Amazon SageMaker Model Quality Monitor

This notebook's CI test result for us-west-2 is as follows. CI test results in other regions can be found at the end of the notebook.

 This us-west-2 badge failed to load. Check your device's internet connectivity, otherwise the service is currently unavailable

---

Section 1 - Setup

In this section, you will import the necessary libraries, setup variables and examine data that was used to train the XGBoost customer churn model provided with this notebook.

Let's start by specifying:

- The AWS region used to host your model.
- The IAM role associated with this SageMaker notebook instance.
- The S3 bucket used to store the data used to train your model, any additional model data, and the data captured from model invocations.

1.1 Import necessary libraries

In [91]:
```
%%time

from datetime import datetime, timedelta, timezone
import json
import os
import re
import boto3
from time import sleep
from threading import Thread

import pandas as pd

from sagemaker import get_execution_role, session, Session, image_uris
from sagemaker.s3 import S3Downloader, S3Uploader
from sagemaker.processing import ProcessingJob
from sagemaker.serializers import CSVSerializer

from sagemaker.model import Model
from sagemaker.model_monitor import DataCaptureConfig

session = Session()
```

```
CPU times: user 125 ms, sys: 35.1 ms, total: 160 ms
Wall time: 176 ms
```

1.2 AWS region and IAM Role

In [92]:
```
# Get Execution role
role = get_execution_role()
print("RoleArn:", role)

region = session.boto_region_name
print("Region:", region)
```

```
RoleArn: arn:aws:iam::672518276407:role/LabRole
Region: us-east-1
```

1.3 S3 bucket and prefixes

In [93]:
```
# Setup S3 bucket
# You can use a different bucket, but make sure the role you chose for this n
# has the s3:PutObject permissions. This is the bucket into which the data i
bucket = session.default_bucket()
print("Demo Bucket:", bucket)
prefix = "sagemaker/Churn-ModelQualityMonitor-20201201"

##S3 prefixes
data_capture_prefix = f"{prefix}/datacapture"
s3_capture_upload_path = f"s3://{bucket}/{data_capture_prefix}"

ground_truth_upload_path = (
    f"s3://{bucket}/{prefix}/ground_truth_data/{datetime.now():%Y-%m-%d-%H-%M
)

reports_prefix = f"{prefix}/reports"
s3_report_path = f"s3://{bucket}/{reports_prefix}"

##Get the model monitor image
monitor_image_uri = image_uris.retrieve(framework="model-monitor", region=re

print("Image URI:", monitor_image_uri)
print(f"Capture path: {s3_capture_upload_path}")
print(f"Ground truth path: {ground_truth_upload_path}")
print(f"Report path: {s3_report_path}")
```

```
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
Demo Bucket: sagemaker-us-east-1-672518276407
Image URI: 156813124566.dkr.ecr.us-east-1.amazonaws.com/sagemaker-model-monit
or-analyzer
Capture path: s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQual
ityMonitor-20201201/datacapture
Ground truth path: s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-Mode
lQualityMonitor-20201201/ground_truth_data/2025-06-07-20-05-31
Report path: s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQuali
tyMonitor-20201201/reports
```

1.4 Test access to the S3 bucket

Let's quickly verify that the notebook has the right permissions to access the S3 bucket specified above. Upload a simple test object into the S3 bucket. If this command fails, the data capture and model monitoring capabilities will not work from this notebook. You can fix this by updating the role associated with this notebook instance to have "s3:PutObject" permissions and try this validation again

In [94]:
```
# Upload some test files
S3Uploader.upload("test_data/upload-test-file.txt", f"s3://{bucket}/test_uplo
print("Success! You are all set to proceed.")
```

```
Success! You are all set to proceed.
```

Section 2 - Deploy pre-trained model with data capture enabled

In this section, you will upload the pretrained model to the S3 bucket, create an Amazon SageMaker Model, create an Amazon SageMaker real time endpoint, and enable data capture on the endpoint to capture endpoint invocations, predictions, and metadata.

2.1 Upload the pre-trained model to S3

This code uploads a pre-trained XGBoost model that is ready for you to deploy. This model was trained using the XGB Churn Prediction Notebook in SageMaker. You can also use your own pre-trained model in this step. If you already have a pretrained model in Amazon S3, you can add it instead by specifying the s3_key.

In [95]:
```
##Upload the pretrained model to S3
s3_key = f"s3://{bucket}/{prefix}"
model_url = S3Uploader.upload("model/xgb-churn-prediction-model.tar.gz", s3_
model_url
```

Out[95]:
```
's3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualityMonitor-
20201201/xgb-churn-prediction-model.tar.gz'
```

2.2 Create SageMaker Model entity

This step creates an Amazon SageMaker model from the model file uploaded to S3.

In [96]:
```
model_name = f"DEMO-xgb-churn-pred-model-monitor-{datetime.utcnow():%Y-%m-%d

image_uri = image_uris.retrieve(framework="xgboost", version="0.90-1", regio

model = Model(image_uri=image_uri, model_data=model_url, role=role, sagemake
```

```
/tmp/ipykernel_226/953180226.py:1: DeprecationWarning: datetime.datetime.utcn
ow() is deprecated and scheduled for removal in a future version. Use timezon
e-aware objects to represent datetimes in UTC: datetime.datetime.now(datetim
e.UTC).
  model_name = f"DEMO-xgb-churn-pred-model-monitor-{datetime.utcnow():%Y-%m-%
d-%H%M}"
INFO:sagemaker.image_uris:Defaulting to only available Python version: py3
INFO:sagemaker.image_uris:Defaulting to only supported image scope: cpu.
```

2.3 Deploy the model with data capture enabled.

Next, deploy the SageMaker model on a specific instance with data capture enabled.

In [97]:
```
endpoint_name = f"DEMO-xgb-churn-model-quality-monitor-{datetime.utcnow():%Y
print("EndpointName =", endpoint_name)

data_capture_config = DataCaptureConfig(
    enable_capture=True, sampling_percentage=100, destination_s3_uri=s3_capt
)

model.deploy(
    initial_instance_count=1,
    instance_type="ml.m5.xlarge",
    endpoint_name=endpoint_name,
    data_capture_config=data_capture_config,
)
```

```
/tmp/ipykernel_226/1694334827.py:1: DeprecationWarning: datetime.datetime.utc
now() is deprecated and scheduled for removal in a future version. Use timezo
ne-aware objects to represent datetimes in UTC: datetime.datetime.now(datetim
e.UTC).
  endpoint_name = f"DEMO-xgb-churn-model-quality-monitor-{datetime.utcnow():%
Y-%m-%d-%H%M}"
INFO:sagemaker:Creating model with name: sagemaker-xgboost-2025-06-07-20-05-3
3-043
EndpointName = DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005
INFO:sagemaker:Creating endpoint-config with name DEMO-xgb-churn-model-quality
-monitor-2025-06-07-2005
INFO:sagemaker:Creating endpoint with name DEMO-xgb-churn-model-quality-monit
or-2025-06-07-2005
-----!
```

2.4 Create the SageMaker Predictor object from the endpoint to be used for invoking the model

In [98]:
```
from sagemaker.predictor import Predictor

predictor = Predictor(
    endpoint_name=endpoint_name, sagemaker_session=session, serializer=CSVSe
)
```

Section 3 - Generate a baseline for model quality performance

In this section, you will invoke the endpoint created above using validation data. Predictions from the deployed model using this validation data will be used as a baseline dataset. You will then use SageMaker's Model Monitoring to execute a baseline job that computes model performance data, and suggest model quality constraints based on the baseline dataset.

3.1 Execute predictions using the validation dataset.

The deployed model returns probability that a customer will churn. Let's choose an arbitrary 0.8 cutoff to consider that a customer will churn.

In [99]:
```
churn_cutoff = 0.8
validate_dataset = "validation_with_predictions.csv"
```

In [100…
```
limit = 200  # Need at least 200 samples to compute standard deviations
i = 0
with open(f"test_data/{validate_dataset}", "w") as baseline_file:
    baseline_file.write("probability,prediction,label\n")  # our header
    with open("test_data/validation.csv", "r") as f:
        for row in f:
            (label, input_cols) = row.split(",", 1)
            probability = float(predictor.predict(input_cols))
            prediction = "1" if probability > churn_cutoff else "0"
            baseline_file.write(f"{probability},{prediction},{label}\n")
            i += 1
            if i > limit:
                break
            print(".", end="", flush=True)
            sleep(0.5)
print()
print("Done!")
```

```
..................................................................................
..................................................................................
.........................................
Done!
```

3.2 Examine the predictions from the model

In [101…
```
!head test_data/validation_with_predictions.csv
```

```
probability,prediction,label
0.01516005303710699,0,0
0.1684480607509613,0,0
0.21427156031131744,0,0
0.06330718100070953,0,0
0.02791607193648815,0,0
0.014169521629810333,0,0
0.00571369007229805,0,0
0.10534518957138062,0,0
0.025899196043610573,0,0
```

3.3 Upload the predictions as a baseline dataset.

Now we will upload the predictions made using validation dataset to S3 which will be used for creating model quality baseline statistics and constraints

In [102…
```
baseline_prefix = prefix + "/baselining"
baseline_data_prefix = baseline_prefix + "/data"
baseline_results_prefix = baseline_prefix + "/results"

baseline_data_uri = f"s3://{bucket}/{baseline_data_prefix}"
baseline_results_uri = f"s3://{bucket}/{baseline_results_prefix}"
print(f"Baseline data uri: {baseline_data_uri}")
print(f"Baseline results uri: {baseline_results_uri}")
```

```
Baseline data uri: s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-Mode
lQualityMonitor-20201201/baselining/data
Baseline results uri: s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-M
odelQualityMonitor-20201201/baselining/results
```

In [103…
```
baseline_dataset_uri = S3Uploader.upload(f"test_data/{validate_dataset}", ba
baseline_dataset_uri
```

Out[103…
```
's3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualityMonitor-
20201201/baselining/data/validation_with_predictions.csv'
```

3.4 Create a baselining job with validation dataset predictions

Define the model quality monitoring object and execute the model quality monitoring baseline job. Model monitor will automatically generate baseline statistics and constraints based on the validation dataset provided.

In [104…
```
from sagemaker.model_monitor import ModelQualityMonitor
from sagemaker.model_monitor import EndpointInput
from sagemaker.model_monitor.dataset_format import DatasetFormat
```

In [105…
```
# Create the model quality monitoring object
churn_model_quality_monitor = ModelQualityMonitor(
    role=role,
    instance_count=1,
    instance_type="ml.m5.xlarge",
    volume_size_in_gb=20,
    max_runtime_in_seconds=1800,
    sagemaker_session=session,
)
```

```
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
```

In [106…
```
# Name of the model quality baseline job
baseline_job_name = f"DEMO-xgb-churn-model-baseline-job-{datetime.utcnow():%
```

```
/tmp/ipykernel_226/478642366.py:2: DeprecationWarning: datetime.datetime.utcn
ow() is deprecated and scheduled for removal in a future version. Use timezon
e-aware objects to represent datetimes in UTC: datetime.datetime.now(datetim
e.UTC).
  baseline_job_name = f"DEMO-xgb-churn-model-baseline-job-{datetime.utcnow
():%Y-%m-%d-%H%M}"
```

In [107…
```
# Execute the baseline suggestion job.
# You will specify problem type, in this case Binary Classification, and prov
job = churn_model_quality_monitor.suggest_baseline(
    job_name=baseline_job_name,
    baseline_dataset=baseline_dataset_uri,
    dataset_format=DatasetFormat.csv(header=True),
    output_s3_uri=baseline_results_uri,
    problem_type="BinaryClassification",
    inference_attribute="prediction",
    probability_attribute="probability",
    ground_truth_attribute="label",
)
job.wait(logs=False)
```

```
INFO:sagemaker:Creating processing-job with name DEMO-xgb-churn-model-baselin
e-job-2025-06-07-2010
.............................................................!
```

3.5 Explore the results of the baselining job

You could see the baseline constraints and statistics files are uploaded to the S3 location.

In [108…
```
baseline_job = churn_model_quality_monitor.latest_baselining_job
```

3.5.1 View the metrics generated

You could see that the baseline statistics and constraints files are already uploaded to S3.

In [109…
```
binary_metrics = baseline_job.baseline_statistics().body_dict["binary_classi
pd.json_normalize(binary_metrics).T
```

Out[109…

	0
confusion_matrix.0.0	173
confusion_matrix.0.1	0
confusion_matrix.1.0	12
confusion_matrix.1.1	16
recall.value	0.571429
recall.standard_deviation	0.05573
precision.value	1.0
precision.standard_deviation	0.0
accuracy.value	0.940299
accuracy.standard_deviation	0.009702
recall_best_constant_classifier.value	0.0
recall_best_constant_classifier.standard_deviation	0.0
precision_best_constant_classifier.value	0.0
precision_best_constant_classifier.standard_deviation	0.0
accuracy_best_constant_classifier.value	0.860697
accuracy_best_constant_classifier.standard_deviation	0.011526
true_positive_rate.value	0.571429
true_positive_rate.standard_deviation	0.05573
true_negative_rate.value	1.0
true_negative_rate.standard_deviation	0.0
false_positive_rate.value	0.0
false_positive_rate.standard_deviation	0.0
false_negative_rate.value	0.428571
false_negative_rate.standard_deviation	0.05573
receiver_operating_characteristic_curve.false_positive_rates	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.005780346820809248, 0.005780346820809248, 0.005780346820809248, 0.011560693641618497, 0.017341040462427744, 0.017341040462427744, 0.017341040462427744, 0.017341040462427744, 0.023121387283236993, 0.028901734104046242, 0.03468208092485549,

0
0.04046242774566474,
0.046242774566473986,
0.05202312138728324,
0.05202312138728324,
0.057803468208092484,
0.06358381502890173,
0.06358381502890173,
0.06936416184971098,
0.07514450867052024,
0.08092485549132948,
0.08670520231213873,
0.09248554913294797,
0.09826589595375723,
0.10404624277456648,
0.10982658959537572,
0.11560693641618497,
0.12138728323699421,
0.12716763005780346,
0.1329479768786127,
0.13872832369942195,
0.14450867052023122,
0.15028901734104047,
0.15606936416184972,
0.16184971098265896,
0.1676300578034682,
0.17341040462427745,
0.17341040462427745,
0.1791907514450867,
0.18497109826589594,
0.1907514450867052,
0.19653179190751446,
0.2023121387283237,
0.20809248554913296,
0.2138728323699422,
0.21965317919075145,
0.2254335260115607,
0.23121387283236994,
0.23699421965317918,
0.24277456647398843,
0.24855491329479767,
0.2543352601156069,
0.26011560693641617,
0.2658959537572254,
0.27167630057803466,
0.2774566473988439,
0.2774566473988439,
0.2832369942196532,
0.28901734104046245,
0.2947976878612717,
0.30057803468208094,
0.3063583815028902,
0.31213872832369943,
0.3179190751445087,
0.3236994219653179,
0.32947976878612717,
0.3352601156069364,
0.34104046242774566,
0.3468208092485549,

	0
	0.35260115606936415, 0.3583815028901734, 0.36416184971098264, 0.3699421965317919, 0.37572254335260113, 0.3815028901734104, 0.3872832369942196, 0.3930635838150289, 0.3988439306358382, 0.4046242774566474, 0.4046242774566474, 0.41040462427745666, 0.4161849710982659, ...]
receiver_operating_characteristic_curve.true_positive_rates	[0.0, 0.03571428571428571, 0.07142857142857142, 0.10714285714285714, 0.14285714285714285, 0.17857142857142858, 0.21428571428571427, 0.25, 0.2857142857142857, 0.32142857142857145, 0.35714285714285715, 0.39285714285714285, 0.42857142857142855, 0.4642857142857143, 0.5, 0.5357142857142857, 0.5714285714285714, 0.6071428571428571, 0.6071428571428571, 0.6428571428571429, 0.6785714285714286, 0.6785714285714286, 0.6785714285714286, 0.7142857142857143, 0.75, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.8214285714285714, 0.8214285714285714, 0.8214285714285714, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571,

	0
	0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.9285714285714286, 0.9642857142857143, 0.9642857142857143, 0.9642857142857143, ...]
precision_recall_curve.precisions	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.9444444444444444, 0.9473684210526315, 0.95,

0
0.9047619047619048, 0.8636363636363636, 0.8695652173913043, 0.875, 0.88, 0.8461538461538461, 0.8148148148148148, 0.7857142857142857, 0.7586206896551724, 0.7333333333333333, 0.7096774193548387, 0.71875, 0.696969696969697, 0.6764705882352942, 0.6857142857142857, 0.6666666666666666, 0.6486486486486487, 0.631578947368421, 0.6153846153846154, 0.6, 0.5853658536585366, 0.5714285714285714, 0.5581395348837209, 0.5454545454545454, 0.5333333333333333, 0.5217391304347826, 0.5106382978723404, 0.5, 0.4897959183673469, 0.48, 0.47058823529411764, 0.46153846153846156, 0.4528301886792453, 0.4444444444444444, 0.45454545454545453, 0.44642857142857145, 0.43859649122807015, 0.43103448275862066, 0.423728813559322, 0.4166666666666667, 0.4098360655737705, 0.4032258064516129, 0.3968253968253968, 0.390625, 0.38461538461538464, 0.3787878787878788, 0.373134328358209, 0.36764705882352944, 0.36231884057971014, 0.35714285714285715, 0.352112676056338, 0.3472222222222222, 0.3424657534246575, 0.35135135135135137, 0.3466666666666667, 0.34210526315789475, 0.33766233766233766, 0.3333333333333333, 0.3291139240506329, 0.325, 0.32098765432098764, 0.3170731707317073, 0.3132530120481928, 0.30952380952380953,

	0
	0.3058823529411765, 0.3023255813953488, 0.2988505747126437, 0.29545454545454547, 0.29213483146067415, 0.28888888888888886, 0.2857142857142857, 0.2826086956521739, 0.27956989247311825, 0.2765957446808511, 0.2736842105263158, 0.2708333333333333, 0.27835051546391754, 0.2755102040816326, 0.2727272727272727, ...]
precision_recall_curve.recalls	[0.0, 0.03571428571428571, 0.07142857142857142, 0.10714285714285714, 0.14285714285714285, 0.17857142857142858, 0.21428571428571427, 0.25, 0.2857142857142857, 0.32142857142857145, 0.35714285714285715, 0.39285714285714285, 0.42857142857142855, 0.4642857142857143, 0.5, 0.5357142857142857, 0.5714285714285714, 0.6071428571428571, 0.6071428571428571, 0.6428571428571429, 0.6785714285714286, 0.6785714285714286, 0.6785714285714286, 0.7142857142857143, 0.75, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.7857142857142857, 0.8214285714285714, 0.8214285714285714, 0.8214285714285714, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571,

	0
	0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8571428571428571, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.8928571428571429, 0.9285714285714286, 0.9642857142857143, 0.9642857142857143, 0.9642857142857143, ...]
auc.value	0.939513
auc.standard_deviation	0.010579

	0
au_prc.value	0.863272
au_prc.standard_deviation	0.014946
f0_5.value	0.869565
f0_5.standard_deviation	0.024992
f1.value	0.727273
f1.standard_deviation	0.044285
f2.value	0.625
f2.standard_deviation	0.052957
f0_5_best_constant_classifier.value	0.0
f0_5_best_constant_classifier.standard_deviation	0.0
f1_best_constant_classifier.value	0.0
f1_best_constant_classifier.standard_deviation	0.0
f2_best_constant_classifier.value	0.0
f2_best_constant_classifier.standard_deviation	0.0

3.5.2 View the constraints generated

In [110…
```
pd.DataFrame(baseline_job.suggested_constraints().body_dict["binary_classific
```

Out[110…

	threshold	comparison_operator
recall	0.571429	LessThanThreshold
precision	1.0	LessThanThreshold
accuracy	0.940299	LessThanThreshold
true_positive_rate	0.571429	LessThanThreshold
true_negative_rate	1.0	LessThanThreshold
false_positive_rate	0.0	GreaterThanThreshold
false_negative_rate	0.428571	GreaterThanThreshold
auc	0.939513	LessThanThreshold
f0_5	0.869565	LessThanThreshold
f1	0.727273	LessThanThreshold
f2	0.625	LessThanThreshold

In the above example you can see that model quality monitor suggested a constraint that will ensure that the model F2 score should note drop below 0.625. Few generated constraints

may be a tad aggressive like precision, where it will alert on any drops below 1.0. It is recommended to modify this file as necessary prior to using for monitoring.

Section 4 - Setup continuous model monitoring to identify model quality drift

In this section, you will setup a continuous model monitoring job that monitors the quality of the deployed model against the baseline generated in the previous section. This is to ensure that the quality does not degrade over time.

In addition to the generated baseline, Amazon SageMaker Model Quality Monitoring needs two additional inputs - predictions made by the deployed model endpoint and the ground truth data to be provided by the model consuming application. Since you already enabled data capture on the endpoint, prediction data is captured in S3. The ground truth data depends on the what your model is predicting and what the business use case is. In this case, since the model is predicting customer churn, ground truth data may indicate if the customer actually left the company or not. For the purposes of this notebook, you will generate synthetic data as ground truth.

4.1 Generate prediction data for Model Quality Monitoring

Start generating some artificial traffic. The cell below starts a thread to send some traffic to the endpoint. Note that you need to stop the kernel to terminate this thread. If there is no traffic, the monitoring jobs are marked as `Failed` since there is no data to process.

In [111…

```
def invoke_endpoint(ep_name, file_name):
    with open(file_name, "r") as f:
        i = 0
        for row in f:
            payload = row.rstrip("\n")
            response = session.sagemaker_runtime_client.invoke_endpoint(
                EndpointName=endpoint_name,
                ContentType="text/csv",
                Body=payload,
                InferenceId=str(i),  # unique ID per row
            )["Body"].read()
            i += 1
            sleep(1)


def invoke_endpoint_forever():
    while True:
        try:
            invoke_endpoint(endpoint_name, "test_data/test-dataset-input-col
        except session.sagemaker_runtime_client.exceptions.ValidationError:
            pass
```

```
thread = Thread(target=invoke_endpoint_forever)
thread.start()
```

Notice the new attribute `inferenceId`, which we're setting when invoking the endpoint. This is used to join the prediction data with the ground truth data.

4.2 View captured data

Now list the data capture files stored in Amazon S3. You should expect to see different files from different time periods organized based on the hour in which the invocation occurred. The format of the Amazon S3 path is:

`s3://{destination-bucket-prefix}/{endpoint-name}/{variant-name}/yyyy/mm/dd/hh/filename.jsonl`

In [112…
```
print("Waiting for captures to show up", end="")
for _ in range(120):
    capture_files = sorted(S3Downloader.list(f"{s3_capture_upload_path}/{endp
    if capture_files:
        capture_file = S3Downloader.read_file(capture_files[-1]).split("\n")
        capture_record = json.loads(capture_file[0])
        if "inferenceId" in capture_record["eventMetadata"]:
            break
    print(".", end="", flush=True)
    sleep(1)
print()
print("Found Capture Files:")
print("\n ".join(capture_files[-3:]))
```

```
Waiting for captures to show up
Found Capture Files:
s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualityMonitor-202
01201/datacapture/DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005/AllTra
ffic/2025/06/07/20/11-27-913-2722bf7b-f673-45bc-9270-4c848069d8ba.jsonl
 s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualityMonitor-20
201201/datacapture/DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005/AllTr
affic/2025/06/07/20/12-28-617-838f0788-4122-430f-b109-e40ff8a82291.jsonl
 s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualityMonitor-20
201201/datacapture/DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005/AllTr
affic/2025/06/07/20/13-29-268-5ab7aa2a-55d7-47a3-890e-97c442f846d4.jsonl
```

Next, view the contents of a single capture file. Here you should see all the data captured in an Amazon SageMaker specific JSON-line formatted file. Take a quick peek at the first few lines in the captured file.

In [113…
```
print("\n".join(capture_file[-3:-1]))
```

```
{"captureData":{"endpointInput":{"observedContentType":"text/csv","mode":"INP
UT","data":"126,0,122.4,88,143.8,111,157.0,106,11.5,3,1,0,0,0,0,0,0,0,1,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,1,1,0,1,0","encoding":"CSV"},"endpointOutput":{"observedContentTyp
e":"text/csv; charset=utf-8","mode":"OUTPUT","data":"0.02458457089960575","en
coding":"CSV"}},"eventMetadata":{"eventId":"2b0588d8-0bee-4b84-8191-404b0671c
195","inferenceId":"23","inferenceTime":"2025-06-07T20:14:27Z"},"eventVersio
n":"0"}
{"captureData":{"endpointInput":{"observedContentType":"text/csv","mode":"INP
UT","data":"108,15,165.1,85,267.0,93,250.7,114,10.9,4,1,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,
0,0,0,1,0,0,1,0,0,1","encoding":"CSV"},"endpointOutput":{"observedContentTyp
e":"text/csv; charset=utf-8","mode":"OUTPUT","data":"0.029797600582242012","e
ncoding":"CSV"}},"eventMetadata":{"eventId":"c6bfcda9-cceb-40e0-aa52-99fe8eef
a106","inferenceId":"24","inferenceTime":"2025-06-07T20:14:28Z"},"eventVersio
n":"0"}
```


Finally, the contents of a single line is present below in a formatted JSON file so that you can observe a little better.

Again, notice the `inferenceId` attribute that is set as part of the invoke_endpoint call. If this is present, it will be used to join with ground truth data (otherwise `eventId` will be used):

In [114…

```python
print(json.dumps(capture_record, indent=2))
```


```
{
  "captureData": {
    "endpointInput": {
      "observedContentType": "text/csv",
      "mode": "INPUT",
      "data": "64,0,168.0,116,192.4,94,166.5,98,10.1,3,2,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,
0,0,0,0,1,0,1,0,1,0",
      "encoding": "CSV"
    },
    "endpointOutput": {
      "observedContentType": "text/csv; charset=utf-8",
      "mode": "OUTPUT",
      "data": "0.010930510237812996",
      "encoding": "CSV"
    }
  },
  "eventMetadata": {
    "eventId": "91bb7030-438b-4b4a-b85b-bf65a54a8adf",
    "inferenceId": "299",
    "inferenceTime": "2025-06-07T20:13:29Z"
  },
  "eventVersion": "0"
}
```


4.3 Generate synthetic ground truth

Next, start generating ground truth data. The model quality job will fail if there's no ground truth data to merge.

In [115…

```
import random


def ground_truth_with_id(inference_id):
    random.seed(inference_id)  # to get consistent results
    rand = random.random()
    return {
        "groundTruthData": {
            "data": "1" if rand < 0.7 else "0",  # randomly generate positiv
            "encoding": "CSV",
        },
        "eventMetadata": {
            "eventId": str(inference_id),
        },
        "eventVersion": "0",
    }


def upload_ground_truth(records, upload_time):
    fake_records = [json.dumps(r) for r in records]
    data_to_upload = "\n".join(fake_records)
    target_s3_uri = f"{ground_truth_upload_path}/{upload_time:%Y/%m/%d/%H/%M
    print(f"Uploading {len(fake_records)} records to", target_s3_uri)
    S3Uploader.upload_string_as_file_body(data_to_upload, target_s3_uri)
```

In [116…

```
NUM_GROUND_TRUTH_RECORDS = 334  # 334 are the number of rows in data we're s


def generate_fake_ground_truth_forever():
    j = 0
    while True:
        fake_records = [ground_truth_with_id(i) for i in range(NUM_GROUND_TR
        upload_ground_truth(fake_records, datetime.utcnow())
        j = (j + 1) % 5
        sleep(60 * 60)  # do this once an hour


gt_thread = Thread(target=generate_fake_ground_truth_forever)
gt_thread.start()
```

```
Uploading 334 records to s3://sagemaker-us-east-1-672518276407/sagemaker/Chur
n-ModelQualityMonitor-20201201/ground_truth_data/2025-06-07-20-05-31/2025/06/
07/20/1522.jsonl
```

```
/tmp/ipykernel_226/2345861737.py:8: DeprecationWarning: datetime.datetime.utc
now() is deprecated and scheduled for removal in a future version. Use timezo
ne-aware objects to represent datetimes in UTC: datetime.datetime.now(datetim
e.UTC).
  upload_ground_truth(fake_records, datetime.utcnow())
/tmp/ipykernel_226/2345861737.py:8: DeprecationWarning: datetime.datetime.utc
now() is deprecated and scheduled for removal in a future version. Use timezo
ne-aware objects to represent datetimes in UTC: datetime.datetime.now(datetim
e.UTC).
  upload_ground_truth(fake_records, datetime.utcnow())
Uploading 334 records to s3://sagemaker-us-east-1-672518276407/sagemaker/Chur
n-ModelQualityMonitor-20201201/ground_truth_data/2025-06-07-20-05-31/2025/06/
07/21/1523.jsonl
/tmp/ipykernel_226/2345861737.py:8: DeprecationWarning: datetime.datetime.utc
now() is deprecated and scheduled for removal in a future version. Use timezo
ne-aware objects to represent datetimes in UTC: datetime.datetime.now(datetim
e.UTC).
  upload_ground_truth(fake_records, datetime.utcnow())
Uploading 334 records to s3://sagemaker-us-east-1-672518276407/sagemaker/Chur
n-ModelQualityMonitor-20201201/ground_truth_data/2025-06-07-20-05-31/2025/06/
07/22/1523.jsonl
```

4.4 Create a monitoring schedule

Now that you have the baseline information and ground truth labels, create a monitoring schedule to run model quality monitoring job.

In [117…
```python
##Monitoring schedule name
churn_monitor_schedule_name = (
    f"DEMO-xgb-churn-monitoring-schedule-{datetime.utcnow():%Y-%m-%d-%H%M}"
)
```

```
/tmp/ipykernel_226/2544654367.py:3: DeprecationWarning: datetime.datetime.utc
now() is deprecated and scheduled for removal in a future version. Use timezo
ne-aware objects to represent datetimes in UTC: datetime.datetime.now(datetim
e.UTC).
  f"DEMO-xgb-churn-monitoring-schedule-{datetime.utcnow():%Y-%m-%d-%H%M}"
```

For the monitoring schedule you need to specify how to interpret an endpoint's output. Given that the endpoint in this notebook outputs CSV data, the below code specifies that the first column of the output, `0`, contains a probability (of churn in this example). You will further specify `0.5` as the cutoff used to determine a positive label (that is, predict that a customer will churn).

In [118…
```python
# Create an enpointInput
endpointInput = EndpointInput(
    endpoint_name=predictor.endpoint_name,
    probability_attribute="0",
    probability_threshold_attribute=0.5,
    destination="/opt/ml/processing/input_data",
)
```

In [119…
```
# Create the monitoring schedule to execute every hour.
from sagemaker.model_monitor import CronExpressionGenerator

response = churn_model_quality_monitor.create_monitoring_schedule(
    monitor_schedule_name=churn_monitor_schedule_name,
    endpoint_input=endpointInput,
    output_s3_uri=baseline_results_uri,
    problem_type="BinaryClassification",
    ground_truth_input=ground_truth_upload_path,
    constraints=baseline_job.suggested_constraints(),
    schedule_cron_expression=CronExpressionGenerator.hourly(),
    enable_cloudwatch_metrics=True,
)
```

```
INFO:sagemaker.model_monitor.model_monitoring:Creating Monitoring Schedule wi
th name: DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015
```

In [120…
```
# Create the monitoring schedule
# You will see the monitoring schedule in the 'Scheduled' status
churn_model_quality_monitor.describe_schedule()
```

Out[120…
```
{'MonitoringScheduleArn': 'arn:aws:sagemaker:us-east-1:672518276407:monitor
ing-schedule/DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015',
 'MonitoringScheduleName': 'DEMO-xgb-churn-monitoring-schedule-2025-06-07-2
015',
 'MonitoringScheduleStatus': 'Pending',
 'MonitoringType': 'ModelQuality',
 'CreationTime': datetime.datetime(2025, 6, 7, 20, 15, 23, 507000, tzinfo=t
zlocal()),
 'LastModifiedTime': datetime.datetime(2025, 6, 7, 20, 15, 23, 623000, tzinf
o=tzlocal()),
 'MonitoringScheduleConfig': {'ScheduleConfig': {'ScheduleExpression': 'cron
(0 * ? * * *)'},
  'MonitoringJobDefinitionName': 'model-quality-job-definition-2025-06-07-20-
15-22-805',
  'MonitoringType': 'ModelQuality'},
 'EndpointName': 'DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005',
 'ResponseMetadata': {'RequestId': '955c2ef6-49d6-45d9-81f0-37b41447491d',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '955c2ef6-49d6-45d9-81f0-37b41447491
d',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '630',
   'date': 'Sat, 07 Jun 2025 20:15:23 GMT'},
  'RetryAttempts': 0}}
```

4.5 Examine monitoring schedule executions

In [121…
```
# Initially there will be no executions since the first execution happens at
# Note that it is common for the execution to luanch upto 20 min after the h
executions = churn_model_quality_monitor.list_executions()
executions
```

```
WARNING:sagemaker.model_monitor.model_monitoring:No executions found for sche
dule. monitoring_schedule_name: DEMO-xgb-churn-monitoring-schedule-2025-06-07
-2015
```

Out[121… []

In [ ]:
```
# Wait for the first execution of the monitoring_schedule
print("Waiting for first execution", end="")
while True:
    execution = churn_model_quality_monitor.describe_schedule().get(
        "LastMonitoringExecutionSummary"
    )
    if execution:
        break
    print(".", end="", flush=True)
    sleep(10)
print()
print("Execution found!")
```

```
Waiting for first executio
n.....................................................................
.....................................................................
.....................................................
Execution found!
```

In [ ]:
```
while not executions:
    executions = churn_model_quality_monitor.list_executions()
    print(".", end="", flush=True)
    sleep(10)
latest_execution = executions[-1]
latest_execution.describe()
```

```
....
```

Out[ ]:

```
{'ProcessingInputs': [{'InputName': 'groundtruth_input_1',
   'AppManaged': False,
   'S3Input': {'S3Uri': 's3://sagemaker-us-east-1-672518276407/sagemaker/Ch
urn-ModelQualityMonitor-20201201/ground_truth_data/2025-06-07-20-05-31/202
5/06/07/20',
    'LocalPath': '/opt/ml/processing/groundtruth/2025/06/07/20',
    'S3DataType': 'S3Prefix',
    'S3InputMode': 'File',
    'S3DataDistributionType': 'FullyReplicated',
    'S3CompressionType': 'None'}},
  {'InputName': 'endpoint_input_1',
   'AppManaged': False,
   'S3Input': {'S3Uri': 's3://sagemaker-us-east-1-672518276407/sagemaker/Ch
urn-ModelQualityMonitor-20201201/datacapture/DEMO-xgb-churn-model-quality-m
onitor-2025-06-07-2005/AllTraffic/2025/06/07/20',
    'LocalPath': '/opt/ml/processing/input_data/DEMO-xgb-churn-model-qualit
y-monitor-2025-06-07-2005/AllTraffic/2025/06/07/20',
    'S3DataType': 'S3Prefix',
    'S3InputMode': 'File',
    'S3DataDistributionType': 'FullyReplicated',
    'S3CompressionType': 'None'}}],
 'ProcessingOutputConfig': {'Outputs': [{'OutputName': 'result',
    'S3Output': {'S3Uri': 's3://sagemaker-us-east-1-672518276407/sagemaker/
Churn-ModelQualityMonitor-20201201/baselining/results/merge',
     'LocalPath': '/opt/ml/processing/output',
     'S3UploadMode': 'EndOfJob'},
    'AppManaged': False}]},
 'ProcessingJobName': 'groundtruth-merge-202506072100-0b362d86ea6613c993bd8
1bf',
 'ProcessingResources': {'ClusterConfig': {'InstanceCount': 1,
   'InstanceType': 'ml.m5.xlarge',
   'VolumeSizeInGB': 20}},
 'StoppingCondition': {'MaxRuntimeInSeconds': 1800},
 'AppSpecification': {'ImageUri': '156813124566.dkr.ecr.us-east-1.amazonaws.
com/sagemaker-model-monitor-groundtruth-merger'},
 'Environment': {'dataset_format': '{"sagemakerCaptureJson":{"captureIndexN
ames":null}}',
  'dataset_source': '/opt/ml/processing/input_data',
  'end_time': '2025-06-07T21:00:00Z',
  'ground_truth_source': '/opt/ml/processing/groundtruth',
  'monitoring_input_type': 'ENDPOINT_INPUT',
  'output_path': '/opt/ml/processing/output',
  'start_time': '2025-06-07T20:00:00Z'},
 'RoleArn': 'arn:aws:iam::672518276407:role/LabRole',
 'ProcessingJobArn': 'arn:aws:sagemaker:us-east-1:672518276407:processing-j
ob/groundtruth-merge-202506072100-0b362d86ea6613c993bd81bf',
 'ProcessingJobStatus': 'InProgress',
 'LastModifiedTime': datetime.datetime(2025, 6, 7, 21, 2, 22, 112000, tzinfo
=tzlocal()),
 'CreationTime': datetime.datetime(2025, 6, 7, 21, 2, 21, 705000, tzinfo=tz
local()),
 'MonitoringScheduleArn': 'arn:aws:sagemaker:us-east-1:672518276407:monitor
ing-schedule/DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015',
 'ResponseMetadata': {'RequestId': '03b0da72-dfd3-46b6-b484-d0b6c2b170bd',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '03b0da72-dfd3-46b6-b484-d0b6c2b170b
```

```
d',
    'content-type': 'application/x-amz-json-1.1',
    'content-length': '2303',
    'date': 'Sat, 07 Jun 2025 21:02:34 GMT',
    'connection': 'close'},
   'RetryAttempts': 0}}
```

Inspect a specific execution (latest execution)

In the previous cell, you picked up the latest completed or failed scheduled execution. Here are the possible terminal states and what each of them mean:

- Completed - This means the monitoring execution completed and no issues were found in the violations report.
- CompletedWithViolations - This means the execution completed, but constraint violations were detected.
- Failed - The monitoring execution failed, maybe due to client error (perhaps incorrect role permissions) or infrastructure issues. Further examination of FailureReason and ExitMessage is necessary to identify what exactly happened.
- Stopped - job exceeded max runtime or was manually stopped.

In [ ]:
```
status = execution["MonitoringExecutionStatus"]

while status in ["Pending", "InProgress"]:
    print("Waiting for execution to finish", end="")
    latest_execution.wait(logs=False)
    latest_job = latest_execution.describe()
    print()
    print(f"{latest_job['ProcessingJobName']} job status:", latest_job["Proc
    print(
        f"{latest_job['ProcessingJobName']} job exit message, if any:",
        latest_job.get("ExitMessage"),
    )
    print(
        f"{latest_job['ProcessingJobName']} job failure reason, if any:",
        latest_job.get("FailureReason"),
    )
    sleep(
        30
    )  # model quality executions consist of two Processing jobs, wait for s
    latest_execution = churn_model_quality_monitor.list_executions()[-1]
    execution = churn_model_quality_monitor.describe_schedule()["LastMonitor
    status = execution["MonitoringExecutionStatus"]

print("Execution status is:", status)

if status != "Completed":
    print(execution)
    print(
        "====STOP==== \n No completed executions to inspect further. Please
    )
```

```
Waiting for execution to finis
h.....................................................!
groundtruth-merge-202506072100-0b362d86ea6613c993bd81bf job status: Completed
groundtruth-merge-202506072100-0b362d86ea6613c993bd81bf job exit message, if
any: None
groundtruth-merge-202506072100-0b362d86ea6613c993bd81bf job failure reason, i
f any: None
Waiting for execution to finis
h.....................................................!
model-quality-monitoring-202506072100-0b362d86ea6613c993bd81bf job status: Co
mpleted
model-quality-monitoring-202506072100-0b362d86ea6613c993bd81bf job exit messa
ge, if any: CompletedWithViolations: Job completed successfully with 10 viola
tions.
model-quality-monitoring-202506072100-0b362d86ea6613c993bd81bf job failure re
ason, if any: None
Execution status is: CompletedWithViolations
{'MonitoringScheduleName': 'DEMO-xgb-churn-monitoring-schedule-2025-06-07-201
5', 'ScheduledTime': datetime.datetime(2025, 6, 7, 21, 0, tzinfo=tzlocal()),
'CreationTime': datetime.datetime(2025, 6, 7, 21, 1, 46, 475000, tzinfo=tzloc
al()), 'LastModifiedTime': datetime.datetime(2025, 6, 7, 21, 12, 31, 41000, tz
info=tzlocal()), 'MonitoringExecutionStatus': 'CompletedWithViolations', 'Pro
cessingJobArn': 'arn:aws:sagemaker:us-east-1:672518276407:processing-job/mode
l-quality-monitoring-202506072100-0b362d86ea6613c993bd81bf', 'EndpointName':
'DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005'}
====STOP====
 No completed executions to inspect further. Please wait till an execution co
mpletes or investigate previously reported failures.
```

In [ ]:
```
latest_execution = churn_model_quality_monitor.list_executions()[-1]
report_uri = latest_execution.describe()["ProcessingOutputConfig"]["Outputs"]
    "S3Uri"
]
print("Report Uri:", report_uri)
```

```
Report Uri: s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualit
yMonitor-20201201/baselining/results/DEMO-xgb-churn-model-quality-monitor-202
5-06-07-2005/DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015/2025/06/07/21
```

4.5 View violations generated by monitoring schedule

If there are any violations compared to the baseline, they will be listed in the reports uploaded to S3.

In [ ]:
```
pd.options.display.max_colwidth = None
violations = latest_execution.constraint_violations().body_dict["violations"
violations_df = pd.json_normalize(violations)
violations_df.head(10)
```

Out[ ]:

	constraint_check_type	description	metric_name
0	LessThanThreshold	Metric auc with 0.5149098549103291 +/- 5.361576254696403E-4 was LessThanThreshold '0.9395127993393898'	auc
1	LessThanThreshold	Metric precision with 0.6989720998531571 +/- 0.0018777543419145107 was LessThanThreshold '1.0'	precision
2	LessThanThreshold	Metric truePositiveRate with 0.11494808017387105 +/- 8.510006876141779E-4 was LessThanThreshold '0.5714285714285714'	truePositiveRate
3	LessThanThreshold	Metric f1 with 0.19742845292409789 +/- 0.0012994099799027011 was LessThanThreshold '0.7272727272727273'	f1
4	LessThanThreshold	Metric accuracy with 0.3221229637414609 +/- 6.57992592994707E-4 was LessThanThreshold '0.9402985074626866'	accuracy
5	GreaterThanThreshold	Metric falsePositiveRate with 0.13073979591836735 +/- 7.487536756202479E-4 was GreaterThanThreshold '0.0'	falsePositiveRate
6	LessThanThreshold	Metric trueNegativeRate with 0.8692602040816326 +/- 7.487536756202579E-4 was LessThanThreshold '1.0'	trueNegativeRate
7	GreaterThanThreshold	Metric falseNegativeRate with 0.885051919826129 +/- 8.510006876141446E-4 was GreaterThanThreshold '0.4285714285714286'	falseNegativeRate
8	LessThanThreshold	Metric recall with 0.11494808017387105 +/- 8.510006876141779E-4 was LessThanThreshold '0.5714285714285714'	recall
9	LessThanThreshold	Metric f2 with 0.13801101768628588 +/- 9.89833376620542E-4 was LessThanThreshold '0.625'	f2

Here you can see that one of the violations generated is that the f2 score is less than the threshold value set as part of baselining.

Section 5 - Analyze model quality CloudWatch metrics

In addition to the violations, the monitoring schedule also emits CloudWatch metrics. In this section, you will view the metrics generated and setup an CloudWatch alarm to be triggered when the model quality drifts from the baseline thresholds. You could use CloudWatch

alarms to trigger remedial actions such as retraining your model or updating the training dataset.

5.1 List the CW metrics generated.

In [ ]:
```
# Create CloudWatch client
cw_client = boto3.Session().client("cloudwatch")

namespace = "aws/sagemaker/Endpoints/model-metrics"

cw_dimensions = [
    {"Name": "Endpoint", "Value": endpoint_name},
    {"Name": "MonitoringSchedule", "Value": churn_monitor_schedule_name},
]
```

In [ ]:
```
# List metrics through the pagination interface
paginator = cw_client.get_paginator("list_metrics")

for response in paginator.paginate(Dimensions=cw_dimensions, Namespace=names
    model_quality_metrics = response["Metrics"]
    for metric in model_quality_metrics:
        print(metric["MetricName"])
```

```
auc
recall
f2
f0_5_best_constant_classifier
precision_best_constant_classifier
false_positive_rate
f1_best_constant_classifier
total_number_of_violations
recall_best_constant_classifier
au_prc
f1
accuracy_best_constant_classifier
false_negative_rate
f0_5
accuracy
true_positive_rate
true_negative_rate
precision
f2_best_constant_classifier
```

5.2 Create a CloudWatch Alarm

Based on the cloud watch metrics, you can create a cloud watch alarm when a specific metric does not meet the threshold configured. Here you will create an alarm if the f2 value of the model fall below the threshold suggested by the baseline constraints.

In [ ]:
```
alarm_name = "MODEL_QUALITY_F2_SCORE"
alarm_desc = (
    "Trigger an CloudWatch alarm when the f2 score drifts away from the base
)
```

```
mdoel_quality_f2_drift_threshold = (
    0.625  ##Setting this threshold purposefully low to see the alarm quickl
)
metric_name = "f2"
namespace = "aws/sagemaker/Endpoints/model-metrics"

cw_client.put_metric_alarm(
    AlarmName=alarm_name,
    AlarmDescription=alarm_desc,
    ActionsEnabled=True,
    MetricName=metric_name,
    Namespace=namespace,
    Statistic="Average",
    Dimensions=[
        {"Name": "Endpoint", "Value": endpoint_name},
        {"Name": "MonitoringSchedule", "Value": churn_monitor_schedule_name}
    ],
    Period=600,
    EvaluationPeriods=1,
    DatapointsToAlarm=1,
    Threshold=mdoel_quality_f2_drift_threshold,
    ComparisonOperator="LessThanOrEqualToThreshold",
    TreatMissingData="breaching",
)
```

Out[ ]:
```
{'ResponseMetadata': {'RequestId': '6b871aaa-9278-4711-862f-15986c1d09cf',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '6b871aaa-9278-4711-862f-15986c1d09c
f',
   'content-type': 'text/xml',
   'content-length': '214',
   'date': 'Sat, 07 Jun 2025 21:12:59 GMT'},
  'RetryAttempts': 0}}
```

5.3 Validation

In a few minutes, you should see a CloudWatch alarm created. The alarm will first be in "Insufficient Data" state and moves into "Alert" state. This can be verified in the CloudWatch console

 No description has been provided for this image

 No description has been provided for this image

Once the CW Alarm is generated, you can decide on what actions you want to take on these alerts. A possible action could be updating the training data an retraining the model

Define Bias Configuration and creating ModelQualityJobDefinition

In [153…
```
import boto3
```

```
region = "us-east-1"
bucket = "sagemaker-us-east-1-672518276407"
endpoint_name = "DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005"
ground_truth_uri = (
    "s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualityMonit
)
report_path = (
    "s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQualityMonit
)
schedule_name = "DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015"
job_definition_name = "DEMO-xgb-churn-quality-job-def-2025-06-07-v4"

role_arn = boto3.client("sts").get_caller_identity()["Arn"].replace(":user/"

# Step1: Create Model Quality Job Definition
sm_client = boto3.client("sagemaker", region_name=region)

role_arn = "arn:aws:iam::672518276407:role/LabRole"

response = sm_client.create_model_quality_job_definition(
    JobDefinitionName=job_definition_name,
    ModelQualityBaselineConfig={},
    ModelQualityAppSpecification={
        'ImageUri': '156813124566.dkr.ecr.us-east-1.amazonaws.com/sagemaker-
        'ProblemType': 'BinaryClassification'
    },
    ModelQualityJobInput={
        'EndpointInput': {
            'EndpointName': endpoint_name,
            'LocalPath': '/opt/ml/processing/input/endpoint',
            'S3InputMode': 'File',
            'S3DataDistributionType': 'FullyReplicated',
            'InferenceAttribute': 'predicted_label',
        },
        'GroundTruthS3Input': {
            'S3Uri': ground_truth_uri
        }
    },
    ModelQualityJobOutputConfig={
        'MonitoringOutputs': [
            {
                'S3Output': {
                    'S3Uri': report_path,
                    'LocalPath': '/opt/ml/processing/output',
                    'S3UploadMode': 'EndOfJob'
                }
            }
        ]
    },
    JobResources={
        'ClusterConfig': {
            'InstanceCount': 1,
            'InstanceType': 'ml.m5.xlarge',
            'VolumeSizeInGB': 20
        }
    },
```

```
    RoleArn=role_arn,
    StoppingCondition={
        'MaxRuntimeInSeconds': 3500
    }
)

print("\n Created ModelQualityJobDefinition:", response["JobDefinitionArn"])
```

```
 Created ModelQualityJobDefinition: arn:aws:sagemaker:us-east-1:672518276407:m
odel-quality-job-definition/DEMO-xgb-churn-quality-job-def-2025-06-07-v4
```

Create Monitoring Schedule

In [157…

```
schedule_name = "DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015-v4"

schedule_response = sm_client.create_monitoring_schedule(
    MonitoringScheduleName=schedule_name,
    MonitoringScheduleConfig={
        'ScheduleConfig': {
            'ScheduleExpression': 'cron(0 */2 ? * * *)'  # every 2 hour
        },
        'MonitoringJobDefinitionName': job_definition_name,
        'MonitoringType': 'ModelQuality'
    }
)

print("\n Created Monitoring Schedule:", schedule_response["MonitoringSchedu
```

```
 Created Monitoring Schedule: arn:aws:sagemaker:us-east-1:672518276407:monito
ring-schedule/DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015-v4
```

Check the report status

In [160…

```
import json
from datetime import datetime

def json_converter(o):
    if isinstance(o, datetime):
        return o.isoformat()

# Describe MonitoringSchedule
response = sm_client.describe_monitoring_schedule(MonitoringScheduleName=sch

# Print clean JSON
print(json.dumps(response, indent=2, default=json_converter))
```

```
{
  "MonitoringScheduleArn": "arn:aws:sagemaker:us-east-1:672518276407:monitoring-schedule/DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015-v4",
  "MonitoringScheduleName": "DEMO-xgb-churn-monitoring-schedule-2025-06-07-2015-v4",
  "MonitoringScheduleStatus": "Scheduled",
  "MonitoringType": "ModelQuality",
  "CreationTime": "2025-06-07T22:07:21.720000+00:00",
  "LastModifiedTime": "2025-06-07T22:07:29.354000+00:00",
  "MonitoringScheduleConfig": {
    "ScheduleConfig": {
      "ScheduleExpression": "cron(0 */2 ? * * *)"
    },
    "MonitoringJobDefinitionName": "DEMO-xgb-churn-quality-job-def-2025-06-07-v4",
    "MonitoringType": "ModelQuality"
  },
  "EndpointName": "DEMO-xgb-churn-model-quality-monitor-2025-06-07-2005",
  "ResponseMetadata": {
    "RequestId": "64dffc65-8bb9-4212-b722-50a93d291561",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "64dffc65-8bb9-4212-b722-50a93d291561",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "631",
      "date": "Sat, 07 Jun 2025 22:14:03 GMT"
    },
    "RetryAttempts": 0
  }
}
```


Monitor the report

In [ ]:

```python
import s3fs
import pandas as pd
import json

# Report path
report_path = "s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-ModelQu

# Setup S3 FS
fs = s3fs.S3FileSystem()

# List report folders
report_prefix = report_path.replace("s3://", "")
all_reports = fs.ls(report_prefix)

# Sort by latest timestamp
sorted_reports = sorted(all_reports, reverse=True)
latest_report_folder = sorted_reports[0]
print(f"Latest report folder: s3://{latest_report_folder}")

# Read constraint violations
violations_file = latest_report_folder + "/constraint_violations.json"
with fs.open(violations_file) as f:
```

```
    violations_data = json.load(f)
print("\n===== Constraint Violations =====")
print(json.dumps(violations_data, indent=2))

# Read statistics
statistics_file = latest_report_folder + "/statistics.json"
with fs.open(statistics_file) as f:
    statistics_data = json.load(f)
print("\n===== Statistics =====")
print(json.dumps(statistics_data, indent=2))

# Parse model_quality metrics → DataFrame
try:
    metrics = statistics_data["model_quality"]["statistics"]["binary_classifi
    df_metrics = pd.DataFrame([metrics])
    print("\n===== Parsed Model Quality Metrics =====")
    print(df_metrics)
except Exception as e:
    print(f"\n Could not parse model quality metrics: {e}")
```

The job status needs to be monitored for every 2 hours

Monitor and Respond to Bias Drift

In [170…

```
model_bias_monitor.create_monitoring_schedule(
    monitor_schedule_name='bias-drift-monitor-schedule-v3',
    endpoint_input=EndpointInput(
        endpoint_name=endpoint_name,
        destination='/opt/ml/processing/input/endpoint',
        start_time_offset='-PT1H',
        end_time_offset='-PT0H',
        probability_threshold_attribute=0.8
    ),
    ground_truth_input="s3://sagemaker-us-east-1-672518276407/sagemaker/Chur
    output_s3_uri=baseline_results_uri,
    analysis_config="s3://sagemaker-us-east-1-672518276407/sagemaker/Churn-Mo
    schedule_cron_expression=CronExpressionGenerator.hourly(),
    enable_cloudwatch_metrics=True
)

print("\n create_monitoring_schedule for BiasMonitor successfully completed!
```

```
INFO:sagemaker.model_monitor.model_monitoring:Creating Monitoring Schedule wi
th name: bias-drift-monitor-schedule-v3
 create_monitoring_schedule for BiasMonitor successfully completed!
```

Clean up

You can keep your endpoint running to continue capturing data. If you do not plan to collect more data or use this endpoint further, you should delete the endpoint to avoid incurring additional charges. Note that deleting your endpoint does not delete the data that was

captured during the model invocations. That data persists in Amazon S3 until you delete it yourself.

But before that, you need to delete the schedule first.

In [ ]:
```
churn_model_quality_monitor.delete_monitoring_schedule()
sleep(60)  # actually wait for the deletion
```

In [ ]:
```
predictor.delete_model()
predictor.delete_endpoint()
```

Notebook CI Test Results

This notebook was tested in multiple regions. The test results are as follows, except for us-west-2 which is shown at the top of the notebook.


This ap-southeast-2 badge failed to load. Check your device's internet connectivity, otherwise the service is currently unavailable


This ap-northeast-1 badge failed to load. Check your device's internet connectivity, otherwise the service is currently unavailable


This ap-northeast-2 badge failed to load. Check your device's internet connectivity, otherwise the service is currently unavailable


This ap-south-1 badge failed to load. Check your device's internet connectivity, otherwise the service is currently unavailable