XGBoost Built-in Algorithm - Bike Rental Regression Example

In [ ]:
```
import numpy as np
import pandas as pd

import boto3
import re

import sagemaker
from sagemaker import get_execution_role
# SageMaker SDK Documentation: http://sagemaker.readthedocs.io/en/latest/estimators.html
```

Upload Data to S3

In [ ]:
```
# Specify your bucket name
bucket_name = 'dwb-ml-sagemaker'

training_folder = r'bikerental/training/'
validation_folder = r'bikerental/validation/'
test_folder = r'bikerental/test/'

s3_model_output_location = r's3://{0}/bikerental/model'.format(bucket_name)
s3_training_file_location = r's3://{0}/{1}'.format(bucket_name,training_folder)
s3_validation_file_location = r's3://{0}/{1}'.format(bucket_name,validation_folder)
s3_test_file_location = r's3://{0}/{1}'.format(bucket_name,test_folder)
```

In [ ]:
```
print(s3_model_output_location)
print(s3_training_file_location)
print(s3_validation_file_location)
print(s3_test_file_location)
```

In [ ]:
```
# Write and Reading from S3 is just as easy
# files are referred as objects in S3.
# file name is referred as key name in S3

# File stored in S3 is automatically replicated across 3 different availability zones
```

```
# in the region where the bucket was created.

# http://boto3.readthedocs.io/en/latest/guide/s3.html
def write_to_s3(filename, bucket, key):
    with open(filename,'rb') as f: # Read in binary mode
        return boto3.Session().resource('s3').Bucket(bucket).Object(key).upload_fileobj(f)
```

In [ ]:
```
write_to_s3('bike_train.csv',
            bucket_name,
            training_folder + 'bike_train.csv')

write_to_s3('bike_validation.csv',
            bucket_name,
            validation_folder + 'bike_validation.csv')

write_to_s3('bike_test.csv',
            bucket_name,
            test_folder + 'bike_test.csv')
```

Training Algorithm Docker Image

SageMaker maintains a separate image for algorithm and region

https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-algo-docker-registry-paths.html

In [ ]:
```
# Use Spot Instance - Save up to 90% of training cost by using spot instances when compared to on-demand instances
# Reference: https://github.com/aws-samples/amazon-sagemaker-managed-spot-training/blob/main/xgboost_built_in_managed

# if you are still on two-month free-tier you can use the on-demand instance by setting:
#   use_spot_instances = False

# We will use spot for training
use_spot_instances = True
max_run = 3600 # in seconds
max_wait = 7200 if use_spot_instances else None # in seconds

job_name = 'xgboost-bikerental-v1'

checkpoint_s3_uri = None
```

```
if use_spot_instances:
    checkpoint_s3_uri = f's3://{bucket_name}/bikerental/checkpoints/{job_name}'

print (f'Checkpoint uri: {checkpoint_s3_uri}')
```

In [ ]:
```
# Establish a session with AWS
sess = sagemaker.Session()
```

In [ ]:
```
role = get_execution_role()
```

In [ ]:
```
# This role contains the permissions needed to train, deploy models
# SageMaker Service is trusted to assume this role
print(role)
```

In [ ]:
```
# https://sagemaker.readthedocs.io/en/stable/api/utility/image_uris.html#sagemaker.image_uris.retrieve

# SDK 2 uses image_uris.retrieve the container image location

# Use XGBoost 1.2 version
container = sagemaker.image_uris.retrieve("xgboost",sess.boto_region_name,version="1.2-2")

print (f'Using XGBoost Container {container}')
```

Build Model

In [ ]:
```
# Configure the training job
# Specify type and number of instances to use
# S3 location where final artifacts needs to be stored

#   Reference: http://sagemaker.readthedocs.io/en/latest/estimators.html

# for managed spot training, specify the use_spot_instances flag, max_run, max_wait and checkpoint_s3_uri

# SDK 2.x version does not require train prefix for instance count and type
estimator = sagemaker.estimator.Estimator(
    container,
    role,
    instance_count=1,
```

```
    instance_type='ml.m5.xlarge',
    output_path=s3_model_output_location,
    sagemaker_session=sess,
    base_job_name = job_name,
    use_spot_instances=use_spot_instances,
    max_run=max_run,
    max_wait=max_wait,
    checkpoint_s3_uri=checkpoint_s3_uri)
```

In [ ]:
```
# Specify hyper parameters that appropriate for the training algorithm
# XGBoost Training Parameter Reference
#  https://github.com/dmlc/xgboost/blob/master/doc/parameter.rst#learning-task-parameters
estimator.set_hyperparameters(max_depth=5,
                              objective="reg:squarederror",
                              eta=0.1,
                              num_round=150)
```

In [ ]:
```
estimator.hyperparameters()
```

Specify Training Data Location and Optionally, Validation Data Location

In [ ]:
```
# content type can be libsvm or csv for XGBoost
training_input_config = sagemaker.session.TrainingInput(
    s3_data=s3_training_file_location,
    content_type='csv',
    s3_data_type='S3Prefix')

validation_input_config = sagemaker.session.TrainingInput(
    s3_data=s3_validation_file_location,
    content_type='csv',
    s3_data_type='S3Prefix'
)

data_channels = {'train': training_input_config, 'validation': validation_input_config}
```

In [ ]:
```
print(training_input_config.config)
print(validation_input_config.config)
```

Train the model

In [ ]:
```
# XGBoost supports "train", "validation" channels
# Reference: Supported channels by algorithm
#   https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-algo-docker-registry-paths.html
estimator.fit(data_channels)
```

Deploy Model

In [ ]:
```
# Ref: http://sagemaker.readthedocs.io/en/latest/estimators.html
predictor = estimator.deploy(initial_instance_count=1,
                             instance_type='ml.m5.xlarge',
                             endpoint_name = job_name)
```

Run Predictions

In [ ]:
```
# SDK 2.0 serializers
from sagemaker.serializers import CSVSerializer
```

In [ ]:
```
predictor.serializer = CSVSerializer()
```

In [ ]:
```
predictor.predict([[3,0,1,2,28.7,33.335,79,12.998,2011,7,7,3]])
```

Summary

1. Ensure Training, Test and Validation data are in S3 Bucket
2. Select Algorithm Container Registry Path - Path varies by region
3. Configure Estimator for training - Specify Algorithm container, instance count, instance type, model output location
4. Specify algorithm specific hyper parameters
5. Train model
6. Deploy model - Specify instance count, instance type and endpoint name
7. Run Predictions