Build a SageMaker Pipeline to train and deploy a BERT-Based text classifier

Introduction

In this lab, you will do the following:

- Define and run a pipeline using a directed acyclic graph (DAG) with specific pipeline parameters and model hyper-parameters
- Define a processing step that cleans, balances, transforms, and splits our dataset into train, validation, and test dataset
- Define a training step that trains a model using the train and validation datasets
- Define a processing step that evaluates the trained model's performance on the test dataset
- Define a register model step that creates a model package from the trained model
- Define a conditional step that checks the model's performance and conditionally registers the model for deployment

Table of Contents

- 1. Configure the dataset and processing step
  - 1.1. Configure S3 path for raw input data
  - 1.2. Configure processing step
    - Exercise 1
    - Exercise 2
- 2. Configure training step
  - 2.1. Define parameters
  - 2.2. Configure hyper-parameters
  - 2.3. Configure model-evaluation metrics
  - 2.4. Configure the `PyTorchEstimator`
  - 2.5. Setup pipeline step caching
  - 2.6. Configure the `TrainingStep`
    - Exercise 3
- 3. Configure model-evaluation step
- 4. Configure and register model step
  - 4.1. Configure the model for deployment
  - 4.2. Register the model for deployment
    - Exercise 4
- 5. Create model for deployment step
  - Exercise 5
  - Exercise 6

- 6. Check accuracy condition step
- 7. Create pipeline
  - 7.1. Define a pipeline of parameters, steps, and conditions
  - 7.2. Start Pipeline
  - 7.3. Wait for pipeline execution
  - 7.4. Describe completed pipeline
  - 7.5. Wait for the pipeline to complete
- 8. Evaluate the model
  - 8.1. Describe evaluation metrics
    - Exercise 7
  - 8.2. Review the evaluation report
  - 8.3. List pipeline artifacts
    - Exercise 8
- 9. Deploy and test the model
  - 9.1. Approve trained model
  - 9.2. Deploy model
  - 9.3. Create endpoint from registry
  - 9.4. Test model
  - 9.5. SageMaker Studio extensions

Terminology

This notebook focuses on the following features of Amazon SageMaker Pipelines:

- **Pipelines** - a directed acyclic graph (DAG) of steps and conditions to orchestrate SageMaker jobs and resource creation
- **Processing job steps** - a simplified, managed experience on SageMaker to run data processing workloads, such as feature engineering, data validation, model evaluation, and model explainability
- **Training job steps** - an iterative process that teaches a model to make predictions on new data by presenting examples from a training dataset
- **Conditional step execution** - provides conditional execution of branches in a pipeline
- **Registering models** - register a model in a model registry to create a deployable models in Amazon SageMaker
- **Parameterized pipeline executions** - allows pipeline executions to vary by supplied parameters
- **Model endpoint** - hosts the model as a REST endpoint to serve predictions from new data

BERT Pipeline

The pipeline that you will create follows a typical machine learning application pattern of pre-processing, training, evaluation, and model registration.

In the processing step, you will perform feature engineering to transform the `review_body` text into BERT embeddings using the pre-trained BERT model and split the dataset into train, validation and test files. The transformed dataset is stored in a feature store. To optimize for Tensorflow training, the transformed dataset files are saved using the TFRecord format in Amazon S3.

In the training step, you will fine-tune the BERT model to the customer reviews dataset and add a new classification layer to predict the `sentiment` for a given `review_body`.

In the evaluation step, you will take the trained model and a test dataset as input, and produce a JSON file containing classification evaluation metrics.

In the condition step, you will register the trained model if the accuracy of the model, as determined by our evaluation step, exceeds a given threshold value.

First, install the required modules.

In [1]:
```
# please ignore warning messages during the installation
!pip install --disable-pip-version-check -q sagemaker==2.35.0
```

```
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommen
ded to use a virtual environment instead: https://pip.pypa.io/warnings/ve
nv
```

In [2]:
```
import os
import sagemaker
import logging
import boto3
import sagemaker
import pandas as pd
import json
import botocore
from botocore.exceptions import ClientError

config = botocore.config.Config(user_agent_extra='dlai-pds/c2/w3')

# low-level service client of the boto3 session
sm = boto3.client(service_name='sagemaker',
                  config=config)

sm_runtime = boto3.client('sagemaker-runtime',
                          config=config)

sess = sagemaker.Session(sagemaker_client=sm,
                         sagemaker_runtime_client=sm_runtime)

bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = sess.boto_region_name
```

Setup the pipeline name.

In [3]:
```
import time
timestamp = int(time.time())

pipeline_name = 'BERT-pipeline-{}'.format(timestamp)
```

1. Configure the dataset and processing step

1.1. Configure S3 path for raw input data

The raw dataset is in the public S3 bucket. Let's start by specifying the S3 location of it:

In [4]:
```
raw_input_data_s3_uri = 's3://dlai-practical-data-science/data/raw/'
print(raw_input_data_s3_uri)
```

```
s3://dlai-practical-data-science/data/raw/
```

List the files in the S3 bucket (in this case it will be just one file):

In [5]:
```
!aws s3 ls $raw_input_data_s3_uri
```

```
2021-04-30 02:21:06    8457214 womens_clothing_ecommerce_reviews.csv
```

1.2. Configure processing step

For the pipeline workflow you will need to create workflow parameters of a specific type: integer, string, or float.

```
In [6]: from sagemaker.workflow.parameters import (
            ParameterInteger,
            ParameterString,
            ParameterFloat,
        )
```

Now set the parameters for the processing step.

In [7]:
```
processing_instance_type = ParameterString(
    name="ProcessingInstanceType",
    default_value="ml.c5.2xlarge"
)

processing_instance_count = ParameterInteger(
    name="ProcessingInstanceCount",
    default_value=1
)

train_split_percentage = ParameterFloat(
    name="TrainSplitPercentage",
    default_value=0.90,
)

validation_split_percentage = ParameterFloat(
    name="ValidationSplitPercentage",
    default_value=0.05,
)

test_split_percentage = ParameterFloat(
    name="TestSplitPercentage",
    default_value=0.05,
)

balance_dataset = ParameterString(
    name="BalanceDataset",
    default_value="True",
)

max_seq_length = ParameterInteger(
    name="MaxSeqLength",
    default_value=128,
)

feature_store_offline_prefix = ParameterString(
    name="FeatureStoreOfflinePrefix",
    default_value="reviews-feature-store-" + str(timestamp),
)

feature_group_name = ParameterString(
    name="FeatureGroupName",
    default_value="reviews-feature-group-" + str(timestamp)
)

input_data = ParameterString(
    name="InputData",
    default_value=raw_input_data_s3_uri,
)
```

Setting up scikit-learn-based processor, pass the SageMaker execution role, processing instance type and instance count.

In [8]:
```
from sagemaker.sklearn.processing import SKLearnProcessor

processor = SKLearnProcessor(
    framework_version='0.23-1',
    role=role,
    instance_type=processing_instance_type,
    instance_count=processing_instance_count,
    env={'AWS_DEFAULT_REGION': region},
)
```

Now you will use the processor instance to construct a `ProcessingStep`, along with the input and output channels and the code that will be executed when the pipeline invokes pipeline execution. This is very similar to a processor instance's `run` method, for those familiar with the existing Python SDK.

Note the `"sentiment-train"`, `"sentiment-validation"` and `"sentiment-test"` named channels specified in the output configuration for the processing job. Such step `Properties` can be used in subsequent steps and will resolve to their runtime values at execution. In particular, you will call out this usage defining the training step.

In [9]:
```
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker.workflow.steps import ProcessingStep

processing_inputs=[
    ProcessingInput(
        input_name='raw-input-data',
        source=input_data,
        destination='/opt/ml/processing/input/data/',
        s3_data_distribution_type='ShardedByS3Key'
    )
]

processing_outputs=[
    ProcessingOutput(output_name='sentiment-train',
                     source='/opt/ml/processing/output/sentiment/train',
                     s3_upload_mode='EndOfJob'),
    ProcessingOutput(output_name='sentiment-validation',
                     source='/opt/ml/processing/output/sentiment/validati
                     s3_upload_mode='EndOfJob'),
    ProcessingOutput(output_name='sentiment-test',
                     source='/opt/ml/processing/output/sentiment/test',
                     s3_upload_mode='EndOfJob')
]

processing_step = ProcessingStep(
    name='Processing',
    code='src/prepare_data.py',
    processor=processor,
    inputs=processing_inputs,
    outputs=processing_outputs,
    job_arguments=['--train-split-percentage', str(train_split_percentage
                   '--validation-split-percentage', str(validation_split_
                   '--test-split-percentage', str(test_split_percentage.d
                   '--balance-dataset', str(balance_dataset.default_value
                   '--max-seq-length', str(max_seq_length.default_value),
                   '--feature-store-offline-prefix', str(feature_store_of
                   '--feature-group-name', str(feature_group_name.default
                  ]
)

print(processing_step)
```

```
ProcessingStep(name='Processing', step_type=<StepTypeEnum.PROCESSING: 'Pr
ocessing'>)
```

Now you can call out the properties of the processing job as an object using the command `processing_step.properties`. To print out and explore the attributes use `__dict__` method.

In [10]:
```
# print out the list of the processing job properties
print(json.dumps(
    processing_step.properties.__dict__,
    indent=4, sort_keys=True, default=str
))
```

```
{
    "AppSpecification": "<sagemaker.workflow.properties.Properties object
at 0x7f52abbc7390>",
    "AutoMLJobArn": "<sagemaker.workflow.properties.Properties object at
0x7f52abc83f10>",
    "CreationTime": "<sagemaker.workflow.properties.Properties object at
0x7f52abc83990>",
    "Environment": "<sagemaker.workflow.properties.Properties object at 0
x7f52abbc7490>",
    "ExitMessage": "<sagemaker.workflow.properties.Properties object at 0
x7f52abbc7810>",
    "ExperimentConfig": "<sagemaker.workflow.properties.Properties object
at 0x7f52abbc7690>",
    "FailureReason": "<sagemaker.workflow.properties.Properties object at
0x7f52abbc7850>",
    "LastModifiedTime": "<sagemaker.workflow.properties.Properties object
at 0x7f52abc83c50>",
    "MonitoringScheduleArn": "<sagemaker.workflow.properties.Properties o
bject at 0x7f52abc83090>",
    "NetworkConfig": "<sagemaker.workflow.properties.Properties object at
0x7f52abbc74d0>",
    "ProcessingEndTime": "<sagemaker.workflow.properties.Properties objec
t at 0x7f52abbc7890>",
    "ProcessingInputs": "<sagemaker.workflow.properties.PropertiesList ob
ject at 0x7f52abbc7050>",
    "ProcessingJobArn": "<sagemaker.workflow.properties.Properties object
at 0x7f52abbc7790>",
    "ProcessingJobName": "<sagemaker.workflow.properties.Properties objec
t at 0x7f52abbc7150>",
    "ProcessingJobStatus": "<sagemaker.workflow.properties.Properties obj
ect at 0x7f52abbc77d0>",
    "ProcessingOutputConfig": "<sagemaker.workflow.properties.Properties
object at 0x7f52abbc7090>",
    "ProcessingResources": "<sagemaker.workflow.properties.Properties obj
ect at 0x7f52abbc7190>",
    "ProcessingStartTime": "<sagemaker.workflow.properties.Properties obj
ect at 0x7f52abbc78d0>",
    "RoleArn": "<sagemaker.workflow.properties.Properties object at 0x7f5
2abbc7650>",
    "StoppingCondition": "<sagemaker.workflow.properties.Properties objec
t at 0x7f52abbc7310>",
    "TrainingJobArn": "<sagemaker.workflow.properties.Properties object a
t 0x7f52abc83f50>",
    "_path": "Steps.Processing",
    "_shape_name": "DescribeProcessingJobResponse"
}
```

Pull the channel `sentiment-train` from the output configuration of the processing job. Print out the attributes of the resulting object:

In [11]:
```
print(json.dumps(
    processing_step.properties.ProcessingOutputConfig.Outputs['sentiment-
    indent=4, sort_keys=True, default=str
))
```

```
{
    "AppManaged": "<sagemaker.workflow.properties.Properties object at 0x
7f52abc83550>",
    "FeatureStoreOutput": "<sagemaker.workflow.properties.Properties obje
ct at 0x7f52abc83210>",
    "OutputName": "<sagemaker.workflow.properties.Properties object at 0x
7f52abc83d10>",
    "S3Output": "<sagemaker.workflow.properties.Properties object at 0x7f
52abc83e10>",
    "_path": "Steps.Processing.ProcessingOutputConfig.Outputs['sentiment-
train']",
    "_shape_name": "ProcessingOutput"
}
```

Now you can pull and print out attributes of the S3 output path related to the `sentiment-train` output channel:

In [12]:
```
print(json.dumps(
    processing_step.properties.ProcessingOutputConfig.Outputs['sentiment-
    indent=4, sort_keys=True, default=str
))
```

```
{
    "__str__": "S3Uri",
    "_path": "Steps.Processing.ProcessingOutputConfig.Outputs['sentiment-
train'].S3Output.S3Uri",
    "_shape_name": "S3Uri"
}
```

Exercise 1

Pull and print out attributes of the S3 output path object related to the `sentiment-test` output channel.

Instructions: Use the example in the cell above.

In [13]:
```
print(json.dumps(
    ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
    processing_step.properties.ProcessingOutputConfig.Outputs['sentiment-
    ### END SOLUTION - DO NOT delete this comment for grading purposes
    indent=4, sort_keys=True, default=str
))
```

```
{
    "__str__": "S3Uri",
    "_path": "Steps.Processing.ProcessingOutputConfig.Outputs['sentiment-
test'].S3Output.S3Uri",
    "_shape_name": "S3Uri"
}
```

These objects can be passed into the next steps of the workflow. Also, you can pull the arguments of the processing step with the corresponding function. The result is in the dictionary format. Review the keys of this dictionary:

In [14]:
```
processing_step.arguments.keys()
```

```
Out[14]: dict_keys(['ProcessingResources', 'AppSpecification', 'RoleArn', 'Process
ingInputs', 'ProcessingOutputConfig', 'Environment'])
```

Pull and review processing inputs from the arguments of the processing step:

```
In [15]: processing_step.arguments['ProcessingInputs']
```

```
Out[15]: [{'InputName': 'raw-input-data',
  'AppManaged': False,
  'S3Input': {'S3Uri': ParameterString(name='InputData', parameter_type=<
ParameterTypeEnum.STRING: 'String'>, default_value='s3://dlai-practical-d
ata-science/data/raw/'),
   'LocalPath': '/opt/ml/processing/input/data/',
   'S3DataType': 'S3Prefix',
   'S3InputMode': 'File',
   'S3DataDistributionType': 'ShardedByS3Key',
   'S3CompressionType': 'None'}},
 {'InputName': 'code',
  'AppManaged': False,
  'S3Input': {'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker-s
cikit-learn-2022-09-04-06-59-33-899/input/code/prepare_data.py',
   'LocalPath': '/opt/ml/processing/input/code',
   'S3DataType': 'S3Prefix',
   'S3InputMode': 'File',
   'S3DataDistributionType': 'FullyReplicated',
   'S3CompressionType': 'None'}}]
```

Exercise 2

Pull and review configuration of the processing outputs from the arguments of the processing step.

Instructions: Find the required key in the `arguments` dictionary and pull the corresponding value following the example above.

```
In [16]: ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
processing_step.arguments['ProcessingOutputConfig'] # Replace None
### END SOLUTION - DO NOT delete this comment for grading purposes
```

```
Out[16]: {'Outputs': [{'OutputName': 'sentiment-train',
   'AppManaged': False,
   'S3Output': {'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker
-scikit-learn-2022-09-04-06-59-33-489/output/sentiment-train',
    'LocalPath': '/opt/ml/processing/output/sentiment/train',
    'S3UploadMode': 'EndOfJob'}},
  {'OutputName': 'sentiment-validation',
   'AppManaged': False,
   'S3Output': {'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker
-scikit-learn-2022-09-04-06-59-33-489/output/sentiment-validation',
    'LocalPath': '/opt/ml/processing/output/sentiment/validation',
    'S3UploadMode': 'EndOfJob'}},
  {'OutputName': 'sentiment-test',
   'AppManaged': False,
   'S3Output': {'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker
-scikit-learn-2022-09-04-06-59-33-489/output/sentiment-test',
    'LocalPath': '/opt/ml/processing/output/sentiment/test',
    'S3UploadMode': 'EndOfJob'}}]}
```

2. Configure training step

2.1. Define parameters

Setup the parameters for the workflow.

In [17]:
```
freeze_bert_layer = ParameterString(
    name="FreezeBertLayer",
    default_value="False",
)

epochs = ParameterInteger(
    name="Epochs",
    default_value=3
)

learning_rate = ParameterFloat(
    name="LearningRate",
    default_value=0.00001
)

train_batch_size = ParameterInteger(
    name="TrainBatchSize",
    default_value=64
)

train_steps_per_epoch = ParameterInteger(
    name="TrainStepsPerEpoch",
    default_value=50
)

validation_batch_size = ParameterInteger(
    name="ValidationBatchSize",
    default_value=64
)

validation_steps_per_epoch = ParameterInteger(
    name="ValidationStepsPerEpoch",
    default_value=50
)

seed = ParameterInteger(
    name="Seed",
    default_value=42
)

run_validation = ParameterString(
    name="RunValidation",
    default_value="True",
)

train_instance_count = ParameterInteger(
    name="TrainInstanceCount",
    default_value=1
)

train_instance_type = ParameterString(
    name="TrainInstanceType",
    default_value="ml.c5.9xlarge"
)

train_volume_size = ParameterInteger(
    name="TrainVolumeSize",
    default_value=256
)

input_mode = ParameterString(
    name="InputMode",
    default_value="File",
)
```

2.2. Configure hyper-parameters

Setup the dictionary that will be passed into the hyperparameters argument.

In [18]:

```
hyperparameters={
    'max_seq_length': max_seq_length,
    'freeze_bert_layer': freeze_bert_layer,
    'epochs': epochs,
    'learning_rate': learning_rate,
    'train_batch_size': train_batch_size,
    'train_steps_per_epoch': train_steps_per_epoch,
    'validation_batch_size': validation_batch_size,
    'validation_steps_per_epoch': validation_steps_per_epoch,
    'seed': seed,
    'run_validation': run_validation
}
```

2.3. Configure model-evaluation metrics

Choose loss and accuracy as the evaluation metrics.

In [19]:

```
metric_definitions = [
     {'Name': 'validation:loss', 'Regex': 'val_loss: ([0-9.]+)'},
     {'Name': 'validation:accuracy', 'Regex': 'val_acc: ([0-9.]+)'},
]
```

For example, these sample log lines...

```
[step: 100] val_loss: 0.55 - val_acc: 74.64%
```

...will produce the following metrics in CloudWatch:

`validation:loss` = 0.55

`validation:accuracy` = 74.64

2.4. Configure the `PyTorchEstimator`

Configure an estimator and the input dataset. A typical training script loads data from the input channels, configures training with hyperparameters, trains a model, and saves a model to `model_dir` so that it can be hosted later.

In [20]:
```python
from sagemaker.pytorch import PyTorch as PyTorchEstimator

estimator = PyTorchEstimator(
    entry_point='train.py',
    source_dir='src',
    role=role,
    instance_count=train_instance_count,
    instance_type=train_instance_type,
    volume_size=train_volume_size,
    py_version='py3',
    framework_version='1.6.0',
    hyperparameters=hyperparameters,
    metric_definitions=metric_definitions,
    input_mode=input_mode
)
```

2.5. Setup pipeline step caching

Step signature caching allows SageMaker Pipelines, before executing a step, to find a previous execution of a step that was called using the same arguments. Cache hit gets created if the previous execution is found. Then during execution instead of recomputing the step, pipelines propagates the values from the cache hit.

Timeout period is defined using ISO 8601 format, it can contain a year, month, week, day, hour, and minute value.

More details on SageMaker Pipeline step caching can be found here.

In [21]:
```python
from sagemaker.workflow.steps import CacheConfig

cache_config = CacheConfig(enable_caching=True, expire_after="PT1H") # PT
```

2.6. Configure the `TrainingStep`

Now configure the `TrainingStep` calling the outputs of the processing step:

```
In [22]: from sagemaker.inputs import TrainingInput
         from sagemaker.workflow.steps import TrainingStep

         training_step = TrainingStep(
             name='Train',
             estimator=estimator,
             inputs={
                 'train': TrainingInput(
                     s3_data=processing_step.properties.ProcessingOutputConfig.Out
                         'sentiment-train'
                     ].S3Output.S3Uri,
                     content_type='text/csv'
                 ),
                 'validation': TrainingInput(
                     s3_data=processing_step.properties.ProcessingOutputConfig.Out
                         'sentiment-validation'
                     ].S3Output.S3Uri,
                     content_type='text/csv'
                 )
             },
             cache_config=cache_config
         )

         print(training_step)
```

```
TrainingStep(name='Train', step_type=<StepTypeEnum.TRAINING: 'Training'>)
```

Exercise 3

Use `__dict__` method to print out attributes of the training step properties. Briefly review the result. The attributes match the object model of the DescribeTrainingJob response object.

```
In [23]: ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
         training_step.properties.__dict__ # Replace all None
         ### END SOLUTION - DO NOT delete this comment for grading purposes
```

```
Out[23]: {'_path': 'Steps.Train',
          '_shape_name': 'DescribeTrainingJobResponse',
          'TrainingJobName': <sagemaker.workflow.properties.Properties at 0x7f52ab
         de1790>,
          'TrainingJobArn': <sagemaker.workflow.properties.Properties at 0x7f52abd
         e1d10>,
          'TuningJobArn': <sagemaker.workflow.properties.Properties at 0x7f52abde1
         bd0>,
          'LabelingJobArn': <sagemaker.workflow.properties.Properties at 0x7f52abd
         e1b50>,
          'AutoMLJobArn': <sagemaker.workflow.properties.Properties at 0x7f52abde1
         550>,
          'ModelArtifacts': <sagemaker.workflow.properties.Properties at 0x7f52abd
         e1c50>,
          'TrainingJobStatus': <sagemaker.workflow.properties.Properties at 0x7f52
         abaa9ed0>,
          'SecondaryStatus': <sagemaker.workflow.properties.Properties at 0x7f52ab
         aa9fd0>,
          'FailureReason': <sagemaker.workflow.properties.Properties at 0x7f52ac16
         1690>,
          'HyperParameters': <sagemaker.workflow.properties.Properties at 0x7f52ab
         ba8650>,
          'AlgorithmSpecification': <sagemaker.workflow.properties.Properties at 0
         x7f52abba80d0>,
          'RoleArn': <sagemaker.workflow.properties.Properties at 0x7f52abba8d50>,
          'InputDataConfig': <sagemaker.workflow.properties.PropertiesList at 0x7f
         52abba8a90>,
          'OutputDataConfig': <sagemaker.workflow.properties.Properties at 0x7f52a
         bba8810>,
          'ResourceConfig': <sagemaker.workflow.properties.Properties at 0x7f52abb
         a8890>,
          'VpcConfig': <sagemaker.workflow.properties.Properties at 0x7f52abba8e10
         >,
          'StoppingCondition': <sagemaker.workflow.properties.Properties at 0x7f52
         abba8350>,
          'CreationTime': <sagemaker.workflow.properties.Properties at 0x7f52abba8
         4d0>,
          'TrainingStartTime': <sagemaker.workflow.properties.Properties at 0x7f52
         abba8410>,
          'TrainingEndTime': <sagemaker.workflow.properties.Properties at 0x7f52ab
         d86410>,
          'LastModifiedTime': <sagemaker.workflow.properties.Properties at 0x7f52a
         bd86590>,
          'SecondaryStatusTransitions': <sagemaker.workflow.properties.PropertiesL
         ist at 0x7f52abd863d0>,
          'FinalMetricDataList': <sagemaker.workflow.properties.PropertiesList at
         0x7f52abd86250>,
          'EnableNetworkIsolation': <sagemaker.workflow.properties.Properties at 0
         x7f52abd865d0>,
          'EnableInterContainerTrafficEncryption': <sagemaker.workflow.properties.
         Properties at 0x7f52abd86150>,
          'EnableManagedSpotTraining': <sagemaker.workflow.properties.Properties a
         t 0x7f52abd861d0>,
          'CheckpointConfig': <sagemaker.workflow.properties.Properties at 0x7f52a
         bd86190>,
          'TrainingTimeInSeconds': <sagemaker.workflow.properties.Properties at 0x
         7f52abd86350>,
          'BillableTimeInSeconds': <sagemaker.workflow.properties.Properties at 0x
         7f52abd86310>,
          'DebugHookConfig': <sagemaker.workflow.properties.Properties at 0x7f52ab
         d86550>,
          'ExperimentConfig': <sagemaker.workflow.properties.Properties at 0x7f52a
         bd86490>,
          'DebugRuleConfigurations': <sagemaker.workflow.properties.PropertiesList
         at 0x7f52aba96450>,
          'TensorBoardOutputConfig': <sagemaker.workflow.properties.Properties at
         0x7f52aba96610>,
          'DebugRuleEvaluationStatuses': <sagemaker.workflow.properties.Properties
         List at 0x7f52aba96990>,
          'ProfilerConfig': <sagemaker.workflow.properties.Properties at 0x7f52aba
         96910>,
          'ProfilerRuleConfigurations': <sagemaker.workflow.properties.PropertiesL
         ist at 0x7f52aba96290>,
          'ProfilerRuleEvaluationStatuses': <sagemaker.workflow.properties.Propert
         iesList at 0x7f52aba96b10>,
          'ProfilingStatus': <sagemaker.workflow.properties.Properties at 0x7f52ab
         a96a50>,
          'RetryStrategy': <sagemaker.workflow.properties.Properties at 0x7f52aba9
         6590>,
          'Environment': <sagemaker.workflow.properties.Properties at 0x7f52aba962
         50>}
```

3. Configure model-evaluation step

First, develop an evaluation script that will be specified in the model evaluation processing step. The evaluation script users the trained model and the test dataset to produce a JSON file with classification evaluation metrics such as accuracy.

After pipeline execution, you will examine the resulting `evaluation.json` for analysis.

The evaluation script performs the following steps:

- loads in the model
- reads in the test data
- issues a bunch of predictions against the test data
- builds a classification report, including accuracy
- saves the evaluation report to the evaluation directory

Create an instance of the `SKLearnProcessor` to run our evaluation script as a scikit-learn-based SageMaker processing job.

In [24]:
```
from sagemaker.sklearn.processing import SKLearnProcessor

evaluation_processor = SKLearnProcessor(
    framework_version='0.23-1',
    role=role,
    instance_type=processing_instance_type,
    instance_count=processing_instance_count,
    env={'AWS_DEFAULT_REGION': region},
    max_runtime_in_seconds=7200
)
```

Setup the output `PropertyFile`.

In [25]:
```
from sagemaker.workflow.properties import PropertyFile

evaluation_report = PropertyFile(
    name='EvaluationReport',
    output_name='metrics',
    path='evaluation.json'
)
```

Use the processor instance to construct a `ProcessingStep`, along with the input and output channels and the code that will be executed when the pipeline invokes pipeline execution. This is very similar to a processor instance's `run` method.

```
In [26]: from sagemaker.processing import ProcessingInput, ProcessingOutput

         evaluation_step = ProcessingStep(
             name='EvaluateModel',
             processor=evaluation_processor,
             code='src/evaluate_model_metrics.py',
             inputs=[
                 ProcessingInput(
                     source=training_step.properties.ModelArtifacts.S3ModelArtifac
                     destination='/opt/ml/processing/input/model'
                 ),
                 ProcessingInput(
                     source=processing_step.properties.ProcessingOutputConfig.Outp
                     destination='/opt/ml/processing/input/data'
                 )
             ],
             outputs=[
                 ProcessingOutput(output_name='metrics',
                                  s3_upload_mode='EndOfJob',
                                  source='/opt/ml/processing/output/metrics/'),
             ],
             job_arguments=[
                 '--max-seq-length', str(max_seq_length.default_value),
             ],
             property_files=[evaluation_report],
         )
```

4. Configure and register model step

4.1. Configure the model for deployment

Use the estimator instance that was used for the training step to construct an instance of `RegisterModel`. The result of executing `RegisterModel` in a pipeline is a model package. A model package is a reusable model artifacts abstraction that packages all ingredients necessary for inference. Primarily, it consists of an inference specification that defines the inference image to use along with an optional model weights location.

A model package group is a collection of model packages. You can create a model package group for a specific ML business problem, and you can keep adding versions/model packages into it. Typically, customers are expected to create a ModelPackageGroup for a SageMaker workflow pipeline so that they can keep adding versions/model packages to the group for every workflow pipeline run.

The construction of `RegisterModel` is very similar to an estimator instance's `register` method, for those familiar with the existing Python SDK.

In particular, you will pass in the `S3ModelArtifacts` from the `training_step` properties.

Of note, here you will be provided a specific model package group name which will be used in the Model Registry and Continuous Integration/Continuous Deployment (CI/CD) work later on. Let's setup the variables.

In [27]:
```
model_approval_status = ParameterString(
    name="ModelApprovalStatus",
    default_value="PendingManualApproval"
)

deploy_instance_type = ParameterString(
    name="DeployInstanceType",
    default_value="ml.m5.large"
)

deploy_instance_count = ParameterInteger(
    name="DeployInstanceCount",
    default_value=1
)
```

In [28]:
```
model_package_group_name = f"BERT-Reviews-{timestamp}"

print(model_package_group_name)
```

```
BERT-Reviews-1662274772
```

Configure the `ModelMetrics` to be stored as metadata.

In [29]:
```python
from sagemaker.model_metrics import MetricsSource, ModelMetrics

model_metrics = ModelMetrics(
    model_statistics=MetricsSource(
        s3_uri="{}/evaluation.json".format(
            evaluation_step.arguments["ProcessingOutputConfig"]["Outputs"
        ),
        content_type="application/json"
    )
)

print(model_metrics)
```

```
<sagemaker.model_metrics.ModelMetrics object at 0x7f52ab87b490>
```

Define deployment image for inference.

In [30]:
```python
inference_image_uri = sagemaker.image_uris.retrieve(
    framework="pytorch",
    region=region,
    version="1.6.0",
    py_version="py36",
    instance_type=deploy_instance_type,
    image_scope="inference"
)
print(inference_image_uri)
```

```
763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference:1.6.0-cpu-
py36
```

4.2. Register the model for deployment

Exercise 4

Configure the register model step.

Instructions: Pass the inference image defined above into the `image_uri` argument of the function `RegisterModel`. Review the rest of the arguments.

In [31]:
```
from sagemaker.workflow.step_collections import RegisterModel

register_step = RegisterModel(
    name="RegisterModel",
    estimator=estimator,
    ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
    image_uri=inference_image_uri, # Replace None
    ### END SOLUTION - DO NOT delete this comment for grading purposes
    model_data=training_step.properties.ModelArtifacts.S3ModelArtifacts,
    content_types=["application/jsonlines"],
    response_types=["application/jsonlines"],
    inference_instances=[deploy_instance_type],
    transform_instances=[deploy_instance_type], # batch transform is not
    model_package_group_name=model_package_group_name,
    approval_status=model_approval_status,
    model_metrics=model_metrics
)
```

5. Create model for deployment step

Exercise 5

Configure model for deployment.

Instructions: Pass the same inference image into the `image_uri` argument of the function `Model`.

In [32]:
```
from sagemaker.model import Model

model_name = 'bert-model-{}'.format(timestamp)

model = Model(
    name=model_name,
    ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
    image_uri=inference_image_uri, # Replace None
    ### END SOLUTION - DO NOT delete this comment for grading purposes
    model_data=training_step.properties.ModelArtifacts.S3ModelArtifacts,
    sagemaker_session=sess,
    role=role,
)
```

Now configure create model input:

In [33]:
```
from sagemaker.inputs import CreateModelInput

create_inputs = CreateModelInput(
    instance_type=deploy_instance_type,
)
```

Exercise 6

Configure create model step for the workflow.

Instructions: Pass defined above model (the model object, not its name) and model inputs configuration into the related arguments of the function `CreateModelStep`.

In [34]:
```
from sagemaker.workflow.steps import CreateModelStep

create_step = CreateModelStep(
    name="CreateModel",
    ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
    model=model, # Replace None
    inputs=create_inputs, # Replace None
    ### END SOLUTION - DO NOT delete this comment for grading purposes
)
```

6. Check accuracy condition step

Finally, you would like to only register this model if the accuracy of the model, as determined by our evaluation step `evaluation_step`, exceeded some value. A `ConditionStep` allows for pipelines to support conditional execution in the pipeline DAG based on conditions of step properties.

Below, you will:

- define a minimum accuracy value as a parameter
- define a `ConditionGreaterThan` on the accuracy value found in the output of the evaluation step, `evaluation_step`.
- use the condition in the list of conditions in a `ConditionStep`
- pass the `RegisterModel` step collection into the `if_steps` of the `ConditionStep`

In [35]:
```
min_accuracy_value = ParameterFloat(
    name="MinAccuracyValue",
    default_value=0.33 # random choice from three classes
)
```

In [36]:
```python
from sagemaker.workflow.conditions import ConditionGreaterThanOrEqualTo
from sagemaker.workflow.condition_step import (
    ConditionStep,
    JsonGet,
)

minimum_accuracy_condition = ConditionGreaterThanOrEqualTo(
    left=JsonGet(
        step=evaluation_step,
        property_file=evaluation_report,
        json_path="metrics.accuracy.value",
    ),
    right=min_accuracy_value # minimum accuracy threshold
)

minimum_accuracy_condition_step = ConditionStep(
    name="AccuracyCondition",
    conditions=[minimum_accuracy_condition],
    if_steps=[register_step, create_step], # successfully exceeded or equ
    else_steps=[], # did not exceed the minimum accuracy, the model will
)
```

7. Create pipeline

7.1. Define a pipeline of parameters, steps, and conditions

Let's tie it all up into a workflow pipeline so you can execute it, and even schedule it.

A pipeline requires a `name`, `parameters`, and `steps`. Names must be unique within an `(account, region)` pair so you can append the timestamp to the name to reduce the chance of name conflict.

Note:

- All the parameters used in the definitions must be present.
- Steps passed into the pipeline need not be in the order of execution. The SageMaker workflow service will resolve the *data dependency* DAG as steps the execution complete.
- Steps must be unique to either pipeline step list or a single condition step if/else list.

In [37]:
```
from sagemaker.workflow.pipeline import Pipeline

pipeline = Pipeline(
    name=pipeline_name,
    parameters=[
        input_data,
        processing_instance_count,
        processing_instance_type,
        max_seq_length,
        balance_dataset,
        train_split_percentage,
        validation_split_percentage,
        test_split_percentage,
        feature_store_offline_prefix,
        feature_group_name,
        epochs,
        learning_rate,
        train_batch_size,
        train_steps_per_epoch,
        validation_batch_size,
        validation_steps_per_epoch,
        freeze_bert_layer,
        seed,
        train_instance_count,
        train_instance_type,
        train_volume_size,
        input_mode,
        run_validation,
        min_accuracy_value,
        model_approval_status,
        deploy_instance_type,
        deploy_instance_count
    ],
    steps=[processing_step, training_step, evaluation_step, minimum_accur
    sagemaker_session=sess,
)
```

Let's examine the JSON of the pipeline definition that meets the SageMaker Workflow Pipeline DSL specification.

By examining the definition, you are also confirming that the pipeline was well-defined, and that the parameters and step properties resolve correctly.

In [38]:
```
import json
from pprint import pprint

definition = json.loads(pipeline.definition())

pprint(definition)
```

```
No finished training job found associated with this estimator. Please mak
e sure this estimator is only used for building workflow config
{'Metadata': {},
 'Parameters': [{'DefaultValue': 's3://dlai-practical-data-science/data/r
aw/',
                 'Name': 'InputData',
                 'Type': 'String'},
                {'DefaultValue': 1,
                 'Name': 'ProcessingInstanceCount',
                 'Type': 'Integer'},
                {'DefaultValue': 'ml.c5.2xlarge',
                 'Name': 'ProcessingInstanceType',
                 'Type': 'String'},
                {'DefaultValue': 128,
                 'Name': 'MaxSeqLength',
                 'Type': 'Integer'},
                {'DefaultValue': 'True',
                 'Name': 'BalanceDataset',
                 'Type': 'String'},
                {'DefaultValue': 0.9,
                 'Name': 'TrainSplitPercentage',
                 'Type': 'Float'},
                {'DefaultValue': 0.05,
                 'Name': 'ValidationSplitPercentage',
                 'Type': 'Float'},
                {'DefaultValue': 0.05,
                 'Name': 'TestSplitPercentage',
                 'Type': 'Float'},
                {'DefaultValue': 'reviews-feature-store-1662274772',
                 'Name': 'FeatureStoreOfflinePrefix',
                 'Type': 'String'},
                {'DefaultValue': 'reviews-feature-group-1662274772',
                 'Name': 'FeatureGroupName',
                 'Type': 'String'},
                {'DefaultValue': 3, 'Name': 'Epochs', 'Type': 'Integer'},
                {'DefaultValue': 1e-05,
                 'Name': 'LearningRate',
                 'Type': 'Float'},
                {'DefaultValue': 64,
                 'Name': 'TrainBatchSize',
                 'Type': 'Integer'},
                {'DefaultValue': 50,
                 'Name': 'TrainStepsPerEpoch',
                 'Type': 'Integer'},
                {'DefaultValue': 64,
                 'Name': 'ValidationBatchSize',
                 'Type': 'Integer'},
                {'DefaultValue': 50,
                 'Name': 'ValidationStepsPerEpoch',
                 'Type': 'Integer'},
                {'DefaultValue': 'False',
                 'Name': 'FreezeBertLayer',
                 'Type': 'String'},
                {'DefaultValue': 42, 'Name': 'Seed', 'Type': 'Integer'},
                {'DefaultValue': 1,
                 'Name': 'TrainInstanceCount',
                 'Type': 'Integer'},
                {'DefaultValue': 'ml.c5.9xlarge',
                 'Name': 'TrainInstanceType',
                 'Type': 'String'},
                {'DefaultValue': 256,
                 'Name': 'TrainVolumeSize',
                 'Type': 'Integer'},
                {'DefaultValue': 'File', 'Name': 'InputMode', 'Type': 'St
ring'},
                {'DefaultValue': 'True',
                 'Name': 'RunValidation',
                 'Type': 'String'},
                {'DefaultValue': 0.33,
          'Name': 'MinAccuracyValue',
          'Type': 'Float'},
         {'DefaultValue': 'PendingManualApproval',
          'Name': 'ModelApprovalStatus',
          'Type': 'String'},
         {'DefaultValue': 'ml.m5.large',
          'Name': 'DeployInstanceType',
          'Type': 'String'},
         {'DefaultValue': 1,
          'Name': 'DeployInstanceCount',
          'Type': 'Integer'}],
'Steps': [{'Arguments': {'AppSpecification': {'ContainerArguments': ['--
train-split-percentage',
                                                                      '0.
9',
                                                                      '--
validation-split-percentage',
                                                                      '0.
05',
                                                                      '--
test-split-percentage',
                                                                      '0.
05',
                                                                      '--
balance-dataset',
                                                                      'Tr
ue',
                                                                      '--
max-seq-length',
                                                                      '12
8',
                                                                      '--
feature-store-offline-prefix',
                                                                      're
views-feature-store-1662274772',
                                                                      '--
feature-group-name',
                                                                      're
views-feature-group-1662274772'],
                                               'ContainerEntrypoint': ['p
ython3',
                                                                       '/
opt/ml/processing/input/code/prepare_data.py'],
                                               'ImageUri': '683313688378.
dkr.ecr.us-east-1.amazonaws.com/sagemaker-scikit-learn:0.23-1-cpu-py3'},
                          'Environment': {'AWS_DEFAULT_REGION': 'us-east-
1'},
                          'ProcessingInputs': [{'AppManaged': False,
                                                'InputName': 'raw-input-d
ata',
                                                'S3Input': {'LocalPath':
'/opt/ml/processing/input/data/',
                                                            'S3Compressio
nType': 'None',
                                                            'S3DataDistri
butionType': 'ShardedByS3Key',
                                                            'S3DataType':
'S3Prefix',
                                                            'S3InputMode'
: 'File',
                                                            'S3Uri': {'Ge
t': 'Parameters.InputData'}}},
                                                {'AppManaged': False,
                                                 'InputName': 'code',
                                                 'S3Input': {'LocalPath':
'/opt/ml/processing/input/code',
                                                             'S3Compressio
nType': 'None',
                                                             'S3DataDistri
butionType': 'FullyReplicated',
                                                             'S3DataType':
'S3Prefix',
                                                             'S3InputMode'
: 'File',
                                                             'S3Uri': 's3:
//sagemaker-us-east-1-141881372941/sagemaker-scikit-learn-2022-09-04-06-5
9-36-012/input/code/prepare_data.py'}}],
                          'ProcessingOutputConfig': {'Outputs': [{'AppMan
aged': False,
                                                                   'Output
Name': 'sentiment-train',
                                                                   'S3Outp
ut': {'LocalPath': '/opt/ml/processing/output/sentiment/train',

'S3UploadMode': 'EndOfJob',

'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker-scikit-learn-20
22-09-04-06-59-33-489/output/sentiment-train'}},
                                                                  {'AppMan
aged': False,
                                                                   'Output
Name': 'sentiment-validation',
                                                                   'S3Outp
ut': {'LocalPath': '/opt/ml/processing/output/sentiment/validation',

'S3UploadMode': 'EndOfJob',

'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker-scikit-learn-20
22-09-04-06-59-33-489/output/sentiment-validation'}},
                                                                  {'AppMan
aged': False,
                                                                   'Output
Name': 'sentiment-test',
                                                                   'S3Outp
ut': {'LocalPath': '/opt/ml/processing/output/sentiment/test',

'S3UploadMode': 'EndOfJob',

'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker-scikit-learn-20
22-09-04-06-59-33-489/output/sentiment-test'}}]},
                          'ProcessingResources': {'ClusterConfig': {'Inst
anceCount': {'Get': 'Parameters.ProcessingInstanceCount'},
                                                                    'Inst
anceType': {'Get': 'Parameters.ProcessingInstanceType'},
                                                                    'Volu
meSizeInGB': 30}},
                          'RoleArn': 'arn:aws:iam::141881372941:role/sage
maker-studio-vpc-firewall-us-east-1-sagemaker-execution-role'},
            'Name': 'Processing',
            'Type': 'Processing'},
           {'Arguments': {'AlgorithmSpecification': {'EnableSageMakerMetr
icsTimeSeries': True,
                                                      'MetricDefinitions':
[{'Name': 'validation:loss',

'Regex': 'val_loss: '

'([0-9.]+)'},

{'Name': 'validation:accuracy',

'Regex': 'val_acc: '

'([0-9.]+)'}],
                                                      'TrainingImage': '76
3104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-training:1.6.0-cpu-py3
',
                                                      'TrainingInputMode':
{'Get': 'Parameters.InputMode'}},
                           'DebugHookConfig': {'CollectionConfigurations':
[],
                                               'S3OutputPath': 's3://sagem
aker-us-east-1-141881372941/'},
                           'HyperParameters': {'epochs': '3',
                                               'freeze_bert_layer': '"Fals
e"',
                                               'learning_rate': '1e-05',
                                               'max_seq_length': '128',
                                               'run_validation': '"True"',
                                               'sagemaker_container_log_le
vel': '20',
                                               'sagemaker_job_name': '"pyt
orch-training-2022-09-04-06-59-36-136"',
                                               'sagemaker_program': '"trai
n.py"',
                                               'sagemaker_region': '"us-ea
st-1"',
                                               'sagemaker_submit_directory
': '"s3://sagemaker-us-east-1-141881372941/pytorch-training-2022-09-04-06
-59-36-136/source/sourcedir.tar.gz"',
                                               'seed': '42',
                                               'train_batch_size': '64',
                                               'train_steps_per_epoch': '5
0',
                                               'validation_batch_size': '6
4',
                                               'validation_steps_per_epoch
': '50'},
                           'InputDataConfig': [{'ChannelName': 'train',
                                                'ContentType': 'text/csv',
                                                'DataSource': {'S3DataSour
ce': {'S3DataDistributionType': 'FullyReplicated',

'S3DataType': 'S3Prefix',

'S3Uri': {'Get': "Steps.Processing.ProcessingOutputConfig.Outputs['sentim
ent-train'].S3Output.S3Uri"}}}},
                                               {'ChannelName': 'validation
',
                                                'ContentType': 'text/csv',
                                                'DataSource': {'S3DataSour
ce': {'S3DataDistributionType': 'FullyReplicated',
'S3DataType': 'S3Prefix',

'S3Uri': {'Get': "Steps.Processing.ProcessingOutputConfig.Outputs['sentim
ent-validation'].S3Output.S3Uri"}}}}],
                          'OutputDataConfig': {'S3OutputPath': 's3://sage
maker-us-east-1-141881372941/'},
                          'ProfilerConfig': {'S3OutputPath': 's3://sagema
ker-us-east-1-141881372941/'},
                          'ProfilerRuleConfigurations': [{'RuleConfigurat
ionName': 'ProfilerReport-1662274776',
                                                          'RuleEvaluatorI
mage': '503895931360.dkr.ecr.us-east-1.amazonaws.com/sagemaker-debugger-r
ules:latest',
                                                          'RuleParameters
': {'rule_to_invoke': 'ProfilerReport'}}],
                          'ResourceConfig': {'InstanceCount': {'Get': 'Pa
rameters.TrainInstanceCount'},
                                             'InstanceType': {'Get': 'Par
ameters.TrainInstanceType'},
                                             'VolumeSizeInGB': {'Get': 'P
arameters.TrainVolumeSize'}},
                          'RoleArn': 'arn:aws:iam::141881372941:role/sage
maker-studio-vpc-firewall-us-east-1-sagemaker-execution-role',
                          'StoppingCondition': {'MaxRuntimeInSeconds': 86
400}},
            'CacheConfig': {'Enabled': True, 'ExpireAfter': 'PT1H'},
            'Name': 'Train',
            'Type': 'Training'},
           {'Arguments': {'AppSpecification': {'ContainerArguments': ['--
max-seq-length',
                                                                      '12
8'],
                                               'ContainerEntrypoint': ['p
ython3',
                                                                       '/
opt/ml/processing/input/code/evaluate_model_metrics.py'],
                                               'ImageUri': '683313688378.
dkr.ecr.us-east-1.amazonaws.com/sagemaker-scikit-learn:0.23-1-cpu-py3'},
                          'Environment': {'AWS_DEFAULT_REGION': 'us-east-
1'},
                          'ProcessingInputs': [{'AppManaged': False,
                                                'InputName': 'input-1',
                                                'S3Input': {'LocalPath':
'/opt/ml/processing/input/model',
                                                            'S3Compressio
nType': 'None',
                                                            'S3DataDistri
butionType': 'FullyReplicated',
                                                            'S3DataType':
'S3Prefix',
                                                            'S3InputMode'
: 'File',
                                                            'S3Uri': {'Ge
t': 'Steps.Train.ModelArtifacts.S3ModelArtifacts'}}},
                                               {'AppManaged': False,
                                                'InputName': 'input-2',
                                                'S3Input': {'LocalPath':
'/opt/ml/processing/input/data',
                                                            'S3Compressio
nType': 'None',
                                                            'S3DataDistri
```

```
butionType': 'FullyReplicated',
                                                               'S3DataType':
'S3Prefix',
                                                               'S3InputMode'
: 'File',
                                                               'S3Uri': {'Ge
t': "Steps.Processing.ProcessingOutputConfig.Outputs['sentiment-test'].S3
Output.S3Uri"}}},
                                                {'AppManaged': False,
                                                 'InputName': 'code',
                                                 'S3Input': {'LocalPath':
'/opt/ml/processing/input/code',
                                                               'S3Compressio
nType': 'None',
                                                               'S3DataDistri
butionType': 'FullyReplicated',
                                                               'S3DataType':
'S3Prefix',
                                                               'S3InputMode'
: 'File',
                                                               'S3Uri': 's3:
//sagemaker-us-east-1-141881372941/sagemaker-scikit-learn-2022-09-04-06-5
9-36-607/input/code/evaluate_model_metrics.py'}}],
                           'ProcessingOutputConfig': {'Outputs': [{'AppMan
aged': False,
                                                                     'Output
Name': 'metrics',
                                                                     'S3Outp
ut': {'LocalPath': '/opt/ml/processing/output/metrics/',

'S3UploadMode': 'EndOfJob',

'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker-scikit-learn-20
22-09-04-06-59-35-126/output/metrics'}}]},
                           'ProcessingResources': {'ClusterConfig': {'Inst
anceCount': {'Get': 'Parameters.ProcessingInstanceCount'},
                                                                       'Inst
anceType': {'Get': 'Parameters.ProcessingInstanceType'},
                                                                       'Volu
meSizeInGB': 30}},
                           'RoleArn': 'arn:aws:iam::141881372941:role/sage
maker-studio-vpc-firewall-us-east-1-sagemaker-execution-role',
                           'StoppingCondition': {'MaxRuntimeInSeconds': 72
00}},
             'Name': 'EvaluateModel',
             'PropertyFiles': [{'FilePath': 'evaluation.json',
                                'OutputName': 'metrics',
                                'PropertyFileName': 'EvaluationReport'}],
             'Type': 'Processing'},
            {'Arguments': {'Conditions': [{'LeftValue': {'Std:JsonGet': {'
Path': 'metrics.accuracy.value',
                                                                          '
PropertyFile': {'Get': 'Steps.EvaluateModel.PropertyFiles.EvaluationRepor
t'}}},
                                            'RightValue': {'Get': 'Paramete
rs.MinAccuracyValue'},
                                            'Type': 'GreaterThanOrEqualTo'}
],
                           'ElseSteps': [],
                           'IfSteps': [{'Arguments': {'InferenceSpecificat
ion': {'Containers': [{'Image': '763104351884.dkr.ecr.us-east-1.amazonaws
.com/pytorch-inference:1.6.0-cpu-py36',

'ModelDataUrl': {'Get': 'Steps.Train.ModelArtifacts.S3ModelArtifacts'}}],

'SupportedContentTypes': ['application/jsonlines'],

'SupportedRealtimeInferenceInstanceTypes': [{'Get': 'Parameters.DeployIns
tanceType'}],

'SupportedResponseMIMETypes': ['application/jsonlines'],

'SupportedTransformInstanceTypes': [{'Get': 'Parameters.DeployInstanceTyp
e'}]},
                                                     'ModelApprovalStatus
': {'Get': 'Parameters.ModelApprovalStatus'},
                                                     'ModelMetrics': {'Mo
delQuality': {'Statistics': {'ContentType': 'application/json',

'S3Uri': 's3://sagemaker-us-east-1-141881372941/sagemaker-scikit-learn-20
22-09-04-06-59-35-126/output/metrics/evaluation.json'}}},
                                                     'ModelPackageGroupNa
me': 'BERT-Reviews-1662274772'},
                                       'Name': 'RegisterModel',
                                       'Type': 'RegisterModel'},
                                      {'Arguments': {'ExecutionRoleArn':
'arn:aws:iam::141881372941:role/sagemaker-studio-vpc-firewall-us-east-1-s
agemaker-execution-role',
                                                     'PrimaryContainer':
{'Environment': {},

'Image': '763104351884.dkr.ecr.us-east-1.amazonaws.com/pytorch-inference:
1.6.0-cpu-py36',

'ModelDataUrl': {'Get': 'Steps.Train.ModelArtifacts.S3ModelArtifacts'}}},
                                       'Name': 'CreateModel',
                                       'Type': 'Model'}]},
            'Name': 'AccuracyCondition',
            'Type': 'Condition'}],
 'Version': '2020-12-01'}
```

Ignore the `WARNING` below

Create pipeline using the `create` method and then print the Amazon Resource Name (ARN) of it.

In [39]:
```python
response = pipeline.create(role_arn=role)

pipeline_arn = response["PipelineArn"]
print(pipeline_arn)
```

```
No finished training job found associated with this estimator. Please mak
e sure this estimator is only used for building workflow config
arn:aws:sagemaker:us-east-1:141881372941:pipeline/bert-pipeline-166227477
2
```

*Ignore the `WARNING` ^^ above ^^*

7.2. Start Pipeline

Let's submit our pipeline definition to the Amazon SageMaker Pipeline service. The role passed in will be used by the service to create all the jobs defined in the steps. You will start the pipeline using the parameters passed into the `start()` function.

In [40]:
```
execution = pipeline.start(
    parameters=dict(
        InputData=raw_input_data_s3_uri,
        ProcessingInstanceCount=1,
        ProcessingInstanceType='ml.c5.2xlarge',
        MaxSeqLength=128,
        BalanceDataset='True',
        TrainSplitPercentage=0.9,
        ValidationSplitPercentage=0.05,
        TestSplitPercentage=0.05,
        FeatureStoreOfflinePrefix='reviews-feature-store-'+str(timestamp)
        FeatureGroupName='reviews-feature-group-'+str(timestamp),
        Epochs=3,
        LearningRate=0.000012,
        TrainBatchSize=64,
        TrainStepsPerEpoch=50,
        ValidationBatchSize=64,
        ValidationStepsPerEpoch=64,
        FreezeBertLayer='False',
        Seed=42,
        TrainInstanceCount=1,
        TrainInstanceType='ml.c5.9xlarge',
        TrainVolumeSize=256,
        InputMode='File',
        RunValidation='True',
        MinAccuracyValue=0.01,
        ModelApprovalStatus='PendingManualApproval',
        DeployInstanceType='ml.m5.large',
        DeployInstanceCount=1
    )
)

print(execution.arn)
```

```
arn:aws:sagemaker:us-east-1:141881372941:pipeline/bert-pipeline-166227477
2/execution/dh54d7we8u88
```

7.3. Wait for pipeline execution

Now you can describe execution instance and list the steps in the execution to find out more about the execution.

In [41]:
```
from pprint import pprint

execution_run = execution.describe()
pprint(execution_run)
```

```
{'CreatedBy': {'DomainId': 'd-4ftrlzf4sfyd',
               'UserProfileArn': 'arn:aws:sagemaker:us-east-1:14188137294
1:user-profile/d-4ftrlzf4sfyd/sagemaker-user-profile-us-east-1',
               'UserProfileName': 'sagemaker-user-profile-us-east-1'},
 'CreationTime': datetime.datetime(2022, 9, 4, 6, 59, 37, 893000, tzinfo=
tzlocal()),
 'LastModifiedBy': {'DomainId': 'd-4ftrlzf4sfyd',
                    'UserProfileArn': 'arn:aws:sagemaker:us-east-1:141881
372941:user-profile/d-4ftrlzf4sfyd/sagemaker-user-profile-us-east-1',
                    'UserProfileName': 'sagemaker-user-profile-us-east-1'
},
 'LastModifiedTime': datetime.datetime(2022, 9, 4, 6, 59, 37, 893000, tzi
nfo=tzlocal()),
 'PipelineArn': 'arn:aws:sagemaker:us-east-1:141881372941:pipeline/bert-p
ipeline-1662274772',
 'PipelineExecutionArn': 'arn:aws:sagemaker:us-east-1:141881372941:pipeli
ne/bert-pipeline-1662274772/execution/dh54d7we8u88',
 'PipelineExecutionDisplayName': 'execution-1662274777995',
 'PipelineExecutionStatus': 'Executing',
 'ResponseMetadata': {'HTTPHeaders': {'content-length': '815',
                                      'content-type': 'application/x-amz-
json-1.1',
                                      'date': 'Sun, 04 Sep 2022 06:59:37
GMT',
                                      'x-amzn-requestid': 'dea8a727-82a4-
4e49-af16-90d140f2d1fa'},
                      'HTTPStatusCode': 200,
                      'RequestId': 'dea8a727-82a4-4e49-af16-90d140f2d1fa'
,
                      'RetryAttempts': 0}}
```

Print the execution display name and its ARN:

In [42]:
```
execution_run_name = execution_run['PipelineExecutionDisplayName']
print(execution_run_name)
```

```
execution-1662274777995
```

In [43]:
```
pipeline_execution_arn = execution_run['PipelineExecutionArn']
print(pipeline_execution_arn)
```

```
arn:aws:sagemaker:us-east-1:141881372941:pipeline/bert-pipeline-166227477
2/execution/dh54d7we8u88
```

7.4. Describe completed pipeline

Wait for the first step to start running and print the information about it:

In [44]:
```
import time

time.sleep(30)

execution.list_steps()
```

Out[44]:
```
[{'StepName': 'Processing',
  'StartTime': datetime.datetime(2022, 9, 4, 6, 59, 38, 623000, tzinfo=tz
local()),
  'StepStatus': 'Executing',
  'AttemptCount': 0,
  'Metadata': {'ProcessingJob': {'Arn': 'arn:aws:sagemaker:us-east-1:1418
81372941:processing-job/pipelines-dh54d7we8u88-processing-o6skn85hoa'}}}]
```

7.5. Wait for the pipeline to complete

To get the information about the pipeline execution you can use low-level service client of the boto3 session. It is also useful for other operations that you will see below.

In the code below you will be observing the pipeline execution summary and waiting for the execution status to change from `Executing` to `Succeeded`.

This cell will take approximately 30-45 minutes to run.

In [45]:
```
%%time

import time
from pprint import pprint

sm = boto3.Session().client(service_name='sagemaker', region_name=region)

executions_response = sm.list_pipeline_executions(PipelineName=pipeline_n
pipeline_execution_status = executions_response[0]['PipelineExecutionStat
print(pipeline_execution_status)

while pipeline_execution_status=='Executing':
    try:
        executions_response = sm.list_pipeline_executions(PipelineName=pi
        pipeline_execution_status = executions_response[0]['PipelineExecu
    except Exception as e:
        print('Please wait...')
        time.sleep(30)

pprint(executions_response)
```

```
Executing
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
Please wait...
[{'PipelineExecutionArn': 'arn:aws:sagemaker:us-east-1:141881372941:pipel
ine/bert-pipeline-1662274772/execution/dh54d7we8u88',
  'PipelineExecutionDisplayName': 'execution-1662274777995',
  'PipelineExecutionStatus': 'Succeeded',
  'StartTime': datetime.datetime(2022, 9, 4, 6, 59, 37, 893000, tzinfo=tz
local())}]
CPU times: user 11 s, sys: 435 ms, total: 11.5 s
Wall time: 43min 2s
```

Wait for the pipeline ^^ above ^^ to complete.

You can list the execution steps to check out the status and artifacts:

In [46]:
```
pipeline_execution_status = executions_response[0]['PipelineExecutionStat
print(pipeline_execution_status)
```

```
Succeeded
```

In [47]:
```python
pipeline_execution_arn = executions_response[0]['PipelineExecutionArn']
print(pipeline_execution_arn)
```

```
arn:aws:sagemaker:us-east-1:141881372941:pipeline/bert-pipeline-166227477
2/execution/dh54d7we8u88
```

8. Evaluate the model

8.1. Describe evaluation metrics

Examine the resulting model evaluation after the pipeline completes. Download the resulting evaluation.json file from S3 and print the report.

In [48]:
```python
processing_job_name = None

# pull the processing step name
for execution_step in reversed(execution.list_steps()):
    if execution_step['StepName'] == 'Processing':
        processing_job_name=execution_step['Metadata']['ProcessingJob']['

# get the description of the processing job
describe_transform_processing_job_response = sm.describe_processing_job(P

# get the output S3 path
transform_output_s3_uri = describe_transform_processing_job_response['Pro
print('Transform output {}'.format(transform_output_s3_uri))
```

```
Transform output s3://sagemaker-us-east-1-141881372941/sagemaker-scikit-l
earn-2022-09-04-06-59-33-489/output/sentiment-train
```

In [49]:
```python
# list the files in the resulting output S3 path
!aws s3 ls --recursive $transform_output_s3_uri
```

```
2022-09-04 07:13:35    4876932 sagemaker-scikit-learn-2022-09-04-06-59-33
-489/output/sentiment-train/part-algo-1-womens_clothing_ecommerce_reviews
.tsv
```

Exercise 7

Pull the name of the model-evaluation step and then get the S3 path of the evaluation metrics, which will contain the evaluation report.

Instructions: Find the execution step with the step name `EvaluateModel` following the example above.

In [50]:
```
processing_job_name = None

for execution_step in reversed(execution.list_steps()):
    ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
    if execution_step['StepName'] == 'EvaluateModel': # Replace all None
    ### END SOLUTION - DO NOT delete this comment for grading purposes
        processing_job_name=execution_step['Metadata']['ProcessingJob']['

describe_evaluation_processing_job_response = sm.describe_processing_job(

evaluation_metrics_s3_uri = describe_evaluation_processing_job_response['
print('Evaluation output {}'.format(evaluation_metrics_s3_uri))
```

```
Evaluation output s3://sagemaker-us-east-1-141881372941/sagemaker-scikit-
learn-2022-09-04-06-59-35-126/output/metrics
```

8.2. Review the evaluation report

Download the evaluation report and print the accuracy.

In [51]:
```
from pprint import pprint

evaluation_json = sagemaker.s3.S3Downloader.read_file("{}/evaluation.json
    evaluation_metrics_s3_uri
))

pprint(json.loads(evaluation_json))
```

```
{'metrics': {'accuracy': {'value': 0.7475728155339806}}}
```

8.3. List pipeline artifacts

Exercise 8

Find and print the ARN and job name of the training job.

Instructions: Find the execution step with the step name `Train` following the example above.

In [52]:
```
training_job_arn=None

for execution_step in execution.list_steps():
    ### BEGIN SOLUTION - DO NOT delete this comment for grading purposes
    if execution_step['StepName'] == 'Train': # Replace all None
    ### END SOLUTION - DO NOT delete this comment for grading purposes
        training_job_arn = execution_step['Metadata']['TrainingJob']['Arn
        pprint(execution_step)
        break
print('Training job ARN: {}'.format(training_job_arn))

training_job_name = training_job_arn.split('/')[-1]
print('Training job Name: {}'.format(training_job_name))
```

```
{'AttemptCount': 0,
 'EndTime': datetime.datetime(2022, 9, 4, 7, 34, 44, 166000, tzinfo=tzloc
al()),
 'Metadata': {'TrainingJob': {'Arn': 'arn:aws:sagemaker:us-east-1:1418813
72941:training-job/pipelines-dh54d7we8u88-train-aawqvjpxen'}},
 'StartTime': datetime.datetime(2022, 9, 4, 7, 13, 41, 911000, tzinfo=tzl
ocal()),
 'StepName': 'Train',
 'StepStatus': 'Succeeded'}
Training job ARN: arn:aws:sagemaker:us-east-1:141881372941:training-job/p
ipelines-dh54d7we8u88-train-aawqvjpxen
Training job Name: pipelines-dh54d7we8u88-train-aawqvjpxen
```

Using similar approach you can find and print the pipeline artifacts.

In [53]:
```python
processing_job_name=None
training_job_name=None
```

In [54]:
```python
import time
from sagemaker.lineage.visualizer import LineageTableVisualizer

viz = LineageTableVisualizer(sagemaker.session.Session())

for execution_step in reversed(execution.list_steps()):
    pprint(execution_step)
    if execution_step['StepName'] == 'Processing':
        processing_job_name=execution_step['Metadata']['ProcessingJob']['
        print('Processing job name: {}'.format(processing_job_name))
        display(viz.show(processing_job_name=processing_job_name))
    elif execution_step['StepName'] == 'Train':
        training_job_name=execution_step['Metadata']['TrainingJob']['Arn'
        print('Training job name: {}'.format(training_job_name))
        display(viz.show(training_job_name=training_job_name))
    else:
        display(viz.show(pipeline_execution_step=execution_step))
        time.sleep(5)
```

```
{'AttemptCount': 0,
 'EndTime': datetime.datetime(2022, 9, 4, 7, 13, 41, 387000, tzinfo=tzloc
al()),
 'Metadata': {'ProcessingJob': {'Arn': 'arn:aws:sagemaker:us-east-1:14188
1372941:processing-job/pipelines-dh54d7we8u88-processing-o6skn85hoa'}},
 'StartTime': datetime.datetime(2022, 9, 4, 6, 59, 38, 623000, tzinfo=tzl
ocal()),
 'StepName': 'Processing',
 'StepStatus': 'Succeeded'}
Processing job name: pipelines-dh54d7we8u88-processing-o6skn85hoa
```

	Name/Source	Direction	Type	Association Type	Lineage Type
0	s3://...-06-59-36-833/input/code/prepare_data.py	Input	DataSet	ContributedTo	artifact
1	s3://dlai-practical-data-science/data/raw/	Input	DataSet	ContributedTo	artifact
2	68331...om/sagemaker-scikit-learn:0.23-1-cpu-py3	Input	Image	ContributedTo	artifact
3	s3://...09-04-06-59-33-489/output/sentiment-test	Output	DataSet	Produced	artifact
4	s3://...06-59-33-489/output/sentiment-validation	Output	DataSet	Produced	artifact
5	s3://...9-04-06-59-33-489/output/sentiment-train	Output	DataSet	Produced	artifact

```
{'AttemptCount': 0,
 'EndTime': datetime.datetime(2022, 9, 4, 7, 34, 44, 166000, tzinfo=tzloc
al()),
 'Metadata': {'TrainingJob': {'Arn': 'arn:aws:sagemaker:us-east-1:1418813
72941:training-job/pipelines-dh54d7we8u88-train-aawqvjpxen'}},
 'StartTime': datetime.datetime(2022, 9, 4, 7, 13, 41, 911000, tzinfo=tzl
ocal()),
 'StepName': 'Train',
 'StepStatus': 'Succeeded'}
Training job name: pipelines-dh54d7we8u88-train-aawqvjpxen
```

	Name/Source	Direction	Type	Association Type	Lineage Type
0	s3://...06-59-33-489/output/sentiment-validation	Input	DataSet	ContributedTo	artifact
1	s3://...9-04-06-59-33-489/output/sentiment-train	Input	DataSet	ContributedTo	artifact
2	76310...onaws.com/pytorch-training:1.6.0-cpu-py3	Input	Image	ContributedTo	artifact
3	s3://...u88-Train-AAWQVJPxeN/output/model.tar.gz	Output	Model	Produced	artifact

```
{'AttemptCount': 0,
 'EndTime': datetime.datetime(2022, 9, 4, 7, 43, 13, 700000, tzinfo=tzloc
al()),
 'Metadata': {'ProcessingJob': {'Arn': 'arn:aws:sagemaker:us-east-1:14188
1372941:processing-job/pipelines-dh54d7we8u88-evaluatemodel-qkca7pf6qk'}}
,
 'StartTime': datetime.datetime(2022, 9, 4, 7, 34, 45, 183000, tzinfo=tzl
ocal()),
 'StepName': 'EvaluateModel',
 'StepStatus': 'Succeeded'}
```

	Name/Source	Direction	Type	Association Type	Lineag Typ
0	s3://...065/input/code/evaluate_model_metrics.py	Input	DataSet	ContributedTo	artifac
1	s3://...09-04-06-59-33-489/output/sentiment-test	Input	DataSet	ContributedTo	artifac
2	s3://...u88-Train-AAWQVJPxeN/output/model.tar.gz	Input	Model	ContributedTo	artifac
3	68331...om/sagemaker-scikit-learn:0.23-1-cpu-py3	Input	Image	ContributedTo	artifac
4	s3://...n-2022-09-04-06-59-35-126/output/metrics	Output	DataSet	Produced	artifac

```
{'AttemptCount': 0,
 'EndTime': datetime.datetime(2022, 9, 4, 7, 43, 14, 577000, tzinfo=tzloc
al()),
 'Metadata': {'Condition': {'Outcome': 'True'}},
 'StartTime': datetime.datetime(2022, 9, 4, 7, 43, 14, 233000, tzinfo=tzl
ocal()),
 'StepName': 'AccuracyCondition',
 'StepStatus': 'Succeeded'}
None
{'AttemptCount': 0,
 'EndTime': datetime.datetime(2022, 9, 4, 7, 43, 16, 157000, tzinfo=tzloc
al()),
 'Metadata': {'Model': {'Arn': 'arn:aws:sagemaker:us-east-1:141881372941:
model/pipelines-dh54d7we8u88-createmodel-681vuhw2fy'}},
 'StartTime': datetime.datetime(2022, 9, 4, 7, 43, 14, 949000, tzinfo=tzl
ocal()),
 'StepName': 'CreateModel',
 'StepStatus': 'Succeeded'}
None
{'AttemptCount': 0,
 'EndTime': datetime.datetime(2022, 9, 4, 7, 43, 16, 2000, tzinfo=tzlocal
()),
 'Metadata': {'RegisterModel': {'Arn': 'arn:aws:sagemaker:us-east-1:14188
1372941:model-package/bert-reviews-1662274772/1'}},
 'StartTime': datetime.datetime(2022, 9, 4, 7, 43, 14, 949000, tzinfo=tzl
ocal()),
 'StepName': 'RegisterModel',
 'StepStatus': 'Succeeded'}
```

	Name/Source	Direction	Type	Association Type	Lineage Type
0	s3://...u88-Train-AAWQVJPxeN/output/model.tar.gz	Input	Model	ContributedTo	artifact
1	76310...aws.com/pytorch-inference:1.6.0-cpu-py36	Input	Image	ContributedTo	artifact
2	bert-reviews-1662274772-1-PendingManualApprova...	Input	Approval	ContributedTo	action
3	BERT-Reviews-1662274772-1662277395-aws-model-p...	Output	ModelGroup	AssociatedWith	context

9. Deploy and test the model

9.1. Approve trained model

The pipeline created a model package version within the specified model package group and an approval status of `PendingManualApproval`. This requires a separate step to manually approve the model before deploying to production.

You can approve the model using the SageMaker Studio UI or programmatically as shown below.

Get the model package ARN.

In [55]:

```
for execution_step in execution.list_steps():
    if execution_step['StepName'] == 'RegisterModel':
        model_package_arn = execution_step['Metadata']['RegisterModel']['
        break
print(model_package_arn)
```

```
arn:aws:sagemaker:us-east-1:141881372941:model-package/bert-reviews-16622
74772/1
```

Update the model package with the `Approved` status to prepare for deployment.

The model must be `Approved` before it can be deployed.

In [56]:

```
model_package_update_response = sm.update_model_package(
    ModelPackageArn=model_package_arn,
    ModelApprovalStatus="Approved",
)

pprint(model_package_update_response)
```

```
{'ModelPackageArn': 'arn:aws:sagemaker:us-east-1:141881372941:model-packa
ge/bert-reviews-1662274772/1',
 'ResponseMetadata': {'HTTPHeaders': {'content-length': '102',
                                      'content-type': 'application/x-amz-
json-1.1',
                                      'date': 'Sun, 04 Sep 2022 07:47:51
GMT',
                                      'x-amzn-requestid': 'e55ca522-cf2d-
4b91-8b37-0cda586d1231'},
                      'HTTPStatusCode': 200,
                      'RequestId': 'e55ca522-cf2d-4b91-8b37-0cda586d1231'
,
                      'RetryAttempts': 0}}
```

9.2. Deploy model

Get the model ARN and the model name from it.

In [57]:
```
for execution_step in execution.list_steps():
    print(execution_step['StepName'])
    if execution_step['StepName'] == 'CreateModel':
        model_arn = execution_step['Metadata']['Model']['Arn']
        break
print(model_arn)

model_name = model_arn.split('/')[-1]
print(model_name)
```

```
RegisterModel
CreateModel
arn:aws:sagemaker:us-east-1:141881372941:model/pipelines-dh54d7we8u88-cre
atemodel-681vuhw2fy
pipelines-dh54d7we8u88-createmodel-681vuhw2fy
```

9.3. Create endpoint from registry

Configure the endpoint.

In [58]:
```
endpoint_config_name = 'bert-model-epc-{}'.format(timestamp)
print(endpoint_config_name)

create_endpoint_config_response = sm.create_endpoint_config(
    EndpointConfigName = endpoint_config_name,
    ProductionVariants=[{
        'InstanceType':'ml.m5.xlarge',
        'InitialVariantWeight':1,
        'InitialInstanceCount':1,
        'ModelName': model_name,
        'VariantName':'AllTraffic'}])
```

```
bert-model-epc-1662274772
```

Create the endpoint.

In [59]:
```
pipeline_endpoint_name = 'bert-model-ep-{}'.format(timestamp)
print("EndpointName={}".format(pipeline_endpoint_name))

create_endpoint_response = sm.create_endpoint(
    EndpointName=pipeline_endpoint_name,
    EndpointConfigName=endpoint_config_name)
print(create_endpoint_response['EndpointArn'])
```

```
EndpointName=bert-model-ep-1662274772
arn:aws:sagemaker:us-east-1:141881372941:endpoint/bert-model-ep-166227477
2
```

In [60]:
```
from IPython.core.display import display, HTML

display(HTML('<b>Review <a target="blank" href="https://console.aws.amazo
```

Review SageMaker REST Endpoint

Wait until the endpoint is deployed.

This cell will take approximately 5-10 minutes to run.

In [61]:
```
%%time

while True:
    try:
        waiter = sm.get_waiter('endpoint_in_service')
        print('Waiting for endpoint to be in `InService`...')
        waiter.wait(EndpointName=pipeline_endpoint_name)
        break;
    except:
        print('Waiting for endpoint...')
        endpoint_status = sm.describe_endpoint(EndpointName=pipeline_endp
        print('Endpoint status: {}'.format(endpoint_status))
        if endpoint_status == 'Failed':
            break
        time.sleep(30)

print('Endpoint deployed.')
```

```
Waiting for endpoint to be in `InService`...
Endpoint deployed.
CPU times: user 119 ms, sys: 10 ms, total: 129 ms
Wall time: 4min 1s
```

Wait until the endpoint ^^ above ^^ is deployed.

9.4. Test model

Predict the `sentiment` with `review_body` samples and review the result:

In [62]:
```
from sagemaker.predictor import Predictor
from sagemaker.serializers import JSONLinesSerializer
from sagemaker.deserializers import JSONLinesDeserializer

inputs = [
    {"features": ["I love this product!"]},
    {"features": ["OK, but not great."]},
    {"features": ["This is not the right product."]},
]

predictor = Predictor(
    endpoint_name=pipeline_endpoint_name,
    serializer=JSONLinesSerializer(),
    deserializer=JSONLinesDeserializer(),
    sagemaker_session=sess
)

predicted_classes = predictor.predict(inputs)

for predicted_class in predicted_classes:
    print("Predicted class {} with probability {}".format(predicted_class
```

```
Predicted class 1 with probability 0.9341263771057129
Predicted class 0 with probability 0.5311968922615051
Predicted class -1 with probability 0.7742885947227478
```

9.5. SageMaker Studio extensions

SageMaker Studio provides a rich set of features to visually inspect SageMaker resources including pipelines, training jobs, and endpoints. Please take time to explore it opening the facet shown in the following image.

Congratulations! You have just deployed an end-to-end pipeline with BERT and SageMaker Pipelines.

Upload the notebook into S3 bucket for grading purposes.

Note: you may need to click on "Save" button before the upload.

```
In [63]: !aws s3 cp ./C2_W3_Assignment.ipynb s3://$bucket/C2_W3_Assignment_Learner
```

```
upload: ./C2_W3_Assignment.ipynb to s3://sagemaker-us-east-1-141881372941
/C2_W3_Assignment_Learner.ipynb
```

Please go to the main lab window and click on `Submit` button (see the `Finish the lab` section of the instructions).

```
In [ ]:
```