Train Model In sagemaker

#Upload Your Model to Amazon S3
```
import boto3
import sagemaker
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import io
import os
import sys
import time
import json
from IPython.display import display
from time import strftime, gmtime
from sagemaker.inputs import TrainingInput
from sagemaker.serializers import CSVSerializer
from sagemaker import get_execution_role

region = boto3.Session().region_name
bucket = "uip-datalake-bucket-prod"
session = sagemaker.Session(default_bucket=bucket)
prefix = "gdso/datascience/users/ggupta1"
model_path =session.upload_data(path='pre_trained_models/best_model.pkl', bucket=bucket,
key_prefix=prefix)
role = get_execution_role()
tagValue = [{"Key": "app_group", "Value": "uip_gdso_ds"}]

train_data =pd.concat([y_train,x_train],axis=1)
train_file = "train_data_xgb.csv"
train_data.to_csv(train_file, index=False, header=True)
train_data_s3_path = session.upload_data(path=train_file, key_prefix=prefix + "/train")
print("Train data uploaded to: " + train_data_s3_path)

test_file = "test_data_xgb.csv"
y_train.to_csv(test_file, index=False, header=False)

test_data_s3_path = session.upload_data(path=test_file, key_prefix=prefix + "/test")
print("Test data uploaded to: " + test_data_s3_path)

s3_input_train =
TrainingInput(s3_data="s3://uip-datalake-bucket-prod/gdso/datascience/users/ggupta1/train/trai
n_data_xgb.csv", content_type="csv")
```

```
container = sagemaker.image_uris.retrieve("xgboost", region, "1.7-1")
display(container)


sess = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(container,role,
    instance_count=1,
    instance_type="ml.m4.xlarge",
    output_path="s3://{}/{}/output".format(bucket,
prefix),sagemaker_session=sess,tags=tagValue)

xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0.8
,verbosity=0,objective="binary:logistic",num_round=100)

xgb.fit({"train": s3_input_train})


##############

Create Endpoint
##############################


sagemaker_client = boto3.client('sagemaker')
role = get_execution_role()

# Create a Model Package Group
model_package_group_name = "spp12dec"
response = sagemaker_client.create_model_package_group(
    ModelPackageGroupName=model_package_group_name,
    ModelPackageGroupDescription="SPP+",
    Tags=tagValue
)


# After training, your model data is saved in S3, you need its location
model_artifacts = xgb.model_data


#Create a Model Package
model_package_response = sagemaker_client.create_model_package(
    ModelPackageDescription="SPP Model deployment",
    InferenceSpecification={
        "Containers": [
```

```
            {
                "Image": container,
                "ModelDataUrl": model_artifacts,
            }
        ],
        "SupportedTransformInstanceTypes": ["ml.m4.xlarge"],
        "SupportedRealtimeInferenceInstanceTypes": ["ml.m4.xlarge"],
        "SupportedContentTypes": ["text/csv"],
        "SupportedResponseMIMETypes": ["text/csv"],
    },
    ModelPackageGroupName=model_package_group_name,
    ModelApprovalStatus="Approved"
)

print("Model Package ARN : ", model_package_response['ModelPackageArn'])


# Create Endpoint
client = boto3.client('sagemaker')
client.delete_model(ModelName='xgboost-spp')
# Create a model
create_model_response = client.create_model(ModelName='xgboost-spp',
    PrimaryContainer={ 'ModelPackageName':
model_package_response['ModelPackageArn'],},ExecutionRoleArn=role,Tags=tagValue )

# Deploy the model to an endpoint
client.create_endpoint_config(EndpointConfigName='spp12dec',ProductionVariants=[
        {'InstanceType': 'ml.m4.xlarge','ModelName': 'xgboost-spp',
            'InitialInstanceCount': 1,'VariantName': 'AllTraffic',},],Tags=tagValue )

client.create_endpoint(EndpointName='spp12',EndpointConfigName='spp12dec',Tags=tagValue
)


#########################
Call Model
#########################

# Prediction from Endpoint

csv_buffer = io.StringIO()
x_test.head(10).to_csv(csv_buffer, header=False, index=False)
csv_payload = csv_buffer.getvalue()
runtime = boto3.client('runtime.sagemaker')
```

```
endpoint_name = 'spp7'

response = runtime.invoke_endpoint(EndpointName=endpoint_name,
                                   ContentType='text/csv',
                                   Body=csv_payload)

result__ = response['Body'].read().decode('utf-8')
```