container. :type image: str :param model_data_url: S3 URI of data required by this container,

e.g. SageMaker training job model artifacts (default: None).

Parameters **env** (*dict[str, str]*) – Environment variables to set inside the container (default: None).

Returns A complete container definition object usable with the CreateModel API if passed via *PrimaryContainers* field.

Return type dict[str, str]

sagemaker.session.**pipeline_container_def**(*models*, *instance_type=None*)
Create a definition for executing a pipeline of containers as part of a SageMaker model. :param models: this will be a list of `sagemaker.Model` objects in the order the inference :type models: list[sagemaker.Model] :param should be invoked.: :param instance_type: The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge' (default: None). :type instance_type: str

Returns list of container definition objects usable with with the CreateModel API for inference pipelines if passed via *Containers* field.

Return type list[dict[str, str]]

sagemaker.session.**production_variant**(*model_name*, *instance_type*, *initial_instance_count=1*, *variant_name='AllTraffic'*, *initial_weight=1*, *accelerator_type=None*)

Create a production variant description suitable for use in a `ProductionVariant` list as part of a `CreateEndpointConfig` request.

Parameters

- **model_name** (*str*) – The name of the SageMaker model this production variant references.
- **instance_type** (*str*) – The EC2 instance type for this production variant. For example, 'ml.c4.8xlarge'.
- **initial_instance_count** (*int*) – The initial instance count for this production variant (default: 1).
- **variant_name** (*string*) – The `VariantName` of this production variant (default: 'AllTraffic').
- **initial_weight** (*int*) – The relative `InitialVariantWeight` of this production variant (default: 1).
- **accelerator_type** (*str*) – Type of Elastic Inference accelerator for this production variant. For example, 'ml.eia1.medium'. For more information: https://docs.aws.amazon.com/sagemaker/latest/dg/ei.html

Returns An SageMaker `ProductionVariant` description

Return type dict[str, str]

sagemaker.session.**get_execution_role**(*sagemaker_session=None*)

Return the role ARN whose credentials are used to call the API. Throws an exception if :param sagemaker_session: Current sagemaker session :type sagemaker_session: Session

Returns The role ARN

Return type (str)

class sagemaker.session.**s3_input**(*s3_data*, *distribution='FullyReplicated'*, *compression=None*, *content_type=None*, *record_wrapping=None*, *s3_data_type='S3Prefix'*, *input_mode=None*, *attribute_names=None*, *shuffle_config=None*)

Bases: `object`

Amazon SageMaker channel configurations for S3 data sources.

config

A SageMaker `DataSource` referencing a SageMaker `S3DataSource`.

Type dict[str, dict]

Create a definition for input data used by an SageMaker training job.

See AWS documentation on the `CreateTrainingJob` API for more details on the parameters.

Parameters

- **s3_data** (*str*) – Defines the location of s3 data to train on.
- **distribution** (*str*) – Valid values: 'FullyReplicated', 'ShardedByS3Key' (default: 'FullyReplicated').

- **compression** (*str*) – Valid values: 'Gzip', None (default: None). This is used only in Pipe input mode.
- **content_type** (*str*) – MIME type of the input data (default: None).
- **record_wrapping** (*str*) – Valid values: 'RecordIO' (default: None).
- **s3_data_type** (*str*) – Valid values: 'S3Prefix', 'ManifestFile', 'AugmentedManifestFile'. 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' or 'AugmentedManifestFile', then `s3_data` defines a single s3 manifest file or augmented manifest file (respectively), listing the s3 data to train on. Both the ManifestFile and AugmentedManifestFile formats are described in the SageMaker API

  documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/API_S3DataSource.html
- **input_mode** (*str*) – Optional override for this channel's input mode (default: None). By default, channels will use the input mode defined on `sagemaker.estimator.EstimatorBase.input_mode`, but they will ignore that setting if this parameter is set. * None - Amazon SageMaker will use the input mode specified in the `Estimator`. * '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.
- **attribute_names** (*list[str]*) – A list of one or more attribute names to use that are found in a specified AugmentedManifestFile.
- **shuffle_config** (`ShuffleConfig`) – If specified this configuration enables shuffling on this channel. See the SageMaker API documentation for more info: https://docs.aws.amazon.com/sagemaker/latest/dg/API_ShuffleConfig.html

class sagemaker.session.**ShuffleConfig**(*seed*)

Bases: `object`

Used to configure channel shuffling using a seed. See SageMaker documentation for more detail: https://docs.aws.amazon.com/sagemaker/latest/dg/API_ShuffleConfig.html

Create a ShuffleConfig. :param seed: the long value used to seed the shuffled sequence. :type seed: long

class sagemaker.session.**ModelContainer**(*model_data*, *image*, *env=None*)

Bases: `object`

Amazon SageMaker Model configurations for inference pipelines. .. attribute:: model_data

S3 Model artifact location

type str

image

Docker image URL in ECR

Type str

env

Environment variable mapping

Type dict[str,str]

Create a definition of a model which can be part of an Inference Pipeline :param model_data: The S3 location of a SageMaker model data `.tar.gz` file. :type model_data: str :param image: A Docker image URI. :type image: str :param env: Environment variables to run with `image` when hosted in SageMaker (default: None). :type env: dict[str, str]

1.8 Analytics

class sagemaker.analytics.**AnalyticsMetricsBase**
Bases: object

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

export_csv(*filename*)
Persists the analytics dataframe to a file.

Parameters **filename** (*str*) – The name of the file to save to.

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

Parameters **force_refresh** (*bool*) – Set to True to fetch the latest data from SageMaker API.

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

class sagemaker.analytics.**HyperparameterTuningJobAnalytics**(*hyperparameter_tuning_job_name*, *sagemaker_session=None*)
Bases: *sagemaker.analytics.AnalyticsMetricsBase*

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

Initialize a HyperparameterTuningJobAnalytics instance.

Parameters

- **hyperparameter_tuning_job_name** (*str*) – name of the HyperparameterTuningJob to analyze.
- **sagemaker_session** (sagemaker.session.Session) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, one is created using the default AWS configuration chain.

name
Name of the HyperparameterTuningJob being analyzed

clear_cache()
Clear the object of all local caches of API methods.

tuning_ranges
A dictionary describing the ranges of all tuned hyperparameters. The keys are the names of the hyperparameter, and the values are the ranges.

description(*force_refresh=False*)
Call DescribeHyperParameterTuningJob for the hyperparameter tuning job.

Parameters **force_refresh** (*bool*) – Set to True to fetch the latest data from SageMaker API.

Returns The Amazon SageMaker response for DescribeHyperParameterTuningJob.

Return type dict

training_job_summaries(*force_refresh=False*)
A (paginated) list of everything from ListTrainingJobsForTuningJob.

Parameters **force_refresh** (*bool*) – Set to True to fetch the latest data from SageMaker API.

Returns The Amazon SageMaker response for `ListTrainingJobsForTuningJob`.

Return type dict

class sagemaker.analytics.**TrainingJobAnalytics**(*training_job_name*, *metric_names=None*, *sagemaker_session=None*)

Bases: *sagemaker.analytics.AnalyticsMetricsBase*

Fetch training curve data from CloudWatch Metrics for a specific training job.

Initialize a `TrainingJobAnalytics` instance.

Parameters

- **training_job_name** (*str*) – name of the TrainingJob to analyze.
- **metric_names** (*list, optional*) – string names of all the metrics to collect for this training job. If not specified, then it will use all metric names configured for this job.
- **sagemaker_session** (sagemaker.session.Session) – Session object which manages interactions with Amazon SageMaker APIs and any other AWS services needed. If not specified, one is specified using the default AWS configuration chain.

CLOUDWATCH_NAMESPACE = '/aws/sagemaker/TrainingJobs'

name

Name of the TrainingJob being analyzed

clear_cache()

Clear the object of all local caches of API methods, so that the next time any properties are accessed they will be refreshed from the service.

CHAPTER 2

MXNet

A managed environment for MXNet training and hosting on Amazon SageMaker

2.1 MXNet

2.1.1 MXNet Estimator

class sagemaker.mxnet.estimator.**MXNet**(*entry_point*, *source_dir=None*, *hyperparameters=None*, *py_version='py2'*, *framework_version=None*, *image_name=None*, *distributions=None*, ***kwargs*)

Bases: *sagemaker.estimator.Framework*

Handle end-to-end training and deployment of custom MXNet code.

This `Estimator` executes an MXNet script in a managed MXNet execution environment, within a SageMaker Training Job. The managed MXNet environment is an Amazon-built Docker container that executes functions defined in the supplied `entry_point` Python script.

Training is started by calling `fit()` on this Estimator. After training is complete, calling `deploy()` creates a hosted SageMaker endpoint and returns an `MXNetPredictor` instance that can be used to perform inference against the hosted model.

Technical documentation on preparing MXNet scripts for SageMaker training and using the MXNet Estimator is available on the project home-page: https://github.com/aws/sagemaker-python-sdk

Parameters

- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to training. This should be compatible with either Python 2.7 or Python 3.5.
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from tne entry point file (default: None). Structure within this directory are preserved when training on Amazon SageMaker.

- **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.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2'). One of 'py2' or 'py3'.
- **framework_version** (*str*) – MXNet version you want to use for executing your model training code. List of supported versions https://github.com/aws/sagemaker-python-sdk#mxnet-sagemaker-estimators
- **image_name** (*str*) –

  If specified, the estimator will use this image for training and hosting, instead of selecting the appropriate SageMaker official image based on framework_version and py_version. It can be an ECR url or dockerhub image and tag.

  Examples: 123.dkr.ecr.us-west-2.amazonaws.com/my-custom-image:1.0 custom-image:latest.

  distributions (dict): A dictionary with information on how to run distributed training (default: None).
- ****kwargs** – Additional kwargs passed to the *Framework* constructor.

LATEST_VERSION = '1.3'

create_model(*model_server_workers=None*, *role=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*)

Create a SageMaker `MXNetModel` object that can be deployed to an `Endpoint`.

Parameters

- **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.
- **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.

Returns

A SageMaker MXNetModel object. See *MXNetModel()* for full details.

Return type *sagemaker.mxnet.model.MXNetModel*

2.1.2 MXNet Model

class sagemaker.mxnet.model.**MXNetModel**(*model_data*, *role*, *entry_point*, *image=None*, *py_version='py2'*, *framework_version='1.2'*, *predictor_cls=<class 'sagemaker.mxnet.model.MXNetPredictor'>*, *model_server_workers=None*, ***kwargs*)

Bases: `sagemaker.model.FrameworkModel`

An MXNet SageMaker `Model` that can be deployed to a SageMaker `Endpoint`.

Initialize an MXNetModel.

Parameters

- **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 an AWS resource.
- **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.
- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for MXNet will be used.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').
- **framework_version** (*str*) – MXNet version you want to use for executing your model training code.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- ****kwargs** – Keyword arguments passed to the `FrameworkModel` initializer.

prepare_container_def(*instance_type*, *accelerator_type=None*)

Return a container definition with framework configuration set in model environment variables.

Parameters

- **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 A container definition object usable with the CreateModel API.

Return type dict[str, str]

2.1.3 MXNet Predictor

class sagemaker.mxnet.model.**MXNetPredictor**(*endpoint_name*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A RealTimePredictor for inference against MXNet Endpoints.

This is able to serialize Python lists, dictionaries, and numpy arrays to multidimensional tensors for MXNet inference.

Initialize an `MXNetPredictor`.

Parameters

- **endpoint_name** (*str*) – The name of the endpoint to perform inference on.
- **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.

CHAPTER 3

TensorFlow

A managed environment for TensorFlow training and hosting on Amazon SageMaker

3.1 TensorFlow

3.1.1 TensorFlow Estimator

class sagemaker.tensorflow.estimator.**TensorFlow**(*training_steps=None*, *evaluation_steps=None*, *checkpoint_path=None*, *py_version='py2'*, *framework_version=None*, *model_dir=None*, *requirements_file=''*, *image_name=None*, *script_mode=False*, *distributions=None*, ***kwargs*)

Bases: *sagemaker.estimator.Framework*

Handle end-to-end training and deployment of user-provided TensorFlow code.

Initialize an `TensorFlow` estimator. :param training_steps: Perform this many steps of training. *None*, the default means train forever. :type training_steps: int :param evaluation_steps: Perform this many steps of evaluation. *None*, the default means that evaluation

runs until input from eval_input_fn is exhausted (or another exception is raised).

Parameters

- **checkpoint_path** (*str*) – Identifies S3 location where checkpoint data during model training can be saved (default: None). For distributed model training, this parameter is required.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').

- **framework_version** (*str*) – TensorFlow version you want to use for executing your model training code. List of supported versions https://github.com/aws/sagemaker-python-sdk#tensorflow-sagemaker-estimators
- **model_dir** (*str*) – S3 location where the checkpoint data and models can be exported to during training (default: None). If not specified a default S3 URI will be generated. It will be passed in the training script as one of the command line arguments.
- **requirements_file** (*str*) – Path to a `requirements.txt` file (default: ''). The path should be within and relative to `source_dir`. Details on the format can be found in the Pip User Guide.
- **image_name** (*str*) – If specified, the estimator will use this image for training and hosting, instead of selecting the appropriate SageMaker official image based on framework_version and py_version. It can be an ECR url or dockerhub image and tag.

  Examples: 123.dkr.ecr.us-west-2.amazonaws.com/my-custom-image:1.0
  custom-image:latest.
- **script_mode** (*bool*) – If set to True will the estimator will use the Script Mode containers (default: False). This will be ignored if py_version is set to 'py3'.
- **distributions** (*dict*) – A dictionary with information on how to run distributed training (default: None). Currently we only support distributed training with parameter servers. To enable it use the following setup:

  { 'parameter_server': {

  'enabled': True

  }

  }
- ****kwargs** – Additional kwargs passed to the Framework constructor.

LATEST_VERSION = '1.12'

fit(*inputs=None*, *wait=True*, *logs=True*, *job_name=None*, *run_tensorboard_locally=False*)

Train a model using the input training dataset.

See *fit()* for more details.

Parameters

- **inputs** (*str or dict or sagemaker.session.s3_input*) – Information about the training data. This can be one of three types: (str) - the S3 location where training data is saved. (dict[str, str] or dict[str, sagemaker.session.s3_input]) - If using multiple channels for

  training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.

  (sagemaker.session.s3_input) - channel configuration for S3 data sources that can provide additional information as well as the path to the training dataset. See *sagemaker.session.s3_input()* for full details.
- **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.
- **run_tensorboard_locally** (*bool*) – Whether to execute TensorBoard in a different process with downloaded checkpoint information (default: False). This is an experimental feature, and requires TensorBoard and AWS CLI to be installed. It terminates TensorBoard when execution ends.

create_model(*model_server_workers=None*, *role=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*, *endpoint_type=None*)

Create a SageMaker `TensorFlowModel` object that can be deployed to an `Endpoint`.

Parameters

- **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.
- **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.
- **endpoint_type** – Optional. Selects the software stack used by the inference server. If not specified, the model will be configured to use the default SageMaker model server. If 'tensorflow-serving', the model will be configured to use the SageMaker Tensorflow Serving container.

Returns

A SageMaker TensorFlowModel object. See *TensorFlowModel()* for full details.

Return type *sagemaker.tensorflow.model.TensorFlowModel*

hyperparameters()

Return hyperparameters used by your custom TensorFlow code during model training.

train_image()

Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

3.1.2 TensorFlow Model

class sagemaker.tensorflow.model.**TensorFlowModel**(*model_data*, *role*, *entry_point*, *image=None*, *py_version='py2'*, *framework_version='1.11'*, *predictor_cls=<class 'sagemaker.tensorflow.model.TensorFlowPredictor'>*, *model_server_workers=None*, ***kwargs*)

Bases: `sagemaker.model.FrameworkModel`

Initialize an TensorFlowModel.

Parameters

- **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 an AWS resource.
- **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.
- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for TensorFlow will be used.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').
- **framework_version** (*str*) – TensorFlow version you want to use for executing your model training code.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- ****kwargs** – Keyword arguments passed to the `FrameworkModel` initializer.

prepare_container_def(*instance_type*, *accelerator_type=None*)

Return a container definition with framework configuration set in model environment variables.

This also uploads user-supplied code to S3.

Parameters

- **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 A container definition object usable with the CreateModel API.

Return type dict[str, str]

3.1.3 TensorFlow Predictor

class sagemaker.tensorflow.model.**TensorFlowPredictor**(*endpoint_name*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A `RealTimePredictor` for inference against TensorFlow ''Endpoint''s.

This is able to serialize Python lists, dictionaries, and numpy arrays to multidimensional tensors for MXNet inference

Initialize an `TensorFlowPredictor`.

Parameters

- **endpoint_name** (*str*) – The name of the endpoint to perform inference on.
- **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.

3.1.4 TensorFlow Serving Model

class sagemaker.tensorflow.serving.**Model**(*model_data*, *role*, *image=None*, *framework_version='1.11'*, *container_log_level=None*, *predictor_cls=<class 'sagemaker.tensorflow.serving.Predictor'>*, ***kwargs*)

Bases: *sagemaker.model.Model*

Initialize a Model.

Parameters

- **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 APIs that create Amazon SageMaker endpoints use this role to access model artifacts.
- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for TensorFlow Serving will be used.
- **framework_version** (*str*) – Optional. TensorFlow Serving version you want to use.
- **container_log_level** (*int*) – Log level to use within the container (default: logging.ERROR). Valid values are defined in the Python logging module.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- ****kwargs** – Keyword arguments passed to the `Model` initializer.

FRAMEWORK_NAME = 'tensorflow-serving'

LOG_LEVEL_PARAM_NAME = 'SAGEMAKER_TFS_NGINX_LOGLEVEL'

LOG_LEVEL_MAP = {10: 'debug', 20: 'info', 30: 'warn', 40: 'error', 50: 'crit'}

prepare_container_def(*instance_type*, *accelerator_type=None*)

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()`.

Parameters

- **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 A container definition object usable with the CreateModel API.

Return type dict

3.1.5 TensorFlow Serving Predictor

class sagemaker.tensorflow.serving.**Predictor**(*endpoint_name*, *sagemaker_session=None*, *serializer=<sagemaker.predictor._JsonSerializer object>*, *deserializer=<sagemaker.predictor._JsonDeserializer object>*, *content_type=None*, *model_name=None*, *model_version=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A `RealTimePredictor` implementation for inference against TensorFlow Serving endpoints.

Initialize a `TFSPredictor`. See `sagemaker.RealTimePredictor` for more info about parameters.

Parameters

- **endpoint_name** (*str*) – The name of the endpoint to perform inference on.
- **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.
- **serializer** (*callable*) – Optional. Default serializes input data to json. Handles dicts, lists, and numpy arrays.
- **deserializer** (*callable*) – Optional. Default parses the response using `json.load(...)`.
- **content_type** (*str*) – Optional. The "ContentType" for invocation requests. If specified, overrides the `content_type` from the serializer (default: None).
- **model_name** (*str*) – Optional. The name of the SavedModel model that should handle the request. If not specified, the endpoint's default model will handle the request.
- **model_version** (*str*) – Optional. The version of the SavedModel model that should handle the request. If not specified, the latest version of the model will be used.

classify(*data*)

regress(*data*)

predict(*data*, *initial_args=None*)

Return the inference from the specified endpoint.

Parameters

- **data** (*object*) – Input data for which you want the model to provide inference. If a serializer was specified when creating the RealTimePredictor, 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** (*dict[str,str]*) – Optional. Default arguments for boto3 `invoke_endpoint` call. Default is None (no default arguments).

Returns

Inference for the given input. If a deserializer was specified when creating the RealTimePredictor, the result of the deserializer is returned. Otherwise the response returns the sequence of bytes as is.

Return type object

CHAPTER 4

Scikit-Learn

A managed enrionment for Scikit-learn training and hosting on Amazon SageMaker

4.1 Scikit Learn

4.1.1 Scikit Learn Estimator

class sagemaker.sklearn.estimator.**SKLearn**(*entry_point*, *framework_version='0.20.0'*, *source_dir=None*, *hyperparameters=None*, *py_version='py3'*, *image_name=None*, ***kwargs*)

Bases: *sagemaker.estimator.Framework*

Handle end-to-end training and deployment of custom Scikit-learn code.

This `Estimator` executes an Scikit-learn script in a managed Scikit-learn execution environment, within a SageMaker Training Job. The managed Scikit-learn environment is an Amazon-built Docker container that executes functions defined in the supplied `entry_point` Python script.

Training is started by calling `fit()` on this Estimator. After training is complete, calling `deploy()` creates a hosted SageMaker endpoint and returns an `SKLearnPredictor` instance that can be used to perform inference against the hosted model.

Technical documentation on preparing Scikit-learn scripts for SageMaker training and using the Scikit-learn Estimator is available on the project home-page: https://github.com/aws/sagemaker-python-sdk

Parameters

- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to training. This should be compatible with either Python 2.7 or Python 3.5.
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from tne entry point file (default: None). Structure within this directory are preserved when training on Amazon SageMaker.

- **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.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2'). One of 'py2' or 'py3'.
- **framework_version** (*str*) – Scikit-learn version you want to use for executing your model training code. List of supported versions https://github.com/aws/sagemaker-python-sdk#sklearn-sagemaker-estimators
- **image_name** (*str*) – If specified, the estimator will use this image for training and hosting, instead of selecting the appropriate SageMaker official image based on framework_version and py_version. It can be an ECR url or dockerhub image and tag. .. rubric:: Examples

  123.dkr.ecr.us-west-2.amazonaws.com/my-custom-image:1.0 custom-image:latest.
- ****kwargs** – Additional kwargs passed to the *Framework* constructor.

create_model(*model_server_workers=None*, *role=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*, ***kwargs*)

Create a SageMaker `SKLearnModel` object that can be deployed to an `Endpoint`.

Parameters

- **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.
- **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.
- ****kwargs** – Passed to initialization of `SKLearnModel`.

Returns

A SageMaker SKLearnModel object. See *SKLearnModel()* for full details.

Return type *sagemaker.sklearn.model.SKLearnModel*

4.1.2 Scikit Learn Model

class sagemaker.sklearn.model.**SKLearnModel**(*model_data*, *role*, *entry_point*, *image=None*, *py_version='py3'*, *framework_version='0.20.0'*, *predictor_cls=<class 'sagemaker.sklearn.model.SKLearnPredictor'>*, *model_server_workers=None*, ***kwargs*)

Bases: `sagemaker.model.FrameworkModel`

An Scikit-learn SageMaker `Model` that can be deployed to a SageMaker `Endpoint`.

Initialize an SKLearnModel.

Parameters

- **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 an AWS resource.
- **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.
- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for Scikit-learn will be used.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').
- **framework_version** (*str*) – Scikit-learn version you want to use for executing your model training code.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- ****kwargs** – Keyword arguments passed to the `FrameworkModel` initializer.

prepare_container_def(*instance_type*, *accelerator_type=None*)

Return a container definition with framework configuration set in model environment variables.

Parameters

- **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 A container definition object usable with the CreateModel API.

Return type dict[str, str]

4.1.3 Scikit Learn Predictor

class sagemaker.sklearn.model.**SKLearnPredictor**(*endpoint_name*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A RealTimePredictor for inference against Scikit-learn Endpoints.

This is able to serialize Python lists, dictionaries, and numpy arrays to multidimensional tensors for Scikit-learn inference.

Initialize an `SKLearnPredictor`.

Parameters

- **endpoint_name** (*str*) – The name of the endpoint to perform inference on.

- **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.

CHAPTER 5

PyTorch

A managed environment for PyTorch training and hosting on Amazon SageMaker

5.1 PyTorch

5.1.1 PyTorch Estimator

class sagemaker.pytorch.estimator.**PyTorch**(*entry_point*, *source_dir=None*, *hyperparameters=None*, *py_version='py3'*, *framework_version=None*, *image_name=None*, ***kwargs*)

Bases: *sagemaker.estimator.Framework*

Handle end-to-end training and deployment of custom PyTorch code.

This `Estimator` executes an PyTorch script in a managed PyTorch execution environment, within a SageMaker Training Job. The managed PyTorch environment is an Amazon-built Docker container that executes functions defined in the supplied `entry_point` Python script.

Training is started by calling `fit()` on this Estimator. After training is complete, calling `deploy()` creates a hosted SageMaker endpoint and returns an `PyTorchPredictor` instance that can be used to perform inference against the hosted model.

Technical documentation on preparing PyTorch scripts for SageMaker training and using the PyTorch Estimator is available on the project home-page: https://github.com/aws/sagemaker-python-sdk

Parameters

- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to training. This should be compatible with either Python 2.7 or Python 3.5.
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from tne entry point file (default: None). Structure within this directory are preserved when training on Amazon SageMaker.

- **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.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py3'). One of 'py2' or 'py3'.
- **framework_version** (*str*) – PyTorch version you want to use for executing your model training code. List of supported versions https://github.com/aws/sagemaker-python-sdk#pytorch-sagemaker-estimators
- **image_name** (*str*) – If specified, the estimator will use this image for training and hosting, instead of selecting the appropriate SageMaker official image based on framework_version and py_version. It can be an ECR url or dockerhub image and tag. .. rubric:: Examples

  123.dkr.ecr.us-west-2.amazonaws.com/my-custom-image:1.0 custom-image:latest.
- ****kwargs** – Additional kwargs passed to the *Framework* constructor.

LATEST_VERSION = '1.0'

create_model(*model_server_workers=None*, *role=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*)

Create a SageMaker `PyTorchModel` object that can be deployed to an `Endpoint`.

Parameters

- **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.
- **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.

Returns

A SageMaker PyTorchModel object. See *PyTorchModel()* for full details.

Return type *sagemaker.pytorch.model.PyTorchModel*

5.1.2 PyTorch Model

class sagemaker.pytorch.model.**PyTorchModel**(*model_data*, *role*, *entry_point*, *image=None*, *py_version='py3'*, *framework_version='0.4'*, *predictor_cls=<class 'sagemaker.pytorch.model.PyTorchPredictor'>*, *model_server_workers=None*, ***kwargs*)

Bases: `sagemaker.model.FrameworkModel`

An PyTorch SageMaker `Model` that can be deployed to a SageMaker `Endpoint`.

Initialize an PyTorchModel.

Parameters

- **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 an AWS resource.
- **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.
- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for PyTorch will be used.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py3').
- **framework_version** (*str*) – PyTorch version you want to use for executing your model training code.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- ****kwargs** – Keyword arguments passed to the `FrameworkModel` initializer.

prepare_container_def(*instance_type*, *accelerator_type=None*)

Return a container definition with framework configuration set in model environment variables.

Parameters

- **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 A container definition object usable with the CreateModel API.

Return type dict[str, str]

5.1.3 PyTorch Predictor

class sagemaker.pytorch.model.**PyTorchPredictor**(*endpoint_name*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A RealTimePredictor for inference against PyTorch Endpoints.

This is able to serialize Python lists, dictionaries, and numpy arrays to multidimensional tensors for PyTorch inference.

Initialize an `PyTorchPredictor`.

Parameters

- **endpoint_name** (*str*) – The name of the endpoint to perform inference on.

- **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.

CHAPTER 6

Chainer

A managed environment for Chainer training and hosting on Amazon SageMaker

6.1 Chainer

6.1.1 Chainer Estimator

class sagemaker.chainer.estimator.**Chainer**(*entry_point*, *use_mpi=None*, *num_processes=None*, *process_slots_per_host=None*, *additional_mpi_options=None*, *source_dir=None*, *hyperparameters=None*, *py_version='py3'*, *framework_version=None*, *image_name=None*, ***kwargs*)

Bases: *sagemaker.estimator.Framework*

Handle end-to-end training and deployment of custom Chainer code.

This `Estimator` executes an Chainer script in a managed Chainer execution environment, within a SageMaker Training Job. The managed Chainer environment is an Amazon-built Docker container that executes functions defined in the supplied `entry_point` Python script.

Training is started by calling `fit()` on this Estimator. After training is complete, calling `deploy()` creates a hosted SageMaker endpoint and returns an `ChainerPredictor` instance that can be used to perform inference against the hosted model.

Technical documentation on preparing Chainer scripts for SageMaker training and using the Chainer Estimator is available on the project home-page: https://github.com/aws/sagemaker-python-sdk

Parameters

- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to training. This should be compatible with either Python 2.7 or Python 3.5.

- **use_mpi** (*bool*) – If true, entry point is run as an MPI script. By default, the Chainer Framework runs the entry point with 'mpirun' if more than one instance is used.
- **num_processes** (*int*) – Total number of processes to run the entry point with. By default, the Chainer Framework runs one process per GPU (on GPU instances), or one process per host (on CPU instances).
- **process_slots_per_host** (*int*) – The number of processes that can run on each instance. By default, this is set to the number of GPUs on the instance (on GPU instances), or one (on CPU instances).
- **additional_mpi_options** (*str*) – String of options to the 'mpirun' command used to run the entry point. For example, '-X NCCL_DEBUG=WARN' will pass that option string to the mpirun command.
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from tne entry point file (default: None). Structure within this directory are preserved when training on Amazon SageMaker.
- **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.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2'). One of 'py2' or 'py3'.
- **framework_version** (*str*) – Chainer version you want to use for executing your model training code. List of supported versions https://github.com/aws/sagemaker-python-sdk#chainer-sagemaker-estimators
- **image_name** (*str*) – If specified, the estimator will use this image for training and hosting, instead of selecting the appropriate SageMaker official image based on framework_version and py_version. It can be an ECR url or dockerhub image and tag. .. rubric:: Examples

  123.dkr.ecr.us-west-2.amazonaws.com/my-custom-image:1.0 custom-image:latest.
- ****kwargs** – Additional kwargs passed to the *Framework* constructor.

LATEST_VERSION = '5.0.0'

hyperparameters()
Return hyperparameters used by your custom Chainer code during training.

create_model(*model_server_workers=None*, *role=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*)
Create a SageMaker `ChainerModel` object that can be deployed to an `Endpoint`.

Parameters

- **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.
- **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.

Returns

A SageMaker `ChainerModel` object. See `ChainerModel()` for full details.

Return type *sagemaker.chainer.model.ChainerModel*

6.1.2 Chainer Model

class sagemaker.chainer.model.**ChainerModel**(*model_data*, *role*, *entry_point*, *image=None*, *py_version='py3'*, *framework_version='4.1.0'*, *predictor_cls=<class 'sagemaker.chainer.model.ChainerPredictor'>*, *model_server_workers=None*, ***kwargs*)

Bases: `sagemaker.model.FrameworkModel`

An Chainer SageMaker `Model` that can be deployed to a SageMaker `Endpoint`.

Initialize an ChainerModel.

Parameters

- **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 an AWS resource.
- **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.
- **image** (*str*) – A Docker image URI (default: None). If not specified, a default image for Chainer will be used.
- **py_version** (*str*) – Python version you want to use for executing your model training code (default: 'py2').
- **framework_version** (*str*) – Chainer version you want to use for executing your model training code.
- **predictor_cls** (*callable[str, sagemaker.session.Session]*) – A function to call to create a predictor with an endpoint name and SageMaker `Session`. If specified, `deploy()` returns the result of invoking this function on the created endpoint name.
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- ****kwargs** – Keyword arguments passed to the `FrameworkModel` initializer.

prepare_container_def(*instance_type*, *accelerator_type=None*)

Return a container definition with framework configuration set in model environment variables.

Parameters

- **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 A container definition object usable with the CreateModel API.

Return type dict[str, str]

6.1.3 Chainer Predictor

class sagemaker.chainer.model.**ChainerPredictor**(*endpoint_name*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

A RealTimePredictor for inference against Chainer Endpoints.

This is able to serialize Python lists, dictionaries, and numpy arrays to multidimensional tensors for Chainer inference.

Initialize an `ChainerPredictor`.

Parameters

- **endpoint_name** (*str*) – The name of the endpoint to perform inference on.
- **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.

CHAPTER 7

Reinforcement Learning

A managed environment for Reinforcement Learning training and hosting on Amazon SageMaker

7.1 RLEstimator

7.1.1 RLEstimator Estimator

class sagemaker.rl.estimator.**RLEstimator**(*entry_point*, *toolkit=None*, *toolkit_version=None*, *framework=None*, *source_dir=None*, *hyperparameters=None*, *image_name=None*, *metric_definitions=None*, ***kwargs*)

Bases: *sagemaker.estimator.Framework*

Handle end-to-end training and deployment of custom RLEstimator code.

This Estimator executes an RLEstimator script in a managed Reinforcement Learning (RL) execution environment within a SageMaker Training Job. The managed RL environment is an Amazon-built Docker container that executes functions defined in the supplied `entry_point` Python script.

Training is started by calling `fit()` on this Estimator. After training is complete, calling `deploy()` creates a hosted SageMaker endpoint and based on the specified framework returns an `MXNetPredictor` or `Predictor` instance that can be used to perform inference against the hosted model.

Technical documentation on preparing RLEstimator scripts for SageMaker training and using the RLEstimator is available on the project homepage: https://github.com/aws/sagemaker-python-sdk

Parameters

- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point to training. This should be compatible with Python 3.5 for MXNet or Python 3.6 for TensorFlow.
- **toolkit** (*sagemaker.rl.RLToolkit*) – RL toolkit you want to use for executing your model training code.

- **toolkit_version** (*str*) – RL toolkit version you want to be use for executing your model training code.
- **framework** (*sagemaker.rl.RLFramework*) – Framework (MXNet or TensorFlow) you want to be used as a toolkit backed for reinforcement learning training.
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from the entry point file (default: None). Structure within this directory is preserved when training on Amazon SageMaker.
- **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.
- **image_name** (*str*) – An ECR url. If specified, the estimator will use this image for training and hosting, instead of selecting the appropriate SageMaker official image based on framework_version and py_version. Example: 123.dkr.ecr.us-west-2.amazonaws.com/my-custom-image:1.0
- **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.
- ****kwargs** – Additional kwargs passed to the *Framework* constructor.

COACH_LATEST_VERSION = '0.11.0'

RAY_LATEST_VERSION = '0.5.3'

create_model(*role=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*, *entry_point=None*, *source_dir=None*, *dependencies=None*)

Create a SageMaker `RLEstimatorModel` object that can be deployed to an Endpoint.

Parameters

- **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.
- **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.
- **entry_point** (*str*) – Path (absolute or relative) to the Python source file which should be executed as the entry point for MXNet hosting. This should be compatible with Python 3.5 (default: self.entry_point)
- **source_dir** (*str*) – Path (absolute or relative) to a directory with any other training source code dependencies aside from tne entry point file (default: self.source_dir). Structure within this directory are preserved when hosting on Amazon SageMaker.
- **dependencies** (*list[str]*) – A list of paths to directories (absolute or relative) with any additional libraries that will be exported to the container (default: self.dependencies). The library folders will be copied to SageMaker in the same folder where the entry_point is copied. If the `source_dir` points to S3, code will be uploaded and the S3 location will be used instead.

Returns

Depending on input parameters returns one of the following:

- **sagemaker.model.FrameworkModel - in case image_name was specified** on the estimator;
- **sagemaker.mxnet.MXNetModel - if image_name wasn't specified and** MXNet was used as RL backend;
- **sagemaker.tensorflow.serving.Model - if image_name wasn't specified and** TensorFlow was used as RL backend.

Return type sagemaker.model.FrameworkModel

train_image()

Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

hyperparameters()

Return hyperparameters used by your custom TensorFlow code during model training.

classmethod default_metric_definitions(*toolkit*)

Provides default metric definitions based on provided toolkit.

Parameters **toolkit** (*sagemaker.rl.RLToolkit*) – RL Toolkit to be used for training.

Returns metric definitions

Return type list

CHAPTER 8

SparkML Serving

A managed environment for SparkML hosting on Amazon SageMaker

8.1 SparkML Serving

8.1.1 SparkML Model

class sagemaker.sparkml.model.**SparkMLModel**(*model_data*, *role=None*, *spark_version=2.2*, *sagemaker_session=None*, ***kwargs*)

Bases: *sagemaker.model.Model*

Model data and S3 location holder for MLeap serialized SparkML model. Calling *deploy()* creates an Endpoint and return a Predictor to performs predictions against an MLeap serialized SparkML model .

Initialize a SparkMLModel.. :param model_data: The S3 location of a SageMaker model data `.tar.gz` file. For SparkML, this will be the

output that has been produced by the Spark job after serializing the Model via MLeap.

Parameters

- **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.
- **spark_version** (*str*) – Spark version you want to use for executing the inference (default: '2.2').
- **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. For local mode, please do not pass this variable.

8.1.2 SparkML Predictor

class sagemaker.sparkml.model.**SparkMLPredictor**(*endpoint*, *sagemaker_session=None*)
Bases: *sagemaker.predictor.RealTimePredictor*

Performs predictions against an MLeap serialized SparkML model.

The implementation of *predict()* in this *RealTimePredictor* requires a json as input. The input should follow the json format as documented.

predict() returns a csv output, comma separated if the output is a list.

Initializes a SparkMLPredictor which should be used with SparkMLModel to perform predictions against SparkML models serialized via MLeap. The response is returned in text/csv format which is the default response format for SparkML Serving container. :param endpoint: The name of the endpoint to perform inference on. :type endpoint: str :param sagemaker_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.

CHAPTER 9

SageMaker First-Party Algorithms

Amazon provides implementations of some common machine learning algortithms optimized for GPU architecture and massive datasets.

9.1 Amazon Estimators

Base class for Amazon Estimator implementations

class sagemaker.amazon.amazon_estimator.**AmazonAlgorithmEstimatorBase**(*role*, *train_instance_count*, *train_instance_type*, *data_location=None*, ***kwargs*)

Bases: *sagemaker.estimator.EstimatorBase*

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

Initialize an AmazonAlgorithmEstimatorBase.

Parameters **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.

feature_dim
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

mini_batch_size
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

repo_name = None

repo_version = None

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

hyperparameters()
Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

data_location

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)
Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 *record_set()* to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`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.

record_set(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

9.2 FactorizationMachines

The Amazon SageMaker Factorization Machines algorithm.

class sagemaker.**FactorizationMachines**(*role*, *train_instance_count*, *train_instance_type*, *num_factors*, *predictor_type*, *epochs=None*, *clip_gradient=None*, *eps=None*, *rescale_grad=None*, *bias_lr=None*, *linear_lr=None*, *factors_lr=None*, *bias_wd=None*, *linear_wd=None*, *factors_wd=None*, *bias_init_method=None*, *bias_init_scale=None*, *bias_init_sigma=None*, *bias_init_value=None*, *linear_init_method=None*, *linear_init_scale=None*, *linear_init_sigma=None*, *linear_init_value=None*, *factors_init_method=None*, *factors_init_scale=None*, *factors_init_sigma=None*, *factors_init_value=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

Factorization Machines is `Estimator` for general-purpose supervised learning.

Amazon SageMaker Factorization Machines is a general-purpose supervised learning algorithm that you can use for both classification and regression tasks. It is an extension of a linear model that is designed to parsimoniously capture interactions between features within high dimensional sparse datasets.

This Estimator may be fit via calls to *fit()*. It requires Amazon `Record` protobuf serialized data to be stored in S3. There is an utility *record_set()* that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `FactorizationMachinesPredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

FactorizationMachines Estimators can be configured by setting hyperparameters. The available hyperparameters for FactorizationMachines are documented below.

For further information on the AWS FactorizationMachines algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/fact-machines.html

Parameters

- **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 accessing AWS resource.

- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_factors** (*int*) – Dimensionality of factorization.
- **predictor_type** (*str*) – Type of predictor 'binary_classifier' or 'regressor'.
- **epochs** (*int*) – Number of training epochs to run.
- **clip_gradient** (*float*) – Optimizer parameter. Clip the gradient by projecting onto the box [-clip_gradient, +clip_gradient]
- **eps** (*float*) – Optimizer parameter. Small value to avoid division by 0.
- **rescale_grad** (*float*) – Optimizer parameter. If set, multiplies the gradient with rescale_grad before updating. Often choose to be 1.0/batch_size.
- **bias_lr** (*float*) – Non-negative learning rate for the bias term.
- **linear_lr** (*float*) – Non-negative learning rate for linear terms.
- **factors_lr** (*float*) – Noon-negative learning rate for factorization terms.
- **bias_wd** (*float*) – Non-negative weight decay for the bias term.
- **linear_wd** (*float*) – Non-negative weight decay for linear terms.
- **factors_wd** (*float*) – Non-negative weight decay for factorization terms.
- **bias_init_method** (*string*) – Initialization method for the bias term: 'normal', 'uniform' or 'constant'.
- **bias_init_scale** (*float*) – Non-negative range for initialization of the bias term that takes effect when bias_init_method parameter is 'uniform'
- **bias_init_sigma** (*float*) – Non-negative standard deviation for initialization of the bias term that takes effect when bias_init_method parameter is 'normal'.
- **bias_init_value** (*float*) – Initial value of the bias term that takes effect when bias_init_method parameter is 'constant'.
- **linear_init_method** (*string*) – Initialization method for linear term: 'normal', 'uniform' or 'constant'.
- **linear_init_scale** (*float*) – Non-negative range for initialization of linear terms that takes effect when linear_init_method parameter is 'uniform'.
- **linear_init_sigma** (*float*) – Non-negative standard deviation for initialization of linear terms that takes effect when linear_init_method parameter is 'normal'.
- **linear_init_value** (*float*) – Initial value of linear terms that takes effect when linear_init_method parameter is 'constant'.
- **factors_init_method** (*string*) – Initialization method for factorization term: 'normal', 'uniform' or 'constant'.
- **factors_init_scale** (*float*) – Non-negative range for initialization of factorization terms that takes effect when factors_init_method parameter is 'uniform'.
- **factors_init_sigma** (*float*) – Non-negative standard deviation for initialization of factorization terms that takes effect when factors_init_method parameter is 'normal'.

- **factors_init_value** (*float*) – Initial value of factorization terms that takes effect when factors_init_method parameter is 'constant'.
- ****kwargs** – base class keyword argument values.

repo_name = 'factorization-machines'

repo_version = 1

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()
Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)
Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()
Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)
Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`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.

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

hyperparameters()
Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()

Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

training_job_analytics

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)

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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Return a FactorizationMachinesModel referencing the latest s3 model data produced by this Estimator.

Parameters **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.

class sagemaker.**FactorizationMachinesModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)

Bases: *sagemaker.model.Model*

Reference S3 model data created by FactorizationMachines estimator. Calling *deploy()* creates an Endpoint and returns *FactorizationMachinesPredictor*.

class sagemaker.**FactorizationMachinesPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Performs binary-classification or regression prediction from input vectors.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy ndarray as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

predict() returns a list of Record objects, one for each row in the input ndarray. The prediction is stored in the "score" key of the Record.label field. Please refer to the formats details described: https://docs.aws.amazon.com/sagemaker/latest/dg/fm-in-formats.html

9.3 IP Insights

The Amazon SageMaker IP Insights algorithm.

class sagemaker.**IPInsights**(*role*, *train_instance_count*, *train_instance_type*, *num_entity_vectors*, *vector_dim*, *batch_metrics_publish_interval=None*, *epochs=None*, *learning_rate=None*, *num_ip_encoder_layers=None*, *random_negative_sampling_rate=None*, *shuffled_negative_sampling_rate=None*, *weight_decay=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

This estimator is for IP Insights, an unsupervised algorithm that learns usage patterns of IP addresses.

This Estimator may be fit via calls to *fit()*. It requires CSV data to be stored in S3.

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking deploy(). As well as deploying an Endpoint, deploy returns a IPInsightPredictor object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

IPInsights Estimators can be configured by setting hyperparamters. The available hyperparamters are documented below.

For further information on the AWS IPInsights algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/ip-insights-hyperparameters.html

Parameters

- **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 accessing AWS resource.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.m5.xlarge'.
- **num_entity_vectors** (*int*) – Required. The number of embeddings to train for entities accessing online resources. We recommend 2x the total number of unique entity IDs.
- **vector_dim** (*int*) – Required. The size of the embedding vectors for both entity and IP addresses.
- **batch_metrics_publish_interval** (*int*) – Optional. The period at which to publish metrics (batches).
- **epochs** (*int*) – Optional. Maximum number of passes over the training data.
- **learning_rate** (*float*) – Optional. Learning rate for the optimizer.
- **num_ip_encoder_layers** (*int*) – Optional. The number of fully-connected layers to encode IP address embedding.
- **random_negative_sampling_rate** (*int*) – Optional. The ratio of random negative samples to draw during training. Random negative samples are randomly drawn IPv4 addresses.
- **shuffled_negative_sampling_rate** (*int*) – Optional. The ratio of shuffled negative samples to draw during training. Shuffled negative samples are IP addresses picked from within a batch.
- **weight_decay** (*float*) – Optional. Weight decay coefficient. Adds L2 regularization.
- ****kwargs** – base class keyword argument values.

repo_name = 'ipinsights'

repo_version = 1

MINI_BATCH_SIZE = 10000

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()
Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)
Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()
Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)
Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (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.

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

hyperparameters()
Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)
Build a RecordSet from a numpy ndarray matrix and label vector.

For the 2D ndarray train, each row is converted to a 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 train_instance_count property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)
Create a model for the latest s3 model produced by this estimator.

Parameters **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.

Returns references the latest s3 model data produced by this estimator.

Return type `IPInsightsModel`

class sagemaker.**IPInsightsModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)
Bases: *sagemaker.model.Model*

Reference IPInsights s3 model data. Calling *deploy()* creates an Endpoint and returns a Predictor that calculates anomaly scores for data points.

class sagemaker.**IPInsightsPredictor**(*endpoint*, *sagemaker_session=None*)
Bases: *sagemaker.predictor.RealTimePredictor*

Returns dot product of entity and IP address embeddings as a score for compatibility.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain two columns. The first column should contain the entity ID. The second column should contain the IPv4 address in dot notation.

9.4 K-means

The Amazon SageMaker K-means algorithm.

class sagemaker.**KMeans**(*role*, *train_instance_count*, *train_instance_type*, *k*, *init_method=None*, *max_iterations=None*, *tol=None*, *num_trials=None*, *local_init_method=None*, *half_life_time_size=None*, *epochs=None*, *center_factor=None*, *eval_metrics=None*, ***kwargs*)
Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

A k-means clustering `AmazonAlgorithmEstimatorBase`. Finds k clusters of data in an unlabeled dataset.

This Estimator may be fit via calls to `fit_ndarray()` or *fit()*. The former allows a KMeans model to be fit on a 2-dimensional numpy array. The latter requires Amazon `Record` protobuf serialized data to be stored in S3.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html.

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, `deploy` returns a `KMeansPredictor` object that can be used to k-means cluster assignments, using the trained k-means model hosted in the SageMaker Endpoint.

KMeans Estimators can be configured by setting hyperparameters. The available hyperparameters for KMeans are documented below. For further information on the AWS KMeans algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/k-means.html.

Parameters

- **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 accessing AWS resource.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **k** (*int*) – The number of clusters to produce.
- **init_method** (*str*) – How to initialize cluster locations. One of 'random' or 'kmeans++'.

- **max_iterations** (*int*) – Maximum iterations for Lloyds EM procedure in the local kmeans used in finalize stage.
- **tol** (*float*) – Tolerance for change in ssd for early stopping in local kmeans.
- **num_trials** (*int*) – Local version is run multiple times and the one with the best loss is chosen. This determines how many times.
- **local_init_method** (*str*) – Initialization method for local version. One of 'random', 'kmeans++'
- **half_life_time_size** (*int*) – The points can have a decayed weight. When a point is observed its weight, with regard to the computation of the cluster mean is 1. This weight will decay exponentially as we observe more points. The exponent coefficient is chosen such that after observing `half_life_time_size` points after the mentioned point, its weight will become 1/2. If set to 0, there will be no decay.
- **epochs** (*int*) – Number of passes done over the training data.
- **center_factor** (*int*) – The algorithm will create `num_clusters * extra_center_factor` as it runs and reduce the number of centers to `k` when finalizing
- **eval_metrics** (*list*) – JSON list of metrics types to be used for reporting the score for the model. Allowed values are "msd" Means Square Error, "ssd": Sum of square distance. If test data is provided, the score shall be reported in terms of all requested metrics.
- ****kwargs** – base class keyword argument values.

repo_name = 'kmeans'

repo_version = 1

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()

Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`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.

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

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.

- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

eval_metrics
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)
Return a `KMeansModel` referencing the latest s3 model data produced by this Estimator.

Parameters **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.

hyperparameters()
Return the SageMaker hyperparameters for training this KMeans Estimator

class sagemaker.**KMeansModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)
Bases: *sagemaker.model.Model*

Reference KMeans s3 model data. Calling *deploy()* creates an Endpoint and return a Predictor to performs k-means cluster assignment.

class sagemaker.**KMeansPredictor**(*endpoint*, *sagemaker_session=None*)
Bases: *sagemaker.predictor.RealTimePredictor*

Assigns input vectors to their closest cluster in a KMeans model.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The nearest cluster is stored in the `closest_cluster` key of the `Record.label` field.

9.5 K-Nearest Neighbors

The Amazon SageMaker K-Nearest Neighbors (k-NN) algorithm.

class sagemaker.**KNN**(*role*, *train_instance_count*, *train_instance_type*, *k*, *sample_size*, *predictor_type*, *dimension_reduction_type=None*, *dimension_reduction_target=None*, *index_type=None*, *index_metric=None*, *faiss_index_ivf_nlists=None*, *faiss_index_pq_m=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

k-nearest neighbors (KNN) is `Estimator` used for classification and regression. This Estimator may be fit via calls to *fit()*. It requires Amazon `Record` protobuf serialized data to be stored in S3. There is an utility *record_set()* that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call. To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `KNNPredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint. KNN Estimators can be configured by setting hyperparameters. The available hyperparameters for KNN are documented below. For further information on the AWS KNN algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/knn.html :param role: 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 accessing AWS resource.

Parameters

- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **k** (*int*) – Required. Number of nearest neighbors.
- **sample_size** (*int*) – Required. Number of data points to be sampled from the training data set.
- **predictor_type** (*str*) – Required. Type of inference to use on the data's labels, allowed values are 'classifier' and 'regressor'.
- **dimension_reduction_type** (*str*) – Optional. Type of dimension reduction technique to use. Valid values: "sign", "fjlt"
- **dimension_reduction_target** (*int*) – Optional. Target dimension to reduce to. Required when dimension_reduction_type is specified.
- **index_type** (*str*) – Optional. Type of index to use. Valid values are "faiss.Flat", "faiss.IVFFlat", "faiss.IVFPQ".
- **index_metric** (*str*) – Optional. Distance metric to measure between points when finding nearest neighbors. Valid values are "COSINE", "INNER_PRODUCT", "L2"
- **faiss_index_ivf_nlists** (*str*) – Optional. Number of centroids to construct in the index if index_type is "faiss.IVFFlat" or "faiss.IVFPQ".
- **faiss_index_pq_m** (*int*) – Optional. Number of vector sub-components to construct in the index, if index_type is "faiss.IVFPQ".
- ****kwargs** – base class keyword argument values.

repo_name = 'knn'

repo_version = 1

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()
Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)
Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()
Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)
Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`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.

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

hyperparameters()
Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Return a KNNModel referencing the latest s3 model data produced by this Estimator.

Parameters **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.

class sagemaker.**KNNModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)

Bases: *sagemaker.model.Model*

Reference S3 model data created by KNN estimator. Calling *deploy()* creates an Endpoint and returns *KNNPredictor*.

class sagemaker.**KNNPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Performs classification or regression prediction from input vectors.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy ndarray as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

predict() returns a list of Record objects, one for each row in the input ndarray. The prediction is stored in the "predicted_label" key of the Record.label field.

9.6 LDA

The Amazon SageMaker LDA algorithm.

class sagemaker.**LDA**(*role*, *train_instance_type*, *num_topics*, *alpha0=None*, *max_restarts=None*, *max_iterations=None*, *tol=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

Latent Dirichlet Allocation (LDA) is Estimator used for unsupervised learning.

Amazon SageMaker Latent Dirichlet Allocation is an unsupervised learning algorithm that attempts to describe a set of observations as a mixture of distinct categories. LDA is most commonly used to discover a user-specified number of topics shared by documents within a text corpus. Here each observation is a document, the features are the presence (or occurrence count) of each word, and the categories are the topics.

This Estimator may be fit via calls to *fit()*. It requires Amazon Record protobuf serialized data to be stored in S3. There is an utility *record_set()* that can be used to upload data to S3 and creates RecordSet to be passed to the *fit* call.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking deploy(). As well as deploying an Endpoint, deploy returns a LDAPredictor object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

LDA Estimators can be configured by setting hyperparameters. The available hyperparameters for LDA are documented below.

For further information on the AWS LDA algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/lda.html

Parameters

- **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 accessing AWS resource.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_topics** (*int*) – The number of topics for LDA to find within the data.
- **alpha0** (*float*) – Optional. Initial guess for the concentration parameter
- **max_restarts** (*int*) – Optional. The number of restarts to perform during the Alternating Least Squares (ALS) spectral decomposition phase of the algorithm.
- **max_iterations** (*int*) – Optional. The maximum number of iterations to perform during the ALS phase of the algorithm.
- **tol** (*float*) – Optional. Target error tolerance for the ALS phase of the algorithm.
- ****kwargs** – base class keyword argument values.

repo_name = 'lda'

repo_version = 1

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Return a `LDAModel` referencing the latest s3 model data produced by this Estimator.

Parameters **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.

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()

Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`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.

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

hyperparameters()
Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

class sagemaker.**LDAModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)

Bases: *sagemaker.model.Model*

Reference LDA s3 model data. Calling *deploy()* creates an Endpoint and return a Predictor that transforms vectors to a lower-dimensional representation.

class sagemaker.**LDAPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Transforms input vectors to lower-dimesional representations.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The lower dimension vector result is stored in the `projection` key of the `Record.label` field.

9.7 LinearLearner

The Amazon SageMaker LinearLearner algorithm.

class sagemaker.**LinearLearner**(*role*, *train_instance_count*, *train_instance_type*, *predictor_type*, *binary_classifier_model_selection_criteria=None*, *target_recall=None*, *target_precision=None*, *positive_example_weight_mult=None*, *epochs=None*, *use_bias=None*, *num_models=None*, *num_calibration_samples=None*, *init_method=None*, *init_scale=None*, *init_sigma=None*, *init_bias=None*, *optimizer=None*, *loss=None*, *wd=None*, *l1=None*, *momentum=None*, *learning_rate=None*, *beta_1=None*, *beta_2=None*, *bias_lr_mult=None*, *bias_wd_mult=None*, *use_lr_scheduler=None*, *lr_scheduler_step=None*, *lr_scheduler_factor=None*, *lr_scheduler_minimum_lr=None*, *normalize_data=None*, *normalize_label=None*, *unbias_data=None*, *unbias_label=None*, *num_point_for_scaler=None*, *margin=None*, *quantile=None*, *loss_insensitivity=None*, *huber_delta=None*, *early_stopping_patience=None*, *early_stopping_tolerance=None*, *num_classes=None*, *accuracy_top_k=None*, *f_beta=None*, *balance_multiclass_weights=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

An `Estimator` for binary classification and regression.

Amazon SageMaker Linear Learner provides a solution for both classification and regression problems, allowing for exploring different training objectives simultaneously and choosing the best solution from a validation set. It allows the user to explore a large number of models and choose the best, which optimizes either continuous objectives such as mean square error, cross entropy loss, absolute error, etc., or discrete objectives suited for classification such as F1 measure, precision@recall, accuracy. The implementation provides a significant speedup over naive hyperparameter optimization techniques and an added convenience, when compared with solutions providing a solution only to continuous objectives.

This Estimator may be fit via calls to `fit_ndarray()` or *fit()*. The former allows a LinearLearner model to be fit on a 2-dimensional numpy array. The latter requires Amazon `Record` protobuf serialized data to be stored in S3.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, `deploy` returns a `LinearLearnerPredictor` object that can be used to make class or regression predictions, using the trained model.

LinearLearner Estimators can be configured by setting hyperparameters. The available hyperparameters for LinearLearner are documented below. For further information on the AWS LinearLearner algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/linear-learner.html

Parameters

- **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 accessing AWS resource.

- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **predictor_type** (*str*) – The type of predictor to learn. Either "binary_classifier" or
- **or "regressor".** (*"multiclass_classifier"*) –
- **binary_classifier_model_selection_criteria** (*str*) – One of 'accuracy', 'f1', 'f_beta',
- **'recall_at_target_precision', 'cross_entropy_loss', 'loss_function'** (*'precision_at_target_recall',*) –
- **target_recall** (*float*) – Target recall. Only applicable if binary_classifier_model_selection_criteria is precision_at_target_recall.
- **target_precision** (*float*) – Target precision. Only applicable if binary_classifier_model_selection_criteria is recall_at_target_precision.
- **positive_example_weight_mult** (*float*) – The importance weight of positive examples is multiplied by this constant. Useful for skewed datasets. Only applies for classification tasks.
- **epochs** (*int*) – The maximum number of passes to make over the training data.
- **use_bias** (*bool*) – Whether to include a bias field
- **num_models** (*int*) – Number of models to train in parallel. If not set, the number of parallel models to train will be decided by the algorithm itself. One model will be trained according to the given training
- **parameter** (*regularization, optimizer, loss*) –
- **num_calibration_samples** (*int*) – Number of observations to use from validation dataset for doing model
- **calibration** (*finding the best threshold*) –
- **init_method** (*str*) – Function to use to set the initial model weights. One of "uniform" or "normal"
- **init_scale** (*float*) – For "uniform" init, the range of values.
- **init_sigma** (*float*) – For "normal" init, the standard-deviation.
- **init_bias** (*float*) – Initial weight for bias term
- **optimizer** (*str*) – One of 'sgd', 'adam', 'rmsprop' or 'auto'
- **loss** (*str*) – One of 'logistic', 'squared_loss', 'absolute_loss', 'hinge_loss',
- **'eps_insensitive_absolute_loss', 'quantile_loss', 'huber_loss' or** (*'eps_insensitive_squared_loss',*) –
- **or 'auto'.** (*'softmax_loss'*) –
- **wd** (*float*) – L2 regularization parameter i.e. the weight decay parameter. Use 0 for no L2 regularization.
- **l1** (*float*) – L1 regularization parameter. Use 0 for no L1 regularization.
- **momentum** (*float*) – Momentum parameter of sgd optimizer.

- **learning_rate** (*float*) – The SGD learning rate
- **beta_1** (*float*) – Exponential decay rate for first moment estimates. Only applies for adam optimizer.
- **beta_2** (*float*) – Exponential decay rate for second moment estimates. Only applies for adam optimizer.
- **bias_lr_mult** (*float*) – Allows different learning rate for the bias term. The actual learning rate for the
- **is learning rate times bias_lr_mult.** (*bias*) –
- **bias_wd_mult** (*float*) – Allows different regularization for the bias term. The actual L2 regularization weight
- **the bias is wd times bias_wd_mult. By default there is no regularization on the bias term.** (*for*) –
- **use_lr_scheduler** (*bool*) – If true, we use a scheduler for the learning rate.
- **lr_scheduler_step** (*int*) – The number of steps between decreases of the learning rate. Only applies to learning rate scheduler.
- **lr_scheduler_factor** (*float*) – Every lr_scheduler_step the learning rate will decrease by this quantity. Only applies for learning rate scheduler.
- **lr_scheduler_minimum_lr** (*float*) – The learning rate will never decrease to a value lower than this.
- **lr_scheduler_minimum_lr** – Only applies for learning rate scheduler.
- **normalize_data** (*bool*) – Normalizes the features before training to have standard deviation of 1.0.
- **normalize_label** (*bool*) – Normalizes the regression label to have a standard deviation of 1.0. If set for classification, it will be ignored.
- **unbias_data** (*bool*) – If true, features are modified to have mean 0.0.
- **ubias_label** (*bool*) – If true, labels are modified to have mean 0.0.
- **num_point_for_scaler** (*int*) – The number of data points to use for calculating the normalizing and unbiasing terms.
- **margin** (*float*) – the margin for hinge_loss.
- **quantile** (*float*) – Quantile for quantile loss. For quantile q, the model will attempt to produce
- **such that true_label < prediction with probability q.** (*predictions*) –
- **loss_insensitivity** (*float*) – Parameter for epsilon insensitive loss type. During training and metric
- **any error smaller than this is considered to be zero.** (*evaluation,*) –
- **huber_delta** (*float*) – Parameter for Huber loss. During training and metric evaluation, compute L2 loss for
- **smaller than delta and L1 loss for errors larger than delta.** (*errors*) –

- **early_stopping_patience** (*int*) – the number of epochs to wait before ending training if no improvement is
- **The improvement is training loss if validation data is not provided, or else it is the validation** (*made.*) –
- **or the binary classification model selection criteria like accuracy, f1-score etc. To disable early** (*loss*) –
- **set early_stopping_patience to a value larger than epochs.** (*stopping,*) –
- **early_stopping_tolerance** (*float*) – Relative tolerance to measure an improvement in loss. If the ratio of
- **improvement in loss divided by the previous best loss is smaller than this value, early stopping will** (*the*) –
- **the improvement to be zero.** (*consider*) –
- **num_classes** (*int*) – The number of classes for the response variable. Required when predictor_type is
- **and ignored otherwise. The classes are assumed to be labeled 0, .., num_classes - 1.** (*multiclass_classifier*) –
- **accuracy_top_k** (*int*) – The value of k when computing the Top K Accuracy metric for multiclass
- **An example is scored as correct if the model assigns one of the top k scores to the true** (*classification.*) –
- **label.** –
- **f_beta** (*float*) – The value of beta to use when calculating F score metrics for binary or multiclass
- **Also used if binary_classifier_model_selection_criteria is f_beta.** (*classification.*) –
- **balance_multiclass_weights** (*bool*) – Whether to use class weights which give each class equal importance in
- **loss function. Only used when predictor_type is multiclass_classifier.** (*the*) –
- ****kwargs** – base class keyword argument values.

repo_name = 'linear-learner'

repo_version = 1

DEFAULT_MINI_BATCH_SIZE = 1000

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()
Delete an Amazon SageMaker Endpoint.

Raises ValueError – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)
Deploy the trained model to an Amazon SageMaker endpoint and return a sagemaker.RealTimePredictor object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()
Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)
Fit this Estimator on serialized Record objects, stored in S3.

records should be an instance of 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 record_set() to construct a RecordSet object from ndarray arrays.

Parameters

- **records** (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.

get_vpc_config(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Returns VpcConfig dict either from this Estimator's subnets and security groups, or else validate and return an optional override value.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

model_data

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)

Build a RecordSet from a numpy ndarray matrix and label vector.

For the 2D ndarray train, each row is converted to a 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 train_instance_count property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()

Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

normalize_data
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

normalize_label
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

unbias_data
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

unbias_label
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

num_point_for_scaler
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

margin
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

quantile
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

loss_insensitivity
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

huber_delta
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

early_stopping_patience
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

early_stopping_tolerance
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

num_classes
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

accuracy_top_k
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

f_beta
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

balance_multiclass_weights
An algorithm hyperparameter with optional validation. Implemented as a python descriptor object.

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)
Return a `LinearLearnerModel` referencing the latest s3 model data produced by this Estimator.

Parameters **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.

class sagemaker.**LinearLearnerModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)
Bases: *sagemaker.model.Model*

Reference LinearLearner s3 model data. Calling *deploy()* creates an Endpoint and returns a *LinearLearnerPredictor*

class sagemaker.**LinearLearnerPredictor**(*endpoint*, *sagemaker_session=None*)
Bases: *sagemaker.predictor.RealTimePredictor*

Performs binary-classification or regression prediction from input vectors.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The prediction is stored in the `"predicted_label"` key of the `Record.label` field.

9.8 NTM

The Amazon SageMaker NTM algorithm.

class sagemaker.**NTM**(*role*, *train_instance_count*, *train_instance_type*, *num_topics*, *encoder_layers=None*, *epochs=None*, *encoder_layers_activation=None*, *optimizer=None*, *tolerance=None*, *num_patience_epochs=None*, *batch_norm=None*, *rescale_gradient=None*, *clip_gradient=None*, *weight_decay=None*, *learning_rate=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

Neural Topic Model (NTM) is `Estimator` used for unsupervised learning.

This Estimator may be fit via calls to *fit()*. It requires Amazon `Record` protobuf serialized data to be stored in S3. There is an utility *record_set()* that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `NTMPredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

NTM Estimators can be configured by setting hyperparameters. The available hyperparameters for NTM are documented below.

For further information on the AWS NTM algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/ntm.html

Parameters

- **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 accessing AWS resource.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_topics** (*int*) – Required. The number of topics for NTM to find within the data.
- **encoder_layers** (*list*) – Optional. Represents number of layers in the encoder and the output size of each layer.
- **epochs** (*int*) – Optional. Maximum number of passes over the training data.
- **encoder_layers_activation** (*str*) – Optional. Activation function to use in the encoder layers.
- **optimizer** (*str*) – Optional. Optimizer to use for training.
- **tolerance** (*float*) – Optional. Maximum relative change in the loss function within the last num_patience_epochs number of epochs below which early stopping is triggered.
- **num_patience_epochs** (*int*) – Optional. Number of successive epochs over which early stopping criterion is evaluated.
- **batch_norm** (*bool*) – Optional. Whether to use batch normalization during training.
- **rescale_gradient** (*float*) – Optional. Rescale factor for gradient.

- **clip_gradient** (*float*) – Optional. Maximum magnitude for each gradient component.
- **weight_decay** (*float*) – Optional. Weight decay coefficient. Adds L2 regularization.
- **learning_rate** (*float*) – Optional. Learning rate for the optimizer.
- ****kwargs** – base class keyword argument values.

repo_name = 'ntm'

repo_version = 1

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b

- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

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

Returns Whether this Estimator needs network isolation or not.

Return type bool

`fit`(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)
Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **`records`** (`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.

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

`hyperparameters`()
Return the hyperparameters as a dictionary to use for training.

The *`fit()`* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

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

Type str

`record_set`(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()

Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

training_job_analytics

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)

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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Return a `NTMModel` referencing the latest s3 model data produced by this Estimator.

Parameters **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.

class sagemaker.**NTMModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)

Bases: *sagemaker.model.Model*

Reference NTM s3 model data. Calling *deploy()* creates an Endpoint and return a Predictor that transforms vectors to a lower-dimensional representation.

class sagemaker.**NTMPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Transforms input vectors to lower-dimesional representations.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The lower dimension vector result is stored in the `projection` key of the `Record.label` field.

9.9 Object2Vec

The Amazon SageMaker Object2Vec algorithm.

class sagemaker.**Object2Vec**(*role*, *train_instance_count*, *train_instance_type*, *epochs*, *enc0_max_seq_len*, *enc0_vocab_size*, *enc_dim=None*, *mini_batch_size=None*, *early_stopping_patience=None*, *early_stopping_tolerance=None*, *dropout=None*, *weight_decay=None*, *bucket_width=None*, *num_classes=None*, *mlp_layers=None*, *mlp_dim=None*, *mlp_activation=None*, *output_layer=None*, *optimizer=None*, *learning_rate=None*, *enc0_network=None*, *enc1_network=None*, *enc0_cnn_filter_width=None*, *enc1_cnn_filter_width=None*, *enc1_max_seq_len=None*, *enc0_token_embedding_dim=None*, *enc1_token_embedding_dim=None*, *enc1_vocab_size=None*, *enc0_layers=None*, *enc1_layers=None*, *enc0_freeze_pretrained_embedding=None*, *enc1_freeze_pretrained_embedding=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

Object2Vec is `Estimator` used for anomaly detection.

This Estimator may be fit via calls to *`fit()`*. There is an utility *`record_set()`* that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call.

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `RealTimePredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

Object2Vec Estimators can be configured by setting hyperparameters. The available hyperparameters for Object2Vec are documented below.

For further information on the AWS Object2Vec algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/object2vec.html

Parameters

- **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 accessing AWS resource.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **epochs** (*int*) – Total number of epochs for SGD training
- **enc0_max_seq_len** (*int*) – Maximum sequence length
- **enc0_vocab_size** (*int*) – Vocabulary size of tokens
- **enc_dim** (*int*) – Optional. Dimension of the output of the embedding layer
- **mini_batch_size** (*int*) – Optional. mini batch size for SGD training
- **early_stopping_patience** (*int*) – Optional. The allowed number of consecutive epochs without improvement before early stopping is applied
- **early_stopping_tolerance** (*float*) – Optional. The value used to determine whether the algorithm has made improvement between two consecutive epochs for early stopping
- **dropout** (*float*) – Optional. Dropout probability on network layers
- **weight_decay** (*float*) – Optional. Weight decay parameter during optimization
- **bucket_width** (*int*) – Optional. The allowed difference between data sequence length when bucketing is enabled
- **num_classes** (*int*) – Optional. Number of classes for classification training (ignored for regression problems)
- **mlp_layers** (*int*) – Optional. Number of MLP layers in the network
- **mlp_dim** (*int*) – Optional. Dimension of the output of MLP layer
- **mlp_activation** (*str*) – Optional. Type of activation function for the MLP layer
- **output_layer** (*str*) – Optional. Type of output layer
- **optimizer** (*str*) – Optional. Type of optimizer for training
- **learning_rate** (*float*) – Optional. Learning rate for SGD training

- **enc0_network** (*str*) – Optional. Network model of encoder "enc0"
- **enc1_network** (*str*) – Optional. Network model of encoder "enc1"
- **enc0_cnn_filter_width** (*int*) – Optional. CNN filter width
- **enc1_cnn_filter_width** (*int*) – Optional. CNN filter width
- **enc1_max_seq_len** (*int*) – Optional. Maximum sequence length
- **enc0_token_embedding_dim** (*int*) – Optional. Output dimension of token embedding layer
- **enc1_token_embedding_dim** (*int*) – Optional. Output dimension of token embedding layer
- **enc1_vocab_size** (*int*) – Optional. Vocabulary size of tokens
- **enc0_layers** (*int*) – Optional. Number of layers in encoder
- **enc1_layers** (*int*) – Optional. Number of layers in encoder
- **enc0_freeze_pretrained_embedding** (*bool*) – Optional. Freeze pretrained embedding weights
- **enc1_freeze_pretrained_embedding** (*bool*) – Optional. Freeze pretrained embedding weights
- ****kwargs** – base class keyword argument values.

repo_name = 'object2vec'

repo_version = 1

MINI_BATCH_SIZE = 32

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()

Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`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.

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

hyperparameters()
Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Return a `Object2VecModel` referencing the latest s3 model data produced by this Estimator.

Parameters **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.

class sagemaker.**Object2VecModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)

Bases: *sagemaker.model.Model*

Reference Object2Vec s3 model data. Calling *deploy()* creates an Endpoint and returns a Predictor that calculates anomaly scores for datapoints.

9.10 PCA

The Amazon SageMaker PCA algorithm.

class sagemaker.**PCA**(*role*, *train_instance_count*, *train_instance_type*, *num_components*, *algorithm_mode=None*, *subtract_mean=None*, *extra_components=None*, ***kwargs*)

Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

A Principal Components Analysis (PCA) *AmazonAlgorithmEstimatorBase*.

This Estimator may be fit via calls to `fit_ndarray()` or *fit()*. The former allows a PCA model to be fit on a 2-dimensional numpy array. The latter requires Amazon `Record` protobuf serialized data to be stored in S3.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `PCAPredictor` object that can be used to project input vectors to the learned lower-dimensional representation, using the trained PCA model hosted in the SageMaker Endpoint.

PCA Estimators can be configured by setting hyperparameters. The available hyperparameters for PCA are documented below. For further information on the AWS PCA algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/pca.html

This Estimator uses Amazon SageMaker PCA to perform training and host deployed models. To learn more about Amazon SageMaker PCA, please read: https://docs.aws.amazon.com/sagemaker/latest/dg/how-pca-works.html

Parameters

- **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 accessing AWS resource.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.
- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_components** (*int*) – The number of principal components. Must be greater than zero.
- **algorithm_mode** (*str*) – Mode for computing the principal components. One of 'regular' or 'randomized'.
- **subtract_mean** (*bool*) – Whether the data should be unbiased both during train and at inference.
- **extra_components** (*int*) – As the value grows larger, the solution becomes more accurate but the runtime and memory consumption increase linearly. If this value is unset or set to -1, then a default value equal to the maximum of 10 and num_components will be used. Valid for randomized mode only.
- ****kwargs** – base class keyword argument values.

repo_name = 'pca'

repo_version = 1

DEFAULT_MINI_BATCH_SIZE = 500

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Return a `PCAModel` referencing the latest s3 model data produced by this Estimator.

Parameters **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.

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()
Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)
Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html
- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()
Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)
Fit this Estimator on serialized Record objects, stored in S3.

records should be an instance of 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 record_set() to construct a RecordSet object from ndarray arrays.

Parameters

- **records** (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.

get_vpc_config(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Returns VpcConfig dict either from this Estimator's subnets and security groups, or else validate and return an optional override value.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

model_data

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)

Build a RecordSet from a numpy ndarray matrix and label vector.

For the 2D ndarray train, each row is converted to a 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 train_instance_count property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

class sagemaker.**PCAModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)
Bases: *sagemaker.model.Model*

Reference PCA s3 model data. Calling *deploy()* creates an Endpoint and return a Predictor that transforms vectors to a lower-dimensional representation.

class sagemaker.**PCAPredictor**(*endpoint*, *sagemaker_session=None*)
Bases: *sagemaker.predictor.RealTimePredictor*

Transforms input vectors to lower-dimesional representations.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input `ndarray`. The lower dimension vector result is stored in the `projection` key of the `Record.label` field.

9.11 Random Cut Forest

The Amazon SageMaker Random Cut Forest algorithm.

class sagemaker.**RandomCutForest**(*role*, *train_instance_count*, *train_instance_type*, *num_samples_per_tree=None*, *num_trees=None*, *eval_metrics=None*, ***kwargs*)
Bases: *sagemaker.amazon.amazon_estimator.AmazonAlgorithmEstimatorBase*

RandomCutForest is `Estimator` used for anomaly detection.

This Estimator may be fit via calls to *fit()*. It requires Amazon `Record` protobuf serialized data to be stored in S3. There is an utility *record_set()* that can be used to upload data to S3 and creates `RecordSet` to be passed to the *fit* call.

To learn more about the Amazon protobuf Record class and how to prepare bulk data in this format, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/cdf-training.html

After this Estimator is fit, model data is stored in S3. The model may be deployed to an Amazon SageMaker Endpoint by invoking `deploy()`. As well as deploying an Endpoint, deploy returns a `RandomCutForestPredictor` object that can be used for inference calls using the trained model hosted in the SageMaker Endpoint.

RandomCutForest Estimators can be configured by setting hyperparameters. The available hyperparameters for RandomCutForest are documented below.

For further information on the AWS Random Cut Forest algorithm, please consult AWS technical documentation: https://docs.aws.amazon.com/sagemaker/latest/dg/randomcutforest.html

Parameters

- **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 accessing AWS resource.
- **train_instance_count** (*int*) – Number of Amazon EC2 instances to use for training.

- **train_instance_type** (*str*) – Type of EC2 instance to use for training, for example, 'ml.c4.xlarge'.
- **num_samples_per_tree** (*int*) – Optional. The number of samples used to build each tree in the forest. The total number of samples drawn from the train dataset is num_trees * num_samples_per_tree.
- **num_trees** (*int*) – Optional. The number of trees used in the forest.
- **eval_metrics** (*list*) – Optional. JSON list of metrics types to be used for reporting the score for the model. Allowed values are "accuracy", "precision_recall_fscore": positive and negative precision, recall, and f1 scores. If test data is provided, the score shall be reported in terms of all requested metrics.
- ****kwargs** – base class keyword argument values.

repo_name = 'randomcutforest'

repo_version = 1

create_model(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Return a `RandomCutForestModel` referencing the latest s3 model data produced by this Estimator.

Parameters **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.

classmethod attach(*training_job_name*, *sagemaker_session=None*, *model_channel_name='model'*)

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.

Parameters

- **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.

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()
```

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

compile_model(*target_instance_family*, *input_shape*, *output_path*, *framework=None*, *framework_version=None*, *compile_max_run=300*, *tags=None*, ***kwargs*)

Compile a Neo model using the input model.

Parameters

- **target_instance_family** (*str*) – Identifies the device that you want to run your model after compilation, for example: ml_c5. Allowed strings are: ml_c5, ml_m5, ml_c4, ml_m4, jetsontx1, jetsontx2, ml_p2, ml_p3, deeplens, rasp3b
- **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', '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/dg/API_Tag.html.
- ****kwargs** – 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 A SageMaker `Model` object. See *Model()* for full details.

Return type *sagemaker.model.Model*

data_location

delete_endpoint()

Delete an Amazon SageMaker `Endpoint`.

Raises `ValueError` – If the endpoint does not exist.

deploy(*initial_instance_count*, *instance_type*, *accelerator_type=None*, *endpoint_name=None*, *use_compiled_model=False*, ***kwargs*)

Deploy the trained model to an Amazon SageMaker endpoint and return a `sagemaker.RealTimePredictor` object.

More information: http://docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

Parameters

- **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/latest/dg/ei.html

- **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.
- ****kwargs** – 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

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

Return type *sagemaker.predictor.RealTimePredictor*

enable_network_isolation()

Return True if this Estimator will need network isolation to run.

Returns Whether this Estimator needs network isolation or not.

Return type bool

fit(*records*, *mini_batch_size=None*, *wait=True*, *logs=True*, *job_name=None*)

Fit this Estimator on serialized Record objects, stored in S3.

`records` should be an instance of `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 `record_set()` to construct a `RecordSet` object from `ndarray` arrays.

Parameters

- **records** (`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.

get_vpc_config(*vpc_config_override='VPC_CONFIG_DEFAULT'*)

Returns VpcConfig dict either from this Estimator's subnets and security groups, or else validate and return an optional override value.

hyperparameters()

Return the hyperparameters as a dictionary to use for training.

The *fit()* method, which trains the model, calls this method to find the hyperparameters.

Returns The hyperparameters.

Return type dict[str, str]

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

Type str

record_set(*train*, *labels=None*, *channel='train'*)
Build a `RecordSet` from a numpy `ndarray` matrix and label vector.

For the 2D `ndarray` `train`, each row is converted to a `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 `train_instance_count` property on this Estimator. One S3 object is created per training instance.

Parameters

- **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.

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

Return type RecordSet

train_image()
Return the Docker image to use for training.

The *fit()* method, which does the model training, calls this method to find the image to use for model training.

Returns The URI of the Docker image.

Return type str

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

transformer(*instance_count*, *instance_type*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*)
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.

Parameters

- **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: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).

class sagemaker.**RandomCutForestModel**(*model_data*, *role*, *sagemaker_session=None*, ***kwargs*)

Bases: *sagemaker.model.Model*

Reference RandomCutForest s3 model data. Calling *deploy()* creates an Endpoint and returns a Predictor that calculates anomaly scores for datapoints.

class sagemaker.**RandomCutForestPredictor**(*endpoint*, *sagemaker_session=None*)

Bases: *sagemaker.predictor.RealTimePredictor*

Assigns an anomaly score to each of the datapoints provided.

The implementation of *predict()* in this *RealTimePredictor* requires a numpy `ndarray` as input. The array should contain the same number of columns as the feature-dimension of the data used to fit the model this Predictor performs inference on.

`predict()` returns a list of `Record` objects, one for each row in the input. Each row's score is stored in the key `score` of the `Record.label` field.

CHAPTER 10

Workflows

SageMaker APIs to export configurations for creating and managing Airflow workflows.

10.1 Airflow

10.1.1 training_config

sagemaker.workflow.airflow.**training_config**(*estimator*, *inputs=None*, *job_name=None*, *mini_batch_size=None*)

Export Airflow training config from an estimator

Parameters

- **estimator** (sagemaker.estimator.EstimatorBase) – The estimator to export training config from. Can be a BYO estimator, Framework estimator or Amazon algorithm estimator.
- **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, sagemaker.session.s3_input]) - If using multiple channels for** training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.
  - **(sagemaker.session.s3_input) - Channel configuration for S3 data sources that can provide** additional information about the training dataset. See *sagemaker.session.s3_input()* for full details.
  - **(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.

- **(list[sagemaker.amazon.amazon_estimator.RecordSet]) - A list of** :class:~‘sagemaker.amazon.amazon_estimator.RecordSet‘ objects, where each instance is a different channel of training data.

- **job_name** (*str*) – Specify a training job name if needed.
- **mini_batch_size** (*int*) – Specify this argument only when estimator is a built-in estimator of an Amazon algorithm. For other estimators, batch size should be specified in the estimator.

Returns Training config that can be directly used by SageMakerTrainingOperator in Airflow.

Return type dict

10.1.2 tuning_config

sagemaker.workflow.airflow.**tuning_config**(*tuner*, *inputs*, *job_name=None*)
Export Airflow tuning config from an estimator

Parameters

- **tuner** (sagemaker.tuner.HyperparameterTuner) – The tuner to export tuning config from.
- **inputs** –

  Information about the training data. Please refer to the fit() method of the associated estimator in the tuner, as this can take any of the following forms:

  - (str) - The S3 location where training data is saved.
  - **(dict[str, str] or dict[str, sagemaker.session.s3_input]) - If using multiple channels for** training data, you can specify a dict mapping channel names to strings or *s3_input()* objects.
  - **(sagemaker.session.s3_input) - Channel configuration for S3 data sources that can provide** additional information about the training dataset. See *sagemaker.session.s3_input()* for full details.
  - **(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.
  - **(list[sagemaker.amazon.amazon_estimator.RecordSet]) - A list of** :class:~‘sagemaker.amazon.amazon_estimator.RecordSet‘ objects, where each instance is a different channel of training data.

- **job_name** (*str*) – Specify a tuning job name if needed.

Returns Tuning config that can be directly used by SageMakerTuningOperator in Airflow.

Return type dict

10.1.3 model_config

sagemaker.workflow.airflow.**model_config**(*instance_type*, *model*, *role=None*, *image=None*)
Export Airflow model config from a SageMaker model

Parameters

- **instance_type** (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'
- **model** (*sagemaker.model.FrameworkModel*) – The SageMaker model to export Airflow config from
- **role** (*str*) – The ExecutionRoleArn IAM Role ARN for the model
- **image** (*str*) – An container image to use for deploying the model

Returns Model config that can be directly used by SageMakerModelOperator in Airflow. It can also be part of the config used by SageMakerEndpointOperator and SageMakerTransformOperator in Airflow.

Return type dict

10.1.4 model_config_from_estimator

sagemaker.workflow.airflow.**model_config_from_estimator**(*instance_type*, *estimator*, *role=None*, *image=None*, *model_server_workers=None*, *vpc_config_override='VPC_CONFIG_DEFAULT'*)

Export Airflow model config from a SageMaker estimator

Parameters

- **instance_type** (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'
- **estimator** (*sagemaker.model.EstimatorBase*) – The SageMaker estimator to export Airflow config from. It has to be an estimator associated with a training job.
- **role** (*str*) – The ExecutionRoleArn IAM Role ARN for the model
- **image** (*str*) – An container image to use for deploying the model
- **model_server_workers** (*int*) – The number of worker processes used by the inference server. If None, server will use one worker per vCPU. Only effective when estimator is a SageMaker framework.
- **vpc_config_override** (*dict[str, list[str]]*) – 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.

Returns Model config that can be directly used by SageMakerModelOperator in Airflow. It can also be part of the config used by SageMakerEndpointOperator and SageMakerTransformOperator in Airflow.

Return type dict

10.1.5 transform_config

sagemaker.workflow.airflow.**transform_config**(*transformer*, *data*, *data_type='S3Prefix'*, *content_type=None*, *compression_type=None*, *split_type=None*, *job_name=None*)

Export Airflow transform config from a SageMaker transformer

Parameters

- **transformer** (sagemaker.transformer.Transformer) – The SageMaker transformer to export Airflow config from.
- **data** (*str*) – Input data location in S3.
- **data_type** (*str*) – What the S3 location defines (default: 'S3Prefix'). Valid values:
  - **'S3Prefix' - the S3 URI defines a key name prefix. All objects with this prefix will be used as** inputs for the transform job.
  - **'ManifestFile' - the S3 URI points to a single manifest file listing each S3 object to use as** an input for the transform job.
- **content_type** (*str*) – MIME type of the input data (default: None).
- **compression_type** (*str*) – Compression type of the input data, if compressed (default: None). Valid values: 'Gzip', None.
- **split_type** (*str*) – The record delimiter for the input object (default: 'None'). Valid values: 'None', 'Line', 'RecordIO', and 'TFRecord'.
- **job_name** (*str*) – job name (default: None). If not specified, one will be generated.

Returns Transform config that can be directly used by SageMakerTransformOperator in Airflow.

Return type dict

10.1.6 transform_config_from_estimator

sagemaker.workflow.airflow.**transform_config_from_estimator**(*estimator*, *instance_count*, *instance_type*, *data*, *data_type='S3Prefix'*, *content_type=None*, *compression_type=None*, *split_type=None*, *job_name=None*, *strategy=None*, *assemble_with=None*, *output_path=None*, *output_kms_key=None*, *accept=None*, *env=None*, *max_concurrent_transforms=None*, *max_payload=None*, *tags=None*, *role=None*, *volume_kms_key=None*, *model_server_workers=None*, *image=None*, *vpc_config_override=None*)

Export Airflow transform config from a SageMaker estimator

Parameters

- **estimator** (*sagemaker.model.EstimatorBase*) – The SageMaker estimator to export Airflow config from. It has to be an estimator associated with a training job.

- **instance_count** (*int*) – Number of EC2 instances to use.
- **instance_type** (*str*) – Type of EC2 instance to use, for example, 'ml.c4.xlarge'.
- **data** (*str*) – Input data location in S3.
- **data_type** (*str*) – What the S3 location defines (default: 'S3Prefix'). Valid values:
  - **'S3Prefix' - the S3 URI defines a key name prefix. All objects with this prefix will be used as** inputs for the transform job.
  - **'ManifestFile' - the S3 URI points to a single manifest file listing each S3 object to use as** an input for the transform job.
- **content_type** (*str*) – MIME type of the input data (default: None).
- **compression_type** (*str*) – Compression type of the input data, if compressed (default: None). Valid values: 'Gzip', None.
- **split_type** (*str*) – The record delimiter for the input object (default: 'None'). Valid values: 'None', 'Line', 'RecordIO', and 'TFRecord'.
- **job_name** (*str*) – job name (default: None). If not specified, one will be generated.
- **strategy** (*str*) – The strategy used to decide how to batch records in a single request (default: None). Valid values: 'MULTI_RECORD' and 'SINGLE_RECORD'.
- **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 content type accepted by the endpoint deployed during the transform job.
- **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).
- **model_server_workers** (*int*) – Optional. The number of worker processes used by the inference server. If None, server will use one worker per vCPU.
- **image** (*str*) – An container image to use for deploying the model

- **vpc_config_override** (*dict[str, list[str]]*) – 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.

Returns Transform config that can be directly used by SageMakerTransformOperator in Airflow.

Return type dict

10.1.7 deploy_config

sagemaker.workflow.airflow.**deploy_config**(*model*, *initial_instance_count*, *instance_type*, *endpoint_name=None*, *tags=None*)

Export Airflow deploy config from a SageMaker model

Parameters

- **model** (*sagemaker.model.Model*) – The SageMaker model to export the Airflow config from.
- **instance_type** (*str*) – The EC2 instance type to deploy this Model to. For example, 'ml.p2.xlarge'.
- **initial_instance_count** (*int*) – The initial number of instances to run in the Endpoint created from this Model.
- **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]*) – List of tags for labeling a training job. For more, see https://docs.aws.amazon.com/sagemaker/latest/dg/API_Tag.html.

Returns Deploy config that can be directly used by SageMakerEndpointOperator in Airflow.

Return type dict

10.1.8 deploy_config_from_estimator

sagemaker.workflow.airflow.**deploy_config_from_estimator**(*estimator*, *initial_instance_count*, *instance_type*, *endpoint_name=None*, *tags=None*, ***kwargs*)

Export Airflow deploy config from a SageMaker estimator

Parameters

- **estimator** (*sagemaker.model.EstimatorBase*) – The SageMaker estimator to export Airflow config from. It has to be an estimator associated with a training job.
- **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'.
- **endpoint_name** (*str*) – Name to use for creating an Amazon SageMaker endpoint. If not specified, the name of the training job is used.
- **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.

- ****kwargs** – 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 Deploy config that can be directly used by SageMakerEndpointOperator in Airflow.

Return type dict

Python Module Index

S

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W