Save to S3 with a SageMaker Processing Job

Quick Start To save your processed data to S3, select the Run menu above and click **Run all cells**. **View the status of the export job and the output S3 location**.

This notebook executes your Data Wrangler Flow `untitled2.flow` on the entire dataset using a SageMaker Processing Job and will save the processed data to S3.

This notebook saves data from the step `Featurize Date Time` from `Source: Yum-Yum-Ice-Cream.Csv` . To save from a different step, go to Data Wrangler to select a new step to export.

Contents

Inputs and Outputs

The below settings configure the inputs and outputs for the flow export.

Configurable Settings In **Input - Source** you can configure the data sources that will be used as input by Data Wrangler 1. For S3 sources, configure the source attribute that points to the input S3 prefixes 2. For all other sources, configure attributes like query_string, database in the source's **DatasetDefinition** object. If you modify the inputs the provided data must have the same schema and format as the data used in the Flow. You should also re-execute the cells in this section if you have modified the settings in any data sources.

In [2]:

```
from sagemaker.processing import ProcessingInput, ProcessingOutput
from sagemaker.dataset_definition.inputs import AthenaDatasetDefinition, DatasetDefinition, RedshiftDatasetDefinition

data_sources = []
```

Input - S3 Source: yum-yum-ice-cream.csv

In [3]:

```
data_sources.append(ProcessingInput(
    source="s3://mle-udacity/yum-yum-ice-cream.csv", # You can override this to point to other dataset on S3
    destination="/opt/ml/processing/yum-yum-ice-cream.csv",
    input_name="yum-yum-ice-cream.csv",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
))
```

Output: S3 settings

Configurable Settings 1. **bucket**: you can configure the S3 bucket where Data Wrangler will save the output. The default bucket from the SageMaker notebook session is used. 2. **flow_export_id**: A randomly generated export id. The export id must be unique to ensure the results do not conflict with other flow exports 3. **s3_ouput_prefix**: you can configure the directory name in your bucket where your data will be saved.

In [4]:

```
import time
import uuid
import sagemaker

# Sagemaker session
sess = sagemaker.Session()

# You can configure this with your own bucket name, e.g.
# bucket = "my-bucket"
bucket = "mle-udacity"
print(f"Data Wrangler export storage bucket: {bucket}")

# unique flow export ID
flow_export_id = f"{time.strftime('%d-%H-%M-%S', time.gmtime())}-{str(uuid.uuid4())[:8]}"
flow_export_name = f"flow-{flow_export_id}"
```

```
Data Wrangler export storage bucket: mle-udacity
```

Below are the inputs required by the SageMaker Python SDK to launch a processing job.

In [5]:

```
# Output name is auto-generated from the select node's ID + output name from the flow file.
output_name = "72955a16-0246-4626-8d37-42280bb963e7.default"

s3_output_prefix = f"export-{flow_export_name}/output"
s3_output_path = f"s3://{bucket}/{s3_output_prefix}"
print(f"Flow S3 export result path: {s3_output_path}")

processing_job_output = ProcessingOutput(
    output_name=output_name,
    source="/opt/ml/processing/output",
    destination=s3_output_path,
    s3_upload_mode="EndOfJob"
)
```

```
Flow S3 export result path: s3://mle-udacity/export-flow-30-02-49-30-600e351c/output
```

Upload Flow to S3

To use the Data Wrangler as an input to the processing job, first upload your flow file to Amazon S3.

In [6]:

```
import os
import json
import boto3

# name of the flow file which should exist in the current notebook working directory
flow_file_name = "untitled2.flow"

# Load .flow file from current notebook working directory
!echo "Loading flow file from current notebook working directory: $PWD"

with open(flow_file_name) as f:
    flow = json.load(f)

# Upload flow to S3
s3_client = boto3.client("s3")
s3_client.upload_file(flow_file_name, bucket, f"data_wrangler_flows/{flow_export_name}.flow", ExtraArgs={"ServerSideEncry
ption": "aws:kms"})

flow_s3_uri = f"s3://{bucket}/data_wrangler_flows/{flow_export_name}.flow"

print(f"Data Wrangler flow {flow_file_name} uploaded to {flow_s3_uri}")
```

```
Loading flow file from current notebook working directory: /root/nd009t-c1-intro-to-ml-templates/exploratory-
data-analysis/concept2_data_wrangler/starter
Data Wrangler flow untitled2.flow uploaded to s3://mle-udacity/data_wrangler_flows/flow-30-02-49-30-600e351c.
flow
```

The Data Wrangler Flow is also provided to the Processing Job as an input source which we configure below.

In [7]:

```
## Input - Flow: untitled2.flow
flow_input = ProcessingInput(
    source=flow_s3_uri,
    destination="/opt/ml/processing/flow",
    input_name="flow",
    s3_data_type="S3Prefix",
    s3_input_mode="File",
    s3_data_distribution_type="FullyReplicated"
)
```

Run Processing Job

Job Configurations

💡 **Configurable Settings** You can configure the following settings for Processing Jobs. If you change any configurations you will need to re-execute this and all cells below it by selecting the Run menu above and click **Run Selected Cells and All Below** 1. IAM role for executing the processing job. 2. A unique name of the processing job. Give a unique name every time you re-execute processing jobs 3. Data Wrangler Container URL. 4. Instance count, instance type and storage volume size in GB. 5. Content type for each output. Data Wrangler supports CSV as default and Parquet. 6. Network Isolation settings 7. KMS key to encrypt output data

In [8]:

```
# IAM role for executing the processing job.
iam_role = sagemaker.get_execution_role()

# Unique processing job name. Give a unique name every time you re-execute processing jobs
processing_job_name = f"data-wrangler-flow-processing-{flow_export_id}"

# Data Wrangler Container URL.
container_uri = "663277389841.dkr.ecr.us-east-1.amazonaws.com/sagemaker-data-wrangler-container:1.x"
# Pinned Data Wrangler Container URL.
container_uri_pinned = "663277389841.dkr.ecr.us-east-1.amazonaws.com/sagemaker-data-wrangler-container:1.11.2"

# Processing Job Instance count and instance type.
instance_count = 2
instance_type = "ml.m5.4xlarge"

# Size in GB of the EBS volume to use for storing data during processing
volume_size_in_gb = 30

# Content type for each output. Data Wrangler supports CSV as default and Parquet.
output_content_type = "CSV"

# Network Isolation mode; default is off
enable_network_isolation = False

# Output configuration used as processing job container arguments
output_config = {
    output_name: {
        "content_type": output_content_type
    }
}

# KMS key for per object encryption; default is None
kms_key = None
```

Create Processing Job

To launch a Processing Job, you will use the SageMaker Python SDK to create a Processor function.

In [9]:

```
from sagemaker.processing import Processor
from sagemaker.network import NetworkConfig

processor = Processor(
    role=iam_role,
    image_uri=container_uri,
    instance_count=instance_count,
    instance_type=instance_type,
    volume_size_in_gb=volume_size_in_gb,
    network_config=NetworkConfig(enable_network_isolation=enable_network_isolation),
    sagemaker_session=sess,
    output_kms_key=kms_key
)

# Start Job
processor.run(
    inputs=[flow_input] + data_sources,
    outputs=[processing_job_output],
    arguments=[f"--output-config '{json.dumps(output_config)}'"],
    wait=False,
    logs=False,
    job_name=processing_job_name
)
```

```
Job Name:  data-wrangler-flow-processing-30-02-49-30-600e351c
Inputs:  [{'InputName': 'flow', 'AppManaged': False, 'S3Input': {'S3Uri': 's3://mle-udacity/data_wrangler_flo
ws/flow-30-02-49-30-600e351c.flow', 'LocalPath': '/opt/ml/processing/flow', 'S3DataType': 'S3Prefix', 'S3Inpu
tMode': 'File', 'S3DataDistributionType': 'FullyReplicated', 'S3CompressionType': 'None'}}, {'InputName': 'yu
m-yum-ice-cream.csv', 'AppManaged': False, 'S3Input': {'S3Uri': 's3://mle-udacity/yum-yum-ice-cream.csv', 'Lo
calPath': '/opt/ml/processing/yum-yum-ice-cream.csv', 'S3DataType': 'S3Prefix', 'S3InputMode': 'File', 'S3Dat
aDistributionType': 'FullyReplicated', 'S3CompressionType': 'None'}}]
Outputs:  [{'OutputName': '72955a16-0246-4626-8d37-42280bb963e7.default', 'AppManaged': False, 'S3Output':
{'S3Uri': 's3://mle-udacity/export-flow-30-02-49-30-600e351c/output', 'LocalPath': '/opt/ml/processing/outpu
t', 'S3UploadMode': 'EndOfJob'}}]
```

Job Status & S3 Output Location

Below you wait for processing job to finish. If it finishes successfully, the raw parameters used by the Processing Job will be printed

In [10]:

```
s3_job_results_path = f"s3://{bucket}/{s3_output_prefix}/{processing_job_name}"
print(f"Job results are saved to S3 path: {s3_job_results_path}")

job_result = sess.wait_for_processing_job(processing_job_name)
job_result
```

```
Job results are saved to S3 path: s3://mle-udacity/export-flow-30-02-49-30-600e351c/output/data-wrangler-flow
-processing-30-02-49-30-600e351c
................................................................!
```

Out[10]:

```
{'ProcessingInputs': [{'InputName': 'flow',
   'AppManaged': False,
   'S3Input': {'S3Uri': 's3://mle-udacity/data_wrangler_flows/flow-30-02-49-30-600e351c.flow',
    'LocalPath': '/opt/ml/processing/flow',
    'S3DataType': 'S3Prefix',
    'S3InputMode': 'File',
    'S3DataDistributionType': 'FullyReplicated',
    'S3CompressionType': 'None'}},
  {'InputName': 'yum-yum-ice-cream.csv',
   'AppManaged': False,
   'S3Input': {'S3Uri': 's3://mle-udacity/yum-yum-ice-cream.csv',
    'LocalPath': '/opt/ml/processing/yum-yum-ice-cream.csv',
    'S3DataType': 'S3Prefix',
    'S3InputMode': 'File',
    'S3DataDistributionType': 'FullyReplicated',
    'S3CompressionType': 'None'}}],
 'ProcessingOutputConfig': {'Outputs': [{'OutputName': '72955a16-0246-4626-8d37-42280bb963e7.default',
    'S3Output': {'S3Uri': 's3://mle-udacity/export-flow-30-02-49-30-600e351c/output',
     'LocalPath': '/opt/ml/processing/output',
     'S3UploadMode': 'EndOfJob'},
    'AppManaged': False}]},
 'ProcessingJobName': 'data-wrangler-flow-processing-30-02-49-30-600e351c',
 'ProcessingResources': {'ClusterConfig': {'InstanceCount': 2,
   'InstanceType': 'ml.m5.4xlarge',
   'VolumeSizeInGB': 30}},
 'StoppingCondition': {'MaxRuntimeInSeconds': 86400},
 'AppSpecification': {'ImageUri': '663277389841.dkr.ecr.us-east-1.amazonaws.com/sagemaker-data-wrangler-conta
iner:1.x',
  'ContainerArguments': ['--output-config \'{"72955a16-0246-4626-8d37-42280bb963e7.default": {"content_type":
"CSV"}}\'']},
 'NetworkConfig': {'EnableInterContainerTrafficEncryption': False,
  'EnableNetworkIsolation': False},
 'RoleArn': 'arn:aws:iam::664961830858:role/service-role/AmazonSageMaker-ExecutionRole-20211028T004155',
 'ProcessingJobArn': 'arn:aws:sagemaker:us-east-1:664961830858:processing-job/data-wrangler-flow-processing-3
0-02-49-30-600e351c',
 'ProcessingJobStatus': 'Completed',
 'ProcessingEndTime': datetime.datetime(2021, 10, 30, 2, 56, 25, 514000, tzinfo=tzlocal()),
 'ProcessingStartTime': datetime.datetime(2021, 10, 30, 2, 55, 13, 343000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2021, 10, 30, 2, 56, 25, 859000, tzinfo=tzlocal()),
 'CreationTime': datetime.datetime(2021, 10, 30, 2, 51, 3, 91000, tzinfo=tzlocal()),
 'ResponseMetadata': {'RequestId': '8b9a2c6e-b0de-4027-87a0-bf46dd671bdf',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '8b9a2c6e-b0de-4027-87a0-bf46dd671bdf',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '2032',
   'date': 'Sat, 30 Oct 2021 02:56:27 GMT'},
  'RetryAttempts': 0}}
```

(Optional)Next Steps

Now that data is available on S3 you can use other SageMaker components that take S3 URIs as input such as SageMaker Training, Built-in Algorithms, etc. Similarly you can load the dataset into a Pandas dataframe in this notebook for further inspection and work. The examples below show how to do both of these steps.

By default optional steps do not run automatically, set `run_optional_steps` to True if you want to execute optional steps

In [11]:

```
run_optional_steps = False
```

In [12]:

```
# This will stop the below cells from executing if "Run All Cells" was used on the notebook.
if not run_optional_steps:
    raise SystemExit("Stop here. Do not automatically execute optional steps.")
```

```
An exception has occurred, use %tb to see the full traceback.

SystemExit: Stop here. Do not automatically execute optional steps.

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3339: UserWarning: To exit: use 'exi
t', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

(Optional) Load Processed Data into Pandas

We use the AWS Data Wrangler library (https://github.com/awslabs/aws-data-wrangler) to load the exported dataset into a Pandas dataframe for a preview of first 1000 rows.

In [13]:

```
!pip install -q awswrangler pandas
import awswrangler as wr
```

```
/opt/conda/lib/python3.7/site-packages/secretstorage/dhcrypto.py:16: CryptographyDeprecationWarning: int_from
_bytes is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/lib/python3.7/site-packages/secretstorage/util.py:25: CryptographyDeprecationWarning: int_from_byt
es is deprecated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/v
env
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the '/opt/conda/bin/python -m pip install --upgrade pip' command.
```

In [ ]:

```
chunksize = 1000

if output_content_type.upper() == "CSV":
    dfs = wr.s3.read_csv(s3_output_path, chunksize=chunksize)
elif output_content_type.upper() == "PARQUET":
    dfs = wr.s3.read_parquet(s3_output_path, chunked=chunksize)
else:
    print(f"Unexpected output content type {output_content_type}")

df = next(dfs)
df
```

(Optional)Train a model with SageMaker

Now that the data has been processed, you may want to train a model using the data. The following shows an example of doing so using a popular algorithm - XGBoost. For more information on algorithms available in SageMaker, see Getting Started with SageMaker Algorithms (https://docs.aws.amazon.com/sagemaker/latest/dg/algos.html). It is important to note that the following XGBoost objective ['binary', 'regression', 'multiclass'] hyperparameters, or content_type may not be suitable for the output data, and will require changes to train a proper model. Furthermore, for CSV training, the algorithm assumes that the target variable is in the first column. For more information on SageMaker XGBoost, see https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html (https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost.html).

Set Training Data path

We set the training input data path from the output of the Data Wrangler processing job..

In [15]:

```
s3_training_input_path = s3_job_results_path
print(f"training input data path: {s3_training_input_path}")
```

```
training input data path: s3://mle-udacity/export-flow-30-02-49-30-600e351c/output/data-wrangler-flow-process
ing-30-02-49-30-600e351c
```

Configure the algorithm and training job

The Training Job hyperparameters are set. For more information on XGBoost Hyperparameters, see https://xgboost.readthedocs.io/en/latest/parameter.html (https://xgboost.readthedocs.io/en/latest/parameter.html).

In [16]:

```
region = boto3.Session().region_name
container = sagemaker.image_uris.retrieve("xgboost", region, "1.2-1")
hyperparameters = {
    "max_depth":"5",
    "objective": "reg:squarederror",
    "num_round": "10",
}
train_content_type = (
    "application/x-parquet" if output_content_type.upper() == "PARQUET"
    else "text/csv"
)
train_input = sagemaker.inputs.TrainingInput(
    s3_data=s3_training_input_path,
    content_type=train_content_type,
)
```

Start the Training Job

The TrainingJob configurations are set using the SageMaker Python SDK Estimator, and which is fit using the training data from the Processing Job that was run earlier.

In [ ]:

```
estimator = sagemaker.estimator.Estimator(
    container,
    iam_role,
    hyperparameters=hyperparameters,
    instance_count=1,
    instance_type="ml.m5.2xlarge",
)
estimator.fit({"train": train_input})
```

Now that you have a trained model there are a number of different things you can do. For more details on training with SageMaker, please see https://sagemaker.readthedocs.io/en/stable/frameworks/xgboost/using_xgboost.html (https://sagemaker.readthedocs.io/en/stable/frameworks/xgboost/using_xgboost.html).