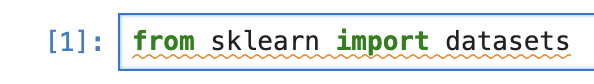
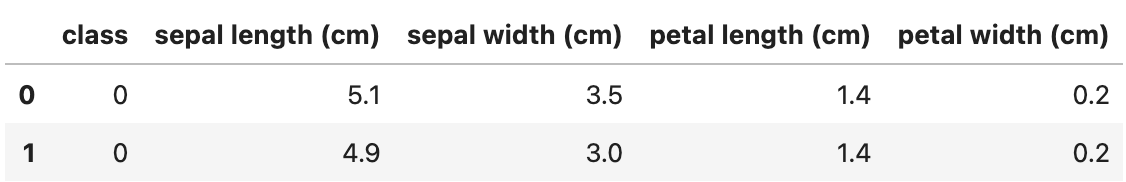
Lab 11: Machine Learning

## Start your SageMaker Notebook 0.5 points

* Let’s read the Overview on Amazon SageMaker AI Service @ [What is Amazon SageMaker AI?](https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html)
* Go to the Amazon SageMaker AI console (AWS Console > Services > SageMaker AI)
* Select **United States (N. Virginia)** from the drop-down region list in the right upper corner
* Create a **Notebook instance** (from Applications and IDEs) with default settings.
  + For IAM role, click on drop down, select *Create a new role* -> *Crete role* with default settings.
* Go to IAM -> Roles:
  + Search for your role Sagemaker-ExecutionRole and **save ARN** for later.
  + Add Permissions -> Attach policies -> check AdministratorAccess -> Add permissions
* Go back to Notebook -> Wait ~3 minutes for **InService** status and then click on *Open JupyterLab.*
* Open a Launcher tab (click on ‘+’ sign) and select Notebook - *conda\_python3*
* Rename the Notebook (we don’t like the *Untitled* notebooks)
* Run!git clone <https://github.com/AlbertJunior/csyi-ml> in the first cell

## Prepare Training data 1.0 point

* Import the iris dataset from sklearn
  + Run from sklearn import datasets code in the second cell (Shift+Enter)
  + Run iris = datasets.load\_iris() and explore the iris object (e.g. type(iris); dir(iris); iris.DESCR; iris.target\_names; iris.feature\_names; iris.data)
* Load data into a DataFrame from pandas
  + Run import pandas as pd to import pandas library
  + Find **x, y, z** using [documentation](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html). See the expected result below:
    - df = pd.DataFrame(data=iris.**x**, columns=iris.**y**)
    - df["class"] = pd.Series(iris.**z**)
    - df = df[[df.columns[-1]] + list(df.columns[:-1])] #class column first
    - print(df.head())



* How many instances are in the dataset in total? What about each class? (use [filtering](https://www.geeksforgeeks.org/ways-to-filter-pandas-dataframe-by-column-values/) and df.shape)
* Spit dataset into training and test datasets using [train\_test\_split](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) from sklearn.
* Save inference data (without label) for later: inference\_df = test\_df.drop(columns=["class"])
* Save datasets in S3
  + Create an S3 bucket (with [AWS CLI](https://docs.aws.amazon.com/AmazonS3/latest/userguide/create-bucket-overview.html))
  + Save training and test datasets in csv files
    - train\_df.to\_csv("train\_iris.csv", index=False, header=None)
    - test\_df.to\_csv("test\_iris.csv", index=False, header=None)
    - inference\_df.to\_csv("inference\_iris.csv", index=False, header=None)
  + Copy them to your S3 bucket using aws s3 cp source target [Documentation](https://docs.aws.amazon.com/cli/latest/reference/s3/cp.html)
* Prepare TrainingInput objects for Training phase:

from sagemaker.inputs import TrainingInput

train\_input = TrainingInput("s3://**[your\_bucket\_name]/**train\_iris.csv", content\_type="csv")

validation\_input = TrainingInput("s3://**[your\_bucket\_name]/**test\_iris.csv", content\_type="csv")

## Training Classification Model. 4 methods 2.0 points

* **1. Custom Training script**
  + Run the csyi-ml/script.py file in a cell, to train a RandomForest Model. Find the right parameters (check the script *script.py*).
    - !python csyi-ml/script.py --n\_estimators 100 --test-file test\_iris.csv --model-dir ./ etc
* **2. SageMaker Training job**
  + Complete the hyperparameters and the parameters to train with one instance of type "ml.t3.large", max run time of 30 minutes, max wait time of 1 hour. Save model artifacts in s3://**[your\_bucket\_name]/**model (check *script.py* for hyperparameters and [SKLearn Doc](https://sagemaker.readthedocs.io/en/stable/frameworks/sklearn/sagemaker.sklearn.html#scikit-learn-estimator) for parameters. See [EstimatorBase class](https://sagemaker.readthedocs.io/en/stable/api/training/estimators.html#sagemaker.estimator.Framework)).

from sagemaker.sklearn.estimator import SKLearn

sklearn\_estimator = SKLearn(

entry\_point="csyi-ml/script.py",

role=[**your\_execution\_role\_ARN\_created\_at\_step\_A**],

framework\_version="0.23-1",

hyperparameters={  
 **...**

"random\_state: 0,  
 "train-file": "train\_iris.csv",

"test-file": "test\_iris.csv",

},

use\_spot\_instances = True

**...**

)

sklearn\_estimator.fit({"train": train\_input, "test": validation\_input}, logs="All")

* + Go to SageMaker AI Service -> Training -> Training Jobs. Check the total training time.
* **3. Custom Model Container**
  + Create **build role**. Go to IAM service from console -> Roles -> Create role -> Custom trust policy -> Replace: "Principal": {"Service": "codebuild.amazonaws.com"},

-> Next -> check AdministratorAccess -> Next -> Fill Role name -> Create role

* + Wait 3m, go back to notebook and run:

!pip install sagemaker-studio-image-build

!cd csyi-ml/Algo\_Container && chmod +x RandomForest/train && chmod +x RandomForest/serve

!cd csyi-ml/Algo\_Container && sm-docker build . --role [**your\_build\_role\_name**] \  
 --repository "random-forest:latest" --bucket [**your\_bucket\_name**]

* + Go to CodeBuild from console and check build status. When finished you should see the image URI and SUCCEEDED in notebook.
  + Go to Elastic Container Registry from console -> Private registry -> Repositories
  + Copy the URI of your image and use it in your notebook to train a model:

import sagemaker as sage  
custom\_estimator = sage.estimator.Estimator([**your image URI**], [**your\_execution\_role**],   
 1, "ml.m5.large", hyperparameters**={…},**   
 use\_spot\_instances=True, max\_run=1800, max\_wait=3600,

output\_path="s3://**[your\_bucket\_name]**/model")

custom\_estimator.fit({"training": train\_input, "validation": validation\_input}, logs="All")

* **4. Jumpstart Model**

model\_img = sage.image\_uris.retrieve("xgboost", "us-east-1", "latest")

jumpstart\_estimator = sage.estimator.Estimator(model\_img, [**your\_execution\_role**],

1, "ml.m5.large", hyperparameters={

"num\_class":3, "max\_depth":5, "num\_round":10,

"objective":"multi:softmax"},

use\_spot\_instances=True, max\_run=1800, max\_wait=3600,

output\_path="s3://**[your\_bucket\_name]**/model")

jumpstart\_estimator.fit({"training": train\_input, "validation": validation\_input}, logs="All")

## Comparative analysis 0.5 point

* Go to SageMaker AI Service -> Training -> Training Jobs. Check validation metric for last job.
* Go to csyi-ml/Algo\_Container/RandomForest/train script and uncomment the validation metric print.
* Rebuild image as in **C. Custom Model Container** step
* Recreate the estimator by passing a new parameter metric\_definitions. Find the right regex for it according to [documentation](https://sagemaker.readthedocs.io/en/v2.42.1/api/training/estimators.html). Example:

metric\_definitions=[

{'Name': 'train:merror', 'Regex': 'Train\_merror=(.\*)?;'},

{'Name': 'validation:merror', 'Regex': 'Valid\_merror=(.\*)?;'}]

* Refit the model. How is the XGBoost *valdation:merror* compared RandomForest *validation:merror*?

## Batch transformation 0.5 point

* Create boto3 sagemaker clients as prerequisites:

import boto3

sgmkr\_clnt = boto3.client("sagemaker")

sgmkr\_rt = boto3.client("runtime.sagemaker")

* Create model with the artifacts saved at output\_path at step **C.Training Classification Model**

sgmkr\_clnt.create\_model( ModelName=**[choose\_a\_model\_name]**,

PrimaryContainer={

"Image": **[xgboost\_image]**,

"ModelDataUrl": "s3://**[your\_bucket\_name]**/model/**xgboost\_[choose\_one\_version]/**output/model.tar.gz"},

ExecutionRoleArn=**[your\_execution\_role]**)

* Go to SageMaker AI Service -> Inference -> Models. Check your model.
* Use your model to run on your inference data:

from sagemaker.transformer import Transformer

transformer = Transformer(model\_name=**[your\_model\_name]**,

instance\_count=1,

instance\_type="ml.m4.xlarge",

output\_path="s3://**[your\_bucket\_name]**/predictions/")

transformer.transform(data="s3://**[your\_bucket\_name]**/inference\_iris.csv",

data\_type="S3Prefix", content\_type="text/csv")

transformer.wait()

* Check in your S3 bucket, the new file with predictions. Save it locally (aws s3 cp) and look at the result.

## Deploy model 0.5 point

* Prepare names for endpoint\_config and endpoint as prerequisites:

from datetime import datetime

ep\_config\_name = "tmp-ep-config-" + datetime.today().strftime("%Y-%m-%d-%H-%M-%S-%f")

ep\_name = "tmp-ep-" + datetime.today().strftime("%Y-%m-%d-%H-%M-%S-%f")

* Create endpoint config:

sgmkr\_clnt.create\_endpoint\_config( EndpointConfigName=ep\_config\_name,

ProductionVariants=[{"VariantName": "version-1", "ModelName": **[your\_model\_name]**,

"InitialInstanceCount": 1, "InstanceType": "ml.m4.xlarge"},],)

* Go to SageMaker AI Service -> Inference -> Endpoint configurations. Check your endpoint config.
* Create endpoint:

sgmkr\_clnt.create\_endpoint(EndpointName=ep\_name, EndpointConfigName=ep\_config\_name)

* Go to SageMaker AI Service -> Inference -> Endpoints. Check your endpoint. Wait for it to be ready.
* Test your endpoint:

payload = "7.7, 3.0, 6.1, 2.3 \n 7.9, 3.8, 6.4, 2.1"

response = sgmkr\_rt.invoke\_endpoint(

EndpointName=ep\_name, ContentType="text/csv", Body=payload)

prediction = response["Body"].read().decode()

print(prediction)

* Delete your endpoint (important!):

response = sgmkr\_clnt.delete\_endpoint(EndpointName = endpoint\_name)

## Hyper tuning 1 point

* Try to make your RandomForest model better than the XGBoost one. Use Hypertuning to find the optimal parameters. See documentation [here](https://sagemaker.readthedocs.io/en/v2.42.1/api/training/tuner.html).