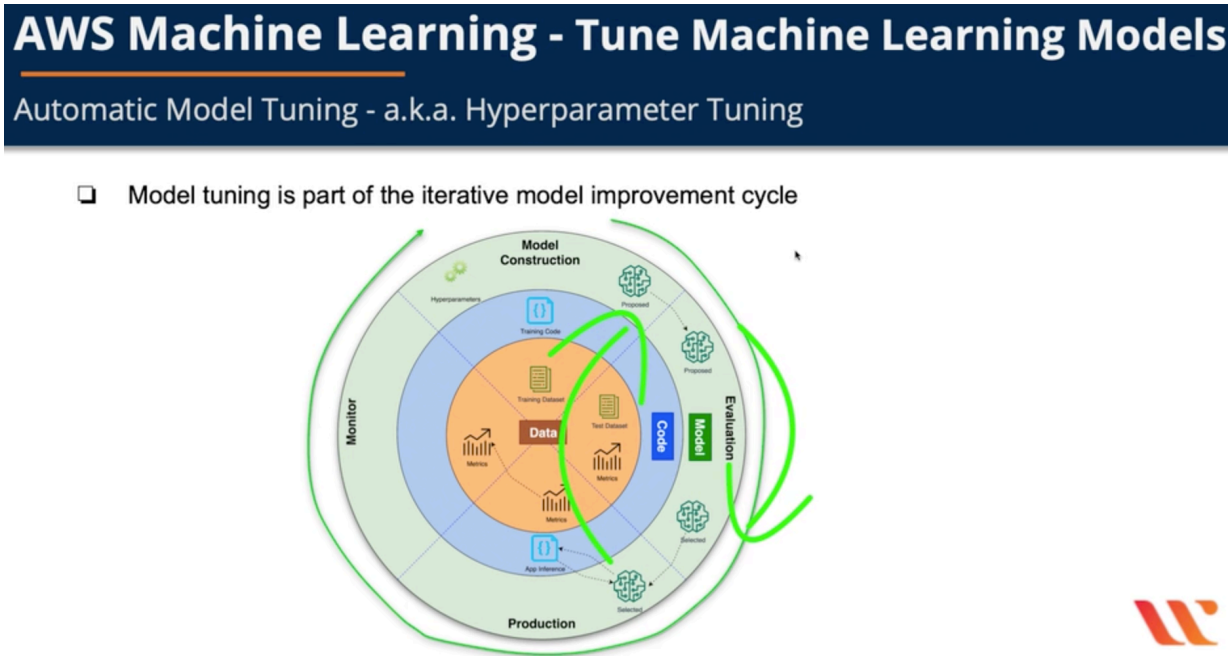
Whizlabs - ML Specialty Exam Course - Modeling - part 2

Automatic Model Tuning / Hyper-parameter Optimization



Definitions of Train, Validation, and Test Datasets

To reiterate the findings from researching the experts above, this section provides unambiguous definitions of the three terms.

- **Training Dataset**: The sample of data used to fit the model.
- **Validation Dataset**: The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration.
- **Test Dataset**: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

Tuning and Evaluating the Model

- Machine learning with SageMaker
  - Generate example data to use in training
  - Train the model to make predictions, or inferences
  - Deploy the model to an endpoint
  - Evaluate the model and find the hyperparameters to produce the most accurate, best performing model
    - Run a hyperparameter tuning job

Hyperparameter Tuning Job

- Hyperparameter tuning job
  - Finds the best version of a model by running many training jobs on your dataset using the algorithm and specified ranges of hyperparameters
  - Chooses the hyperparameter values that result in a model that performs the best, as measured by a specified metric
  - Use automatic model tuning with built-in algorithms, custom algorithms, and SageMaker pre-built containers for machine learning frameworks
  - Prerequisite: successfully run at least one training job
  - Two hyperparameter search approaches
    - Random search: random combination of hyperparameter values from within the specified ranges
    - Bayesian search: treats hyperparameter tuning like a regression problem

Define Metrics

- Built-in algorithms automatically send metrics to hyperparameter tuning
  - Choose one of the metrics as the objective metric for the tuning job
- Custom algorithms must emit at least one metric to stderr or stdout

```
=[
    {
        "Name": "loss",
        "Regex": "Loss = (.*?);",
    },
    {
        "Name": "ganloss",
        "Regex": "GAN_loss=(.*?);",
    },
    {
        "Name": "discloss",
        "Regex": "disc_train_loss=(.*?);",
    },
    {
        "Name": "disc-combined",
        "Regex": "disc-combined=(.*?);",
    },
]
```

Define Hyperparameter Ranges

- Finds best hyperparameter values by searching over ranges of hyperparameters
  - Specify hyperparameters and value ranges to search

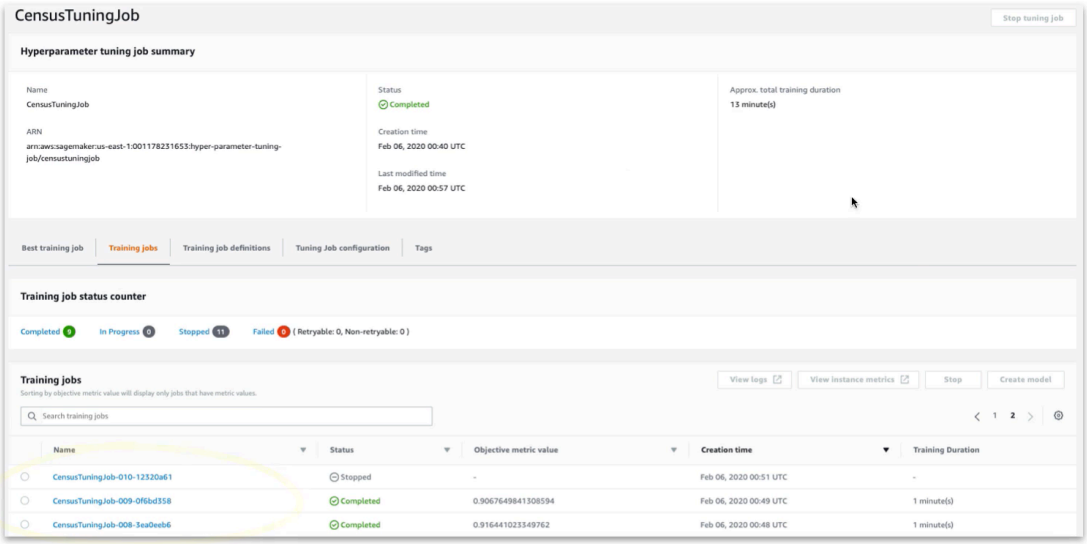
```
"ParameterRanges": {
      "CategoricalParameterRanges": [
        {
          "Name": "tree_method",
          "Values": ["auto", "exact", "approx", "hist"]
        }
      ],
      "ContinuousParameterRanges": [
        {
          "Name": "eta",
          "MaxValue" : "0.5",
          "MinValue": "0",
          "ScalingType": "Auto"
        }
      ],
      "IntegerParameterRanges": [
        {
          "Name": "max_depth",
          "MaxValue": "10",
          "MinValue": "1",
          "ScalingType": "Auto"
        }
      ]
    }
```

Hyperparameter Scaling

- Integer and continuous hyperparameter ranges: choose the scale for hyperparameter tuning to use to search the range of values using ScalingType
  - Auto: chooses the best scale
  - Linear: searches values in range using a linear scale
  - Logarithmic: searches values in range using a logarithmic scale
    - Only work for values greater than 0
    - Use when searching a range that spans several orders of magnitude
  - ReverseLogarithmic: searches values in range using a reverse logarithmic scale
    - Use when searching continuous hyperparameter ranges between 0 and 1.0
    - Use when searching a range that is highly sensitive to small changes that are very close to 1

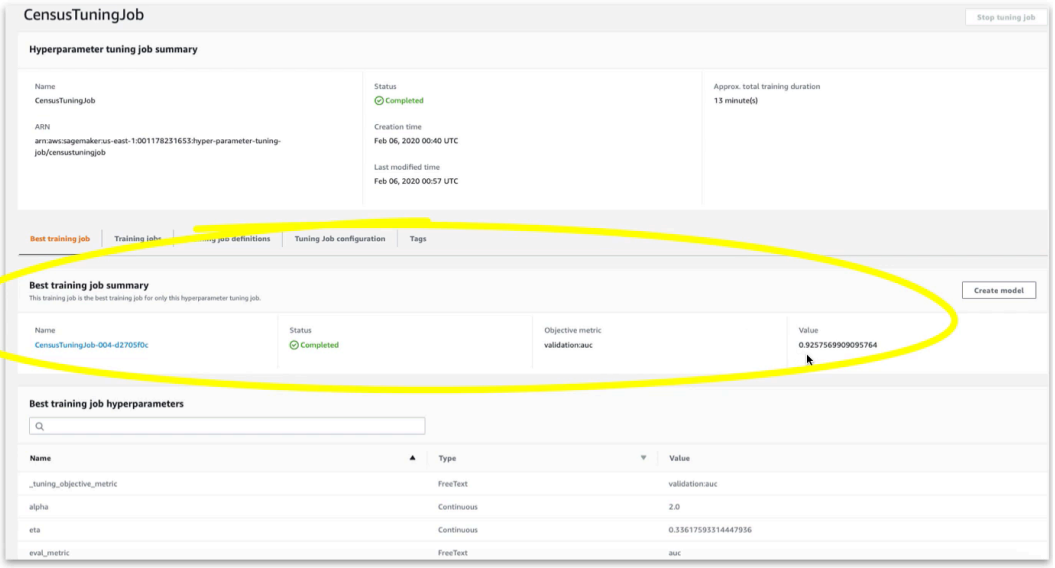
Example of Training jobs.

Hyperparameter Tuning Job



Best training job with a 0.9257 value of AUC

Hyperparameter Tuning Job



Automatic Model Tuning - Lab

Notebook available at: https://www.whizlabs.com/learn/media/2020/03/23/files_42-perform-automatic-model-tuning.ipynb

In [1]:
```
import sagemaker
import boto3
from sagemaker.predictor import csv_serializer    # Converts strings for HTTP POST requests on inference

import numpy as np                                # For performing matrix operations and numerical processing
import pandas as pd                               # For manipulating tabular data
from time import gmtime, strftime
import os

region = boto3.Session().region_name
smclient = boto3.Session().client('sagemaker')
```

In [2]:
```
from sagemaker import get_execution_role

role = get_execution_role()
print(role)
```

```
arn:aws:iam::001178231653:role/service-role/AmazonSageMaker-ExecutionRole-20191211T174650
```

In [3]:
```
bucket = 'train-census-earnings-xgboost'                          # Replace with the name of your S3 bucket
prefix = 'sagemaker/DEMO-automatic-model-tuning-xgboost-dm'
```

In [5]:
```
# Download from your S3 bucket the census data CSV file based on the publically available census data from the ML repos
from io import StringIO
s3 = boto3.resource('s3')
bucket_name = 'train-census-earnings-xgboost' # place the adult_census_clean.csv file in a bucket in your account
object_key = 'adult_census.csv'
# Load the data into a pandas dataframe

csv_obj = s3.Object(bucket_name, object_key)
csv_string = csv_obj.get()['Body'].read().decode('utf-8')

raw_data = pd.read_csv(StringIO(csv_string))
raw_data.head()
```

Out[5]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	y_no	y_yes
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	0	1
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	0	1
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	0	1
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	0	1
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	0	1

Don Onehot encoding

In [7]:
```
model_data = pd.get_dummies(raw_data)
model_data.head()
```

Out[7]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	y_no	y_yes	workclass_0	workclass_Federal-gov	...	native-country_Portugal	native-country_Puerto-Rico	native-country_Scotland	native-country_South	native-country_Taiwan	native-country_Thailand	Tr
0	39	77516	13	2174	0	40	0	1	0	0	...	0	0	0	0	0	0	
1	50	83311	13	0	0	13	0	1	0	0	...	0	0	0	0	0	0	
2	38	215646	9	0	0	40	0	1	0	0	...	0	0	0	0	0	0	
3	53	234721	7	0	0	40	0	1	0	0	...	0	0	0	0	0	0	
4	28	338409	13	0	0	40	0	1	0	0	...	0	0	0	0	0	0	

5 rows × 110 columns

Split the dataset into 3: train, test and validate.
Hyper parameter tuning will use the validation set

In [8]:
```
train_data, validation_data, test_data = np.split(model_data.sample(frac=1,
                                                                    random_state=1729),
                                                  [int(0.7 * len(model_data)),
                                                   int(0.9*len(model_data))])

pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('train.csv',
                                                                                           index=False,
                                                                                           header=False)
pd.concat([validation_data['y_yes'], validation_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('validation.csv',
                                                                                                     index=False,
                                                                                                     header=False)
pd.concat([test_data['y_yes'], test_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('test.csv',
                                                                                         index=False,
                                                                                         header=False)

boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train/train.csv')).upload_file('train.csv')
boto3.Session().resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'validation/validation.csv')).upload_file('va
print(train_data.shape, validation_data.shape, test_data.shape)
```

```
(22792, 110) (6512, 110) (3256, 110)
```

Now set up a configuration for the tuning job:
- 9 training jobs with 3 max in parallel

- validation metric: AUC that we want to maximize
- eta, alpha and min_child_weight, max_depth will be the 4 hyper parameters we want to tune

In [9]:
```
tuning_job_config = {
    "ParameterRanges": {
      "CategoricalParameterRanges": [],
      "ContinuousParameterRanges": [
        {
          "MaxValue": "1",
          "MinValue": "0",
          "Name": "eta"
        },
        {
          "MaxValue": "2",
          "MinValue": "0",
          "Name": "alpha"
        },
        {
          "MaxValue": "10",
          "MinValue": "1",
          "Name": "min_child_weight"
        }
      ],
      "IntegerParameterRanges": [
        {
          "MaxValue": "10",
          "MinValue": "1",
          "Name": "max_depth"
        }
      ]
    },
    "ResourceLimits": {
      "MaxNumberOfTrainingJobs": 9,
      "MaxParallelTrainingJobs": 3
    },
    "Strategy": "Bayesian",
    "HyperParameterTuningJobObjective": {
      "MetricName": "validation:auc",
      "Type": "Maximize"
    }
  }
```

Now run set the training job definition
We have a stopping condition of 30mn

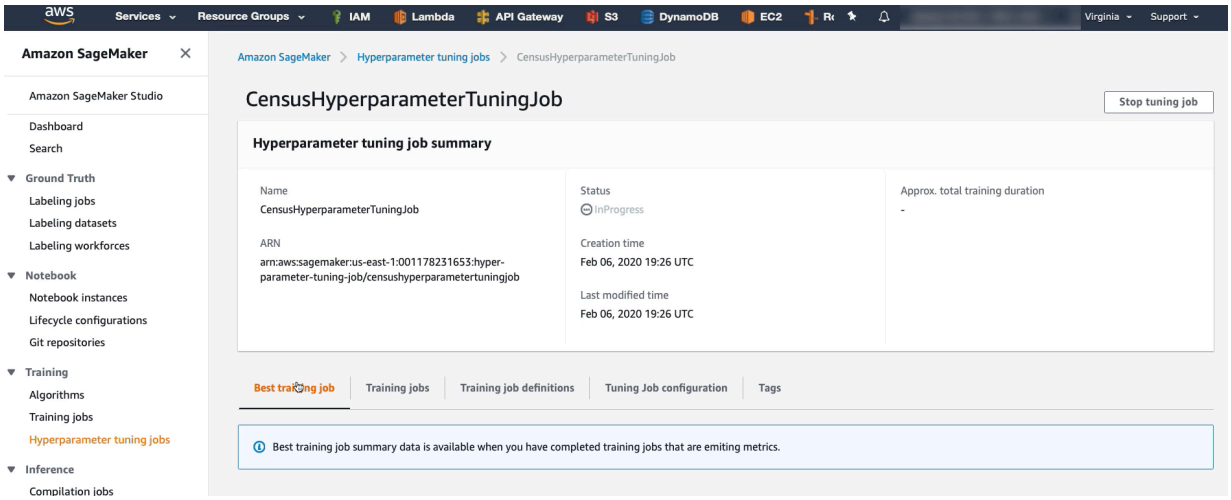
In [11]:
```
from sagemaker.amazon.amazon_estimator import get_image_uri
training_image = get_image_uri(boto3.Session().region_name, 'xgboost', '0.90-1')

s3_input_train = 's3://{}/{}/train'.format(bucket, prefix)
s3_input_validation ='s3://{}/{}/validation/'.format(bucket, prefix)

training_job_definition = {
    "AlgorithmSpecification": {
      "TrainingImage": training_image,
      "TrainingInputMode": "File"
    },
    "InputDataConfig": [
      {
        "ChannelName": "train",
        "CompressionType": "None",
        "ContentType": "csv",
        "DataSource": {
          "S3DataSource": {
            "S3DataDistributionType": "FullyReplicated",
            "S3DataType": "S3Prefix",
            "S3Uri": s3_input_train
          }
        }
      },
      {
        "ChannelName": "validation",
        "CompressionType": "None",
        "ContentType": "csv",
        "DataSource": {
          "S3DataSource": {
            "S3DataDistributionType": "FullyReplicated",
            "S3DataType": "S3Prefix",
            "S3Uri": s3_input_validation
          }
        }
      }
    ],
    "OutputDataConfig": {
      "S3OutputPath": "s3://{}/{}/output".format(bucket,prefix)
    },
    "ResourceConfig": {
      "InstanceCount": 2,
      "InstanceType": "ml.c4.2xlarge",
      "VolumeSizeInGB": 10
    },
    "RoleArn": role,
    "StaticHyperParameters": {
      "eval_metric": "auc",
      "num_round": "100",
      "objective": "binary:logistic",
      "rate_drop": "0.3",
      "tweedie_variance_power": "1.4"
    },
    "StoppingCondition": {
      "MaxRuntimeInSeconds": 1800
    }
}
```

Now rn hyper parameter tuning job

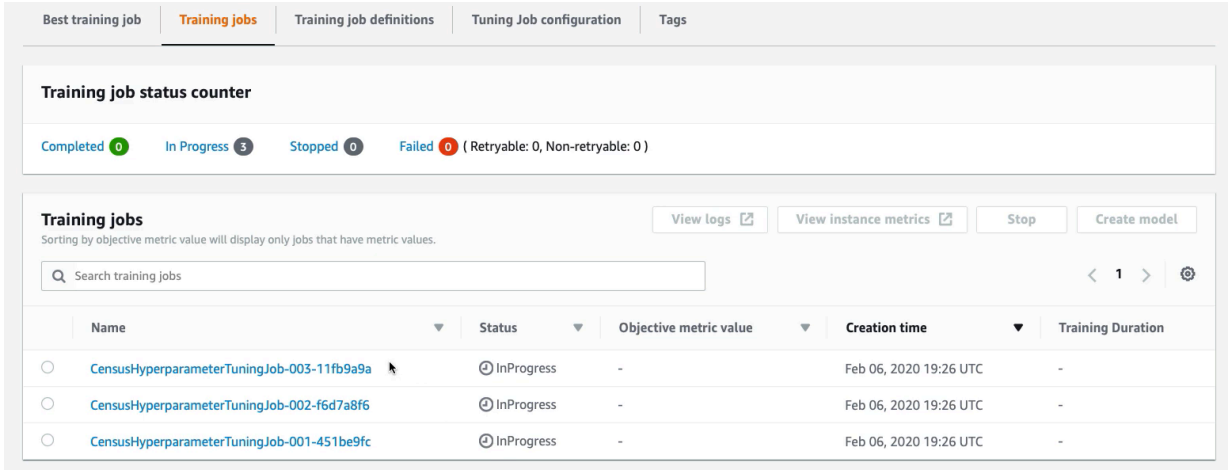
In [12]:
```
tuning_job_name = "CensusTuningJob"
smclient.create_hyper_parameter_tuning_job(HyperParameterTuningJobName = tuning_job_name,
                                           HyperParameterTuningJobConfig = tuning_job_config,
                                           TrainingJobDefinition = training_job_definition)
```

Out[12]:
```
{'HyperParameterTuningJobArn': 'arn:aws:sagemaker:us-east-1:001178231653:hyper-parameter-tuning-job/censustuningjob',
 'ResponseMetadata': {'RequestId': '2c3c5361-28a5-486d-ab41-22e6021bad13',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '2c3c5361-28a5-486d-ab41-22e6021bad13',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '116',
   'date': 'Thu, 06 Feb 2020 00:40:50 GMT'},
  'RetryAttempts': 0}}
```

Got to SageMaker Console

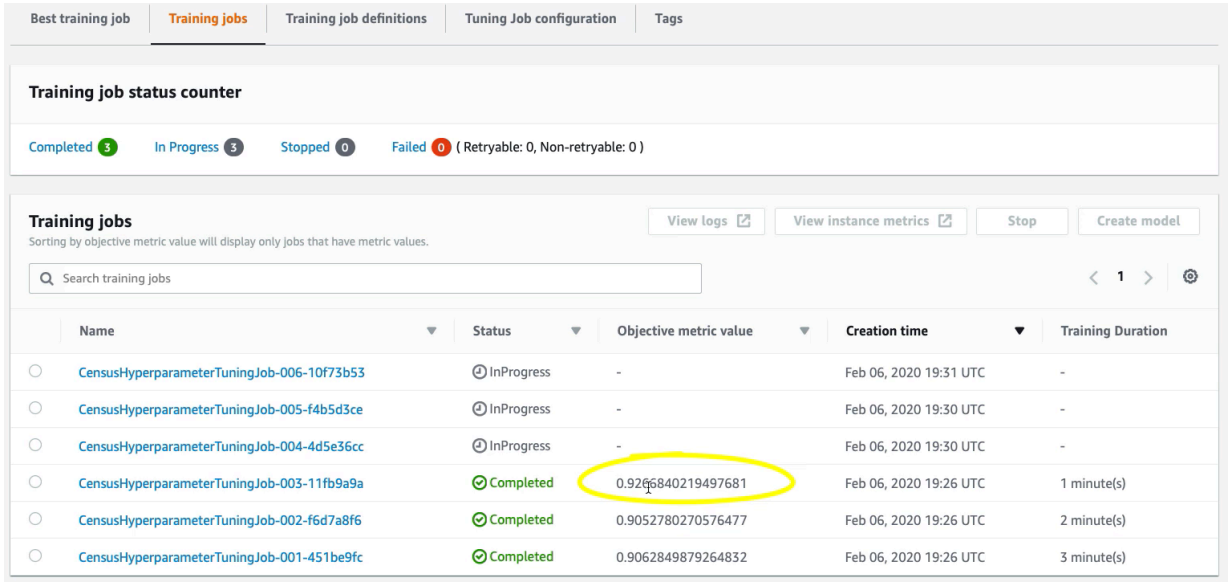


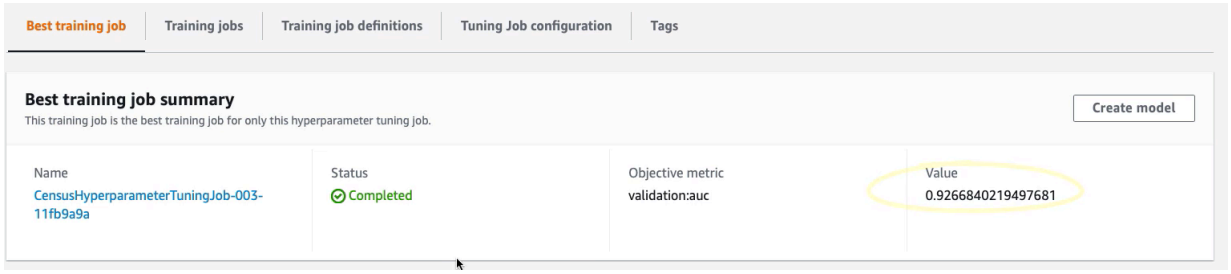
We can see 3 jobs running in parallel

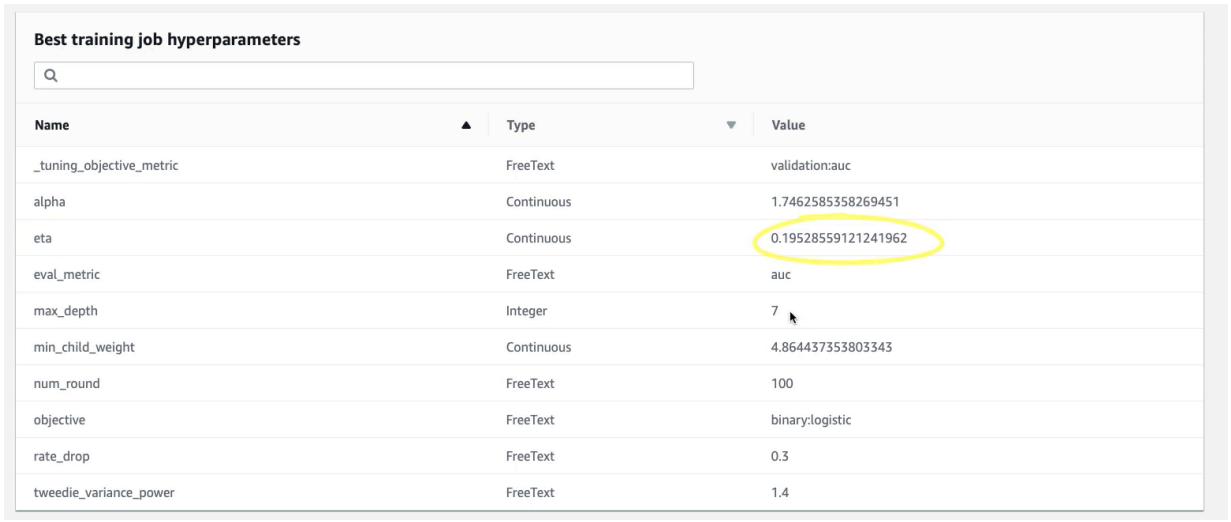


When they complete, we keep going till we hit 20 total (or 30mn)

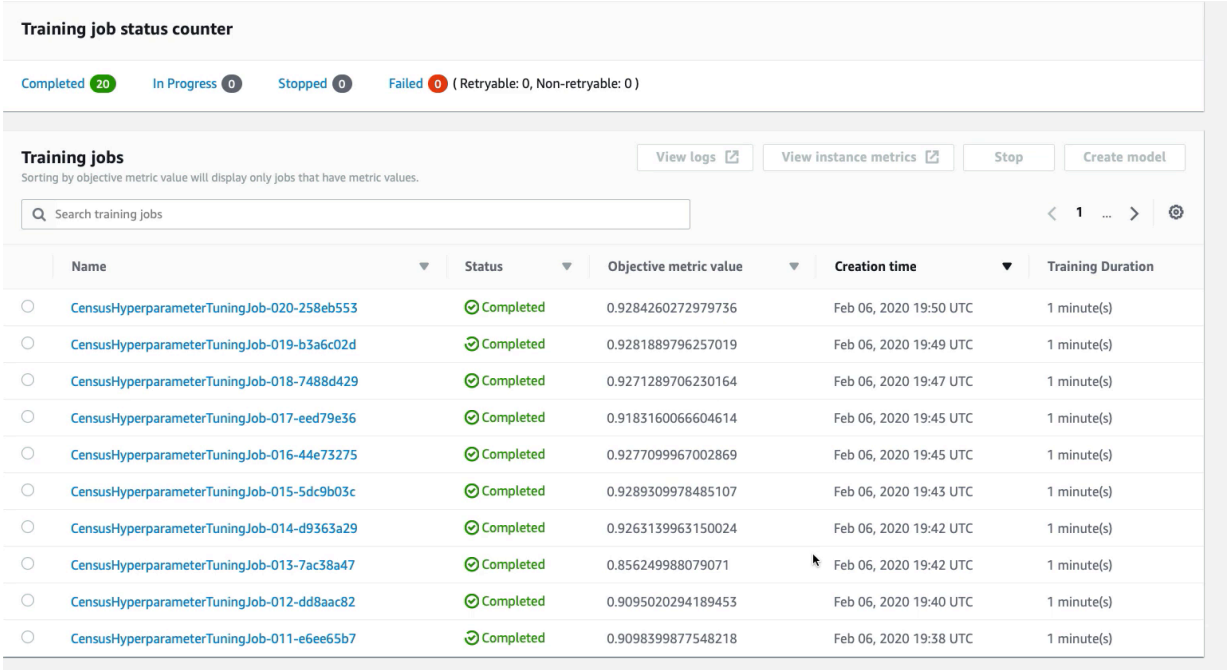
3 completed and we can see the AUC evaluation metric





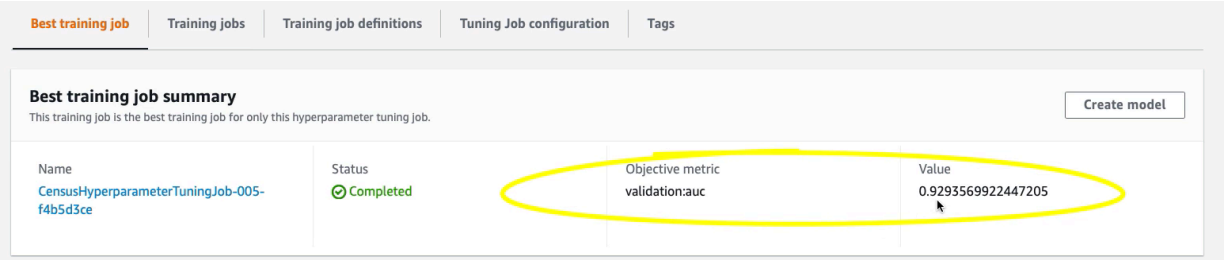


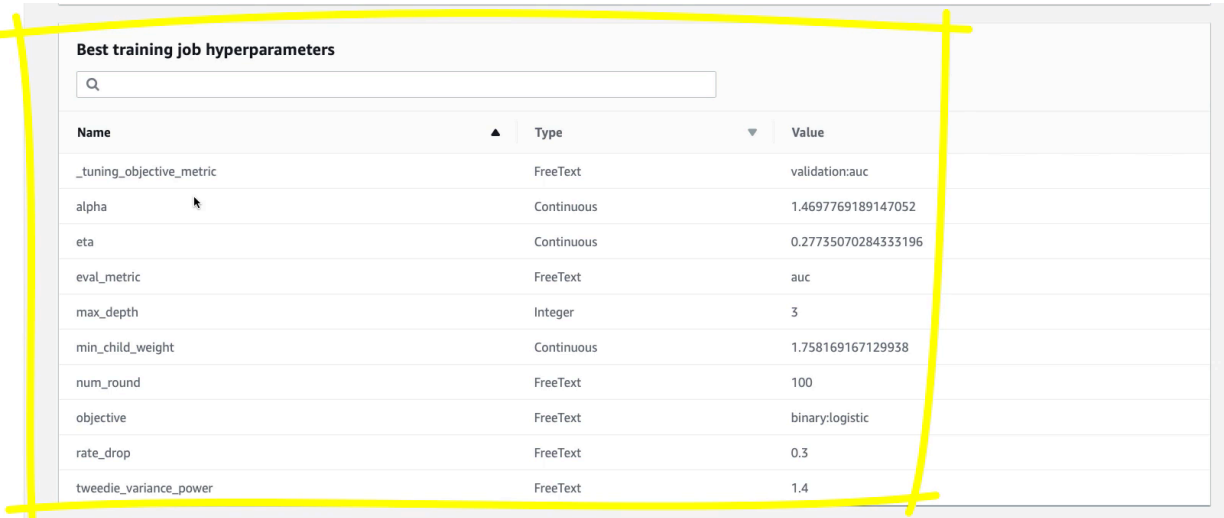
Hyper parameter tuning job completed - run all 20 training jobs



Each one used set of metrics, using the Bayesian technique

Best training job:





We can now use these hyper parameter values as our values and re-train our model and see accuracy/confusion matrix

jupyter train-with-tuned-hyperparameters Last Checkpoint: a few seconds ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python [conda root]

Code

In [1]:
```
# import libraries
import boto3, re, sys, math, json, os, sagemaker, urllib.request
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
# each region has its XGBoost container
containers = {'us-west-2': '433757028032.dkr.ecr.us-west-2.amazonaws.com/xgboost:latest',
              'us-east-1': '811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest',
              'us-east-2': '825641698319.dkr.ecr.us-east-2.amazonaws.com/xgboost:latest',
              'eu-west-1': '685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest'}
my_region = boto3.session.Session().region_name # set the region of the instance
print("Great! - your SageMaker Instance is in the " + my_region + " region. You will use the " + containers[my_region]
```

```
Great! - your SageMaker Instance is in the us-east-1 region. You will use the 811284229777.dkr.ecr.us-east-1.amazonaw
s.com/xgboost:latest container for your SageMaker endpoint to make inference requests.
```

In [2]:
```
# Download from your S3 bucket the census data CSV file based on the publically available census data from the ML repos
from io import StringIO
s3 = boto3.resource('s3')
bucket_name = 'train-census-earnings-xgboost' # place the adult_census.csv file in a bucket in your account
object_key = 'adult_census.csv'

# Load the data into a pandas dataframe

csv_obj = s3.Object(bucket_name, object_key)
csv_string = csv_obj.get()['Body'].read().decode('utf-8')

raw_data = pd.read_csv(StringIO(csv_string))
raw_data.head()
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	y_no	y_yes
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	0	1
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	0	1
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	0	1
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	0	1
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	0	1

In [3]:
```
model_data = pd.get_dummies(raw_data)
model_data.head()
```

Out[3]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	y_no	y_yes	workclass_0	workclass_Federal-gov	...	native-country_Portugal	native-country_Puerto-Rico	native-country_Scotland	native-country_South	native-country_Taiwan	native-country_Thailand	Tr
0	39	77516	13	2174	0	40	0	1	0	0	...	0	0	0	0	0	0	
1	50	83311	13	0	0	13	0	1	0	0	...	0	0	0	0	0	0	
2	38	215646	9	0	0	40	0	1	0	0	...	0	0	0	0	0	0	
3	53	234721	7	0	0	40	0	1	0	0	...	0	0	0	0	0	0	
4	28	338409	13	0	0	40	0	1	0	0	...	0	0	0	0	0	0	

5 rows × 110 columns

In [4]:
```
# Randomize the data and split it between train and test datasets on a 70% 30% split respectively
train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
print(train_data.shape, test_data.shape)
```

```
(22792, 110) (9768, 110)
```

In [5]:
```
# Reformat the header and first column of the training data,
# save the new train dataset to your S3 bucket as train.csv and load the data from the S3 bucket
pd.concat([train_data['y_yes'], train_data.drop(['y_no', 'y_yes'], axis=1)], axis=1).to_csv('train.csv', index=False, h
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train/train.csv')).upload_file('train.c
s3_input_train = sagemaker.s3_input(s3_data='s3://{}/{}/train'.format(bucket_name, prefix), content_type='csv')
```

Now enter the tuned hyper parameters

In [6]:
```
# Set up the SageMaker session, create an instance of the XGBoost model (an estimator),
# and define the model's hyperparameters
session_sm = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(containers[my_region],
                                    role,
                                    train_instance_count=1,
                                    train_instance_type='ml.m4.xlarge',
                                    output_path='s3://{}/{}/output'.format(bucket_name, prefix),
                                    sagemaker_session=session_sm)

xgb.set_hyperparameters(alpha=1.4697769189147052,
                        rate_drop=0.3,
                        tweedie_variance_power=1.4,
                        max_depth=3,
                        eta=0.27735070284333196,
                        min_child_weight=1.758169167129938,
                        objective='binary:logistic',
                        num_round=100)
```

In [7]:
```
# After the data is loaded and the XGBoost estimator is set up,
# train the model using gradient optimization on a ml.m4.xlarge instance
xgb.fit({'train': s3_input_train})
```

```
2020-02-06 22:59:09 Starting - Starting the training job...
2020-02-06 22:59:10 Starting - Launching requested ML instances......
2020-02-06 23:00:24 Starting - Preparing the instances for training......
2020-02-06 23:01:27 Downloading - Downloading input data
2020-02-06 23:01:27 Training - Downloading the training image.............................
2020-02-06 23:06:41 Uploading - Uploading generated training model
2020-02-06 23:06:41 Completed - Training job completed
Arguments: train
[2020-02-06:23:06:26:INFO] Running standalone xgboost training.
[2020-02-06:23:06:26:INFO] Path /opt/ml/input/data/validation does not exist!
[2020-02-06:23:06:26:INFO] File size need to be processed in the node: 4.91mb. Available memory size in the node: 849
9.48mb
[2020-02-06:23:06:26:INFO] Determined delimiter of CSV input is ','
[23:06:26] S3DistributionType set as FullyReplicated
[23:06:27] 22792x108 matrix with 2461536 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delim
iter=,
[23:06:27] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[0]#011train-error:0.157731
```

```
[23:06:30] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10 extra nodes, 0 pruned nodes, max_depth=3
[97]#011train-error:0.121973
[23:06:30] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12 extra nodes, 0 pruned nodes, max_depth=3
[98]#011train-error:0.121929
[23:06:30] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14 extra nodes, 0 pruned nodes, max_depth=3
[99]#011train-error:0.121709

Training seconds: 326
Billable seconds: 326
```

In [8]:
```
# Deploy your model and create an endpoint that you can access
xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')
```

```
----------------!
```

In [9]:
```
# Predict whether census participants in the test dataset earned more than 50K
test_data_array = test_data.drop(['y_no', 'y_yes'], axis=1).values #load the data into an array
xgb_predictor.content_type = 'text/csv' # set the data type for an inference
xgb_predictor.serializer = csv_serializer # set the serializer type
predictions = xgb_predictor.predict(test_data_array).decode('utf-8') # predict!
predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array
print(predictions_array.shape)
```

```
(9768,)
```

We can now see the new accuracy and confusion matrix

In [10]:
```
# Evaluate the performance and accuracy of the model
cm = pd.crosstab(index=test_data['y_yes'],
                 columns=np.round(predictions_array),
                 rownames=['Observed'],
                 colnames=['Predicted'])

tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1];
p = (tp+tn)/(tp+tn+fp+fn)*100

print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
print("{0:<15}{1:<15}{2:>8}".format("Predicted", "Under 50K", "Over 50K"))
print("Observed")
print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("Under 50K", tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("Over 50K", fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))
```

```
Overall Classification Rate: 87.4%

Predicted      Under 50K      Over 50K
Observed
Under 50K      80% (1544)     11% (856)
Over 50K        20% (377)      89% (6991)
```

In [ ]:
```
sagemaker.Session().delete_endpoint(xgb_predictor.endpoint)
```

Under 50k accuracy is much higher than before
Over 50k is about the same