CITS5503 Lab8

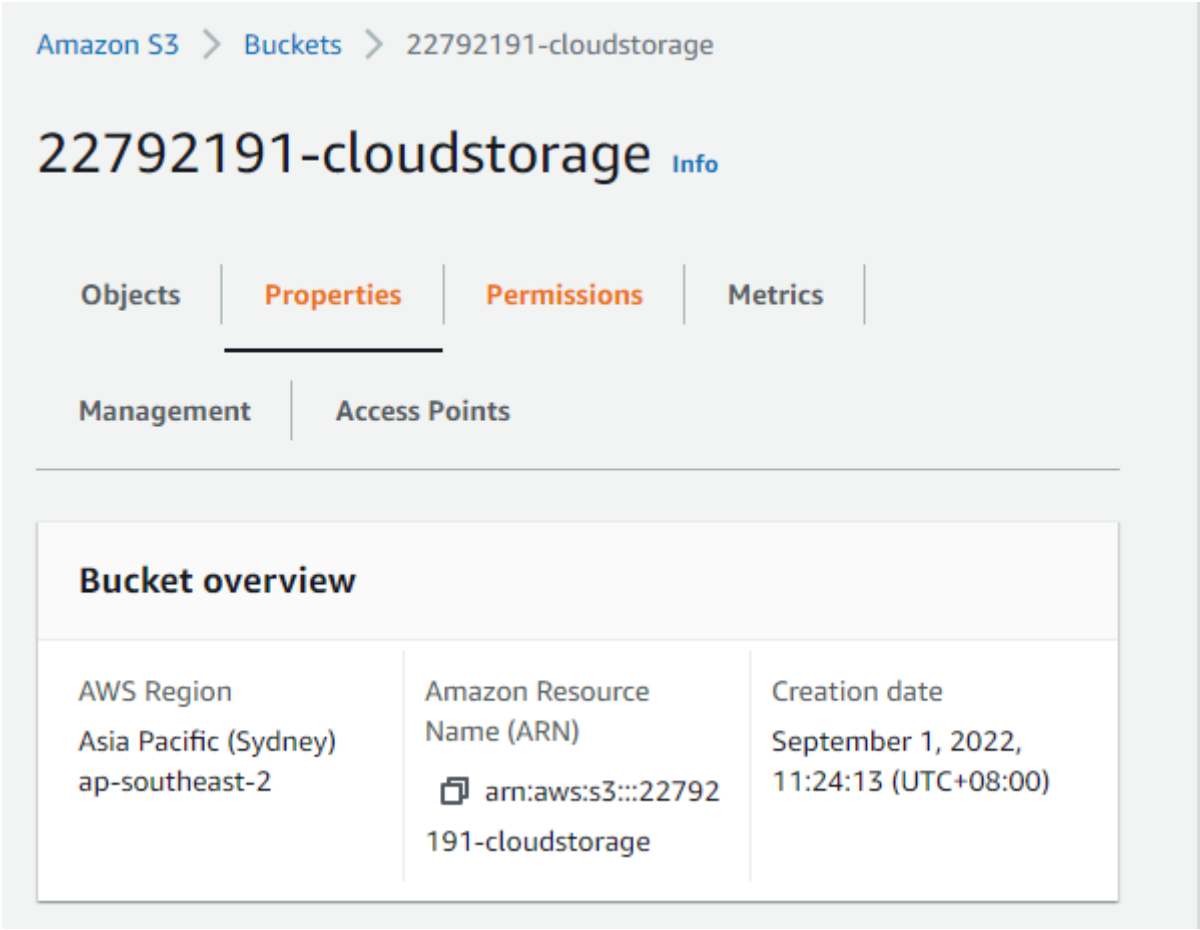
Wenxiao Zhang 22792191

Set Up Python Environment

```
moebuta@Lenovo-MoeBuTa:~/2022s2/cits5503/labs/lab8$ pip3 install sagemaker pandas ipykernel
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: sagemaker in /home/moebuta/.local/lib/python3.8/site-packages (2.110.0)
Requirement already satisfied: pandas in /home/moebuta/.local/lib/python3.8/site-packages (1.4.3)
Requirement already satisfied: ipykernel in /home/moebuta/.local/lib/python3.8/site-packages (6.15.1)
Requirement already satisfied: packaging>=20.0 in /home/moebuta/.local/lib/python3.8/site-packages (from sagemak
Requirement already satisfied: protobuf3-to-dict<1.0,>=0.1.5 in /home/moebuta/.local/lib/python3.8/site-packages
```

Create a bucket

The properties of the created bucket is shown below:



Session preparation

add an student id and bucket name to prepare SageMaker session.

Prepare SageMaker session

```
import sagemaker
import boto3

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

region = 'ap-southeast-2'
smclient = boto3.Session().client("sagemaker")

iam = boto3.client('iam')
sagemaker_role = iam.get_role(RoleName='Role_AWS_SageMaker')['Role']['Arn']

student_id = "22792191"
bucket = '22792191-cloudstorage'
prefix = f"sagemaker/{student_id}-hpo-xgboost-dm"
```

[ ]

Download Dataset

Download Dataset

Please take some time to read about the data with more detail here Let's start by downloading the direct marketing

You can download the dataset manually or use the commands below. These commands should work for Linux and M

```
!wget -N https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
!unzip -o bank-additional.zip
```

✓ 2.3s

```
--2022-10-05 18:04:11--  https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 444572 (434K) [application/x-httpd-php]
Saving to: 'bank-additional.zip'

bank-additional.zip 100%[===================>] 434.15K   415KB/s    in 1.0s

2022-10-05 18:04:13 (415 KB/s) - 'bank-additional.zip' saved [444572/444572]

Archive:  bank-additional.zip
   creating: bank-additional/
  inflating: bank-additional/.DS_Store
   creating: __MACOSX/
   creating: __MACOSX/bank-additional/
  inflating: __MACOSX/bank-additional/._.DS_Store
  inflating: bank-additional/.Rhistory
  inflating: bank-additional/bank-additional-full.csv
  inflating: bank-additional/bank-additional-names.txt
```

Run the rest of the code

for the variables of the dataset in the screenshort, for example, `age`, `duration` are numerical variables, `marital`, `housing` are categorical variables.

Now lets read this into a Pandas data frame and take a look at the data.

```
data = pd.read_csv("./bank-additional/bank-additional-full.csv", sep=";")
pd.set_option("display.max_columns", 500)  # Make sure we can see all of the columns
pd.set_option("display.max_rows", 50)  # Keep the output on one page
data
```

✓ 0.1s Python

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.e
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
41183	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	334	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	
41184	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	383	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	
41185	56	retired	married	university.degree	no	yes	no	cellular	nov	fri	189	2	999	0	nonexistent	-1.1	94.767	-50.8	1.028	
41186	44	technician	married	professional.course	no	no	no	cellular	nov	fri	442	1	999	0	nonexistent	-1.1	94.767	-50.8	1.028	
41187	74	retired	married	professional.course	no	yes	no	cellular	nov	fri	239	3	999	1	failure	-1.1	94.767	-50.8	1.028	

```
data["no_previous_contact"] = np.where(
    data["pdays"] == 999, 1, 0
)  # Indicator variable to capture when pdays takes a value of 999
data["not_working"] = np.where(
    np.in1d(data["job"], ["student", "retired", "unemployed"]), 1, 0
)  # Indicator for individuals not actively employed
model_data = pd.get_dummies(data)  # Convert categorical variables to sets of indicators
model_data
```

✓ 0.9s

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	no_previous_contact	not_w
0	56	261	1	999	0	1.1	93.994	-36.4	4.857	5191.0	1	
1	57	149	1	999	0	1.1	93.994	-36.4	4.857	5191.0	1	
2	37	226	1	999	0	1.1	93.994	-36.4	4.857	5191.0	1	
3	40	151	1	999	0	1.1	93.994	-36.4	4.857	5191.0	1	
4	56	307	1	999	0	1.1	93.994	-36.4	4.857	5191.0	1	
...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	334	1	999	0	-1.1	94.767	-50.8	1.028	4963.6	1	
41184	46	383	1	999	0	-1.1	94.767	-50.8	1.028	4963.6	1	
41185	56	189	2	999	0	-1.1	94.767	-50.8	1.028	4963.6	1	
41186	44	442	1	999	0	-1.1	94.767	-50.8	1.028	4963.6	1	
41187	74	239	3	999	1	-1.1	94.767	-50.8	1.028	4963.6	1	

41188 rows × 67 columns

Let's remove the economic features and duration from our data as they would need to be forecasted with high precision to use as inputs in future predictions.

```
model_data = model_data.drop(
    ["duration", "emp.var.rate", "cons.price.idx", "cons.conf.idx", "euribor3m", "nr.employed"],
    axis=1,
)
```

[5] ✓ 0.2s Python

```
model_data
```

[6] ✓ 0.7s Python

	age	campaign	pdays	previous	no_previous_contact	not_working	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	job_management	job_retired	job_self-employed	job_services	job_student	job_technician	job_
0	56	1	999	0	1	0	0	0	0	1	0	0	0	0	0	0	
1	57	1	999	0	1	0	0	0	0	0	0	0	0	1	0	0	
2	37	1	999	0	1	0	0	0	0	0	0	0	0	1	0	0	
3	40	1	999	0	1	0	1	0	0	0	0	0	0	0	0	0	
4	56	1	999	0	1	0	0	0	0	0	0	0	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
41183	73	1	999	0	1	1	0	0	0	0	0	1	0	0	0	0	
41184	46	1	999	0	1	0	0	1	0	0	0	0	0	0	0	0	
41185	56	2	999	0	1	1	0	0	0	0	0	1	0	0	0	0	
41186	44	1	999	0	1	0	0	0	0	0	0	0	0	0	0	1	
41187	74	3	999	1	1	1	0	0	0	0	0	1	0	0	0	0	

41188 rows × 61 columns

Split Data into training, validation and test

We'll then split the dataset into training (70%), validation (20%), and test (10%) datasets and convert the datasets to the right format the algorithm expects. We will use training and validation datasets during training. Test dataset will be used to evaluate model performance after it is deployed to an endpoint.

Amazon SageMaker's XGBoost algorithm expects data in the libSVM or CSV data format. For this lab, we'll stick to CSV. Note that the first column must be the target variable and the CSV should not include headers. Also, notice that although repetitive it's easier to do this after the train|validation|test split rather than before. This avoids any misalignment issues due to random reordering.

```
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
)
```

✓ 0.2s Python

Now we'll copy the file to S3 for Amazon SageMaker training to pickup.

```
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("validation.csv")
```

[8] ✓ 1.7s

Setup Hyperparameter Optimization

```
from time import gmtime, strftime, sleep

# Names have to be unique. You will get an error if you reuse the same name
tuning_job_name = f"{student_id}-xgboost-tuningjob-01"

print(tuning_job_name)

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

[9] ✓ 0.5s

```
22792191-xgboost-tuningjob-01
```

While Training you can take screenshots of the jobs you just launched on SageMaker-> Training -> Hyperparameter tuning jobs

```
#Launch Hyperparameter Tuning Job
smclient.create_hyper_parameter_tuning_job(
    HyperParameterTuningJobName=tuning_job_name,
    HyperParameterTuningJobConfig=tuning_job_config,
    TrainingJobDefinition=training_job_definition,
)
```

11] ⊗ 0.8s Python

```
---------------------------------------------------------------------------
ResourceLimitExceeded                     Traceback (most recent call last)
/home/moebuta/2022s2/cits5503/labs/lab8/LabAI.ipynb Cell 29 in <cell line: 2>()
      1 #Launch Hyperparameter Tuning Job
----> 2 smclient.create_hyper_parameter_tuning_job(
      3     HyperParameterTuningJobName=tuning_job_name,
      4     HyperParameterTuningJobConfig=tuning_job_config,
      5     TrainingJobDefinition=training_job_definition,
      6 )

File ~/.local/lib/python3.8/site-packages/botocore/client.py:508, in ClientCreator._create_api_method.<locals>._api_call(self, *args, **kwargs)
    504     raise TypeError(
    505         f"{py_operation_name}() only accepts keyword arguments."
    506     )
    507 # The "self" in this scope is referring to the BaseClient.
--> 508 return self._make_api_call(operation_name, kwargs)

File ~/.local/lib/python3.8/site-packages/botocore/client.py:915, in BaseClient._make_api_call(self, operation_name, api_params)
    913     error_code = parsed_response.get("Error", {}).get("Code")
    914     error_class = self.exceptions.from_code(error_code)
--> 915     raise error_class(parsed_response, operation_name)
    916 else:
    917     return parsed_response

ResourceLimitExceeded: An error occurred (ResourceLimitExceeded) when calling the CreateHyperParameterTuningJob operation: The account-level service limit 'ml.m5.xlarge for training job usage' is 0 Instances, with current utilization of 0 Instances and a request delta of 2 Instances. Please contact AWS support to request an increase for this limit.
```

Files in the bucket

22792191-cloudstorage Info

Objects | Properties | Permissions | Metrics | Management

Access Points

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Copy S3 URI | Copy URL | Download

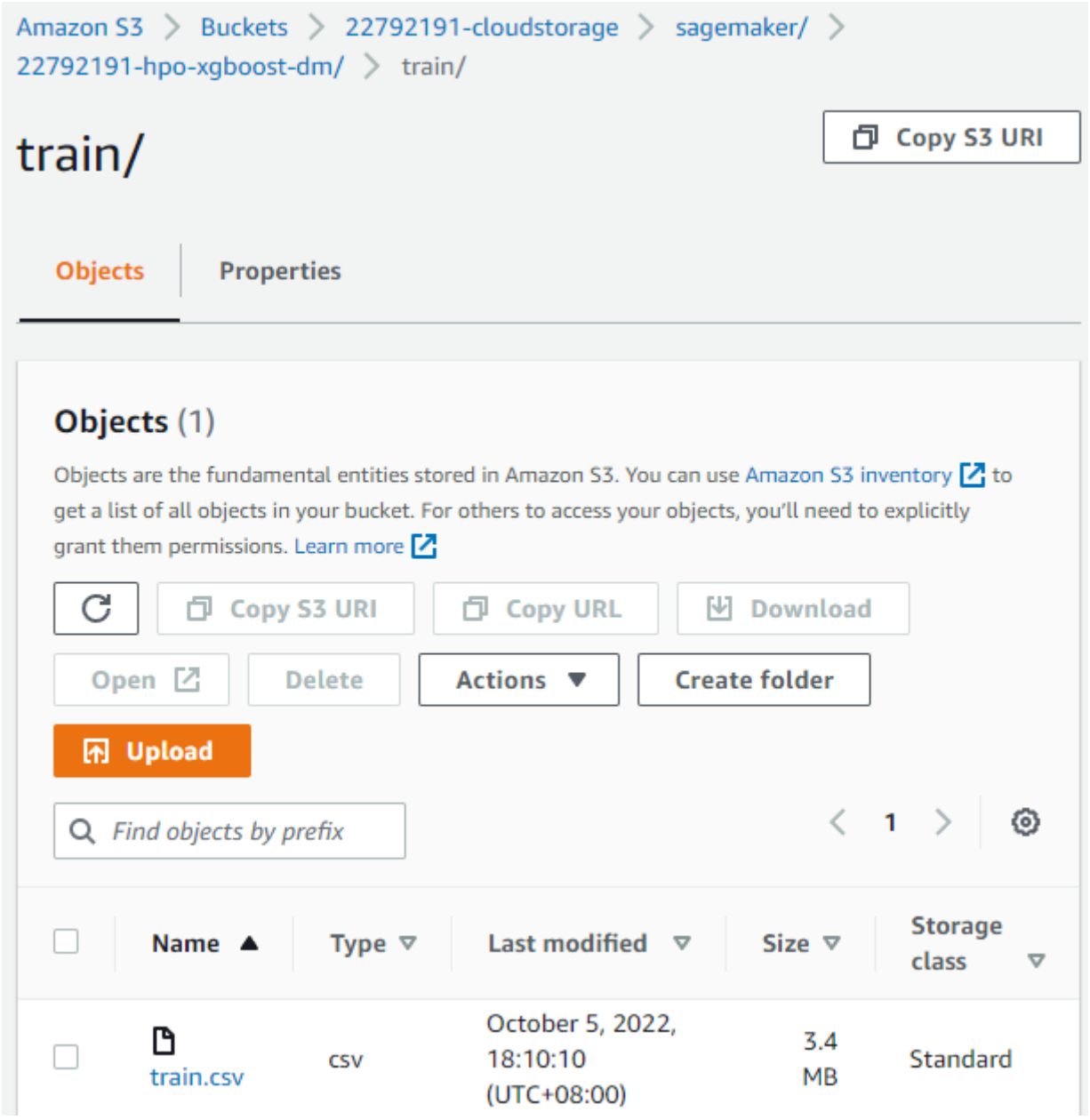
Open | Delete | Actions | Create folder

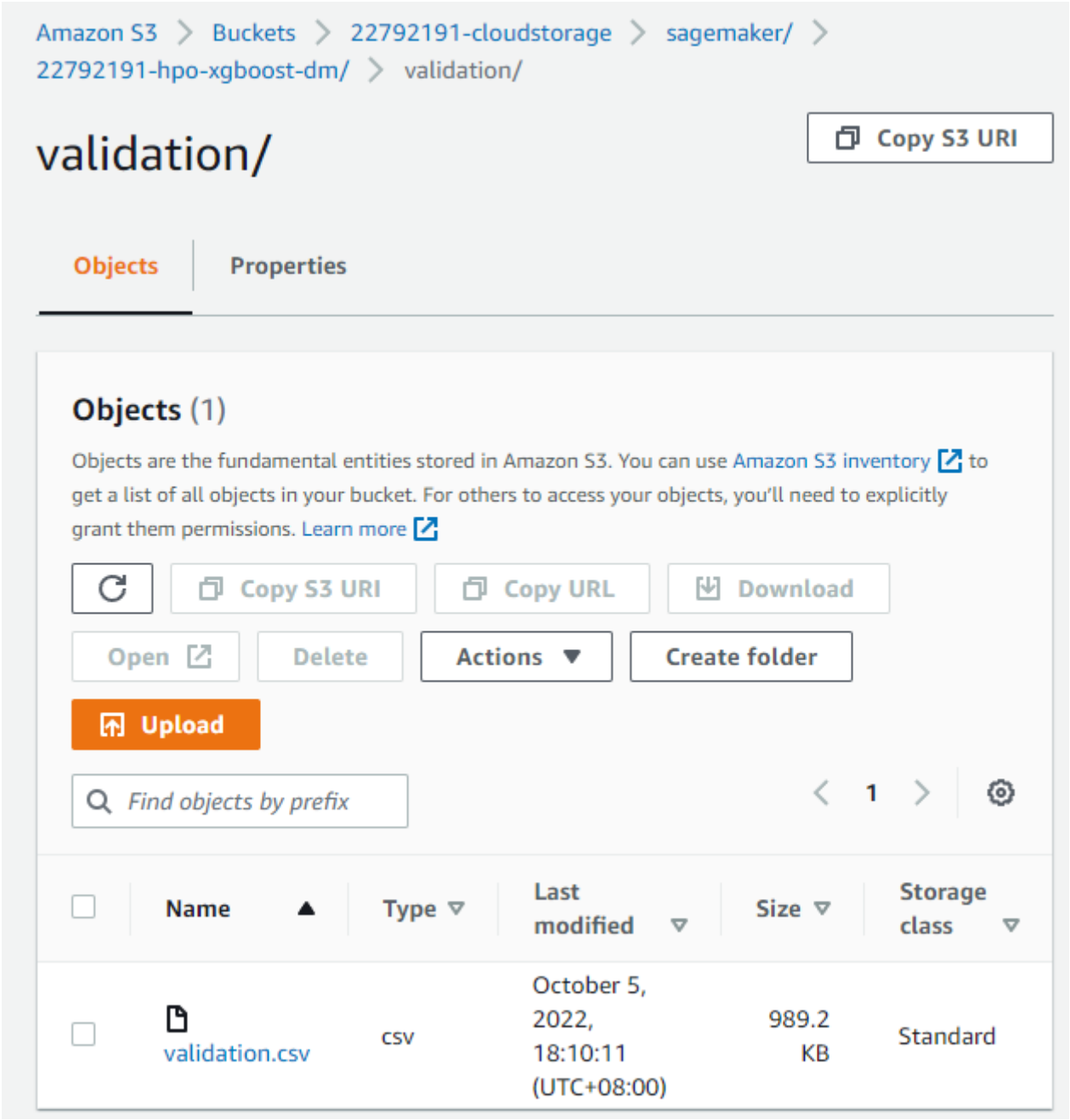
Upload

Find objects by prefix

1

Name	Type	Last modified	Size	Storage class
sagemaker/	Folder	-	-	-


Amazon S3 > Buckets > 22792191-cloudstorage > sagemaker/ >
22792191-hpo-xgboost-dm/ > train/
train/
Copy S3 URI
Objects
Properties
Objects (1)
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
train.csv
csv
October 5, 2022, 18:10:10 (UTC+08:00)
3.4 MB
Standard



Questions

a) In your S3 bucket, how many folders were created using the script (under the "{student_id}-hpo-xgboost-dm" folder)? List their name.

Two folders were created: `train` and `validation`.

b) How many Hyperparameter tuning jobs were created using the script?

Two Hyperparameter tuning jobs were created.

c) What metric was used in this script to evaluate the training results?

The `validation:auc` was used in this script, and the type is `maximize`.

d) What strategy was used in the tuning job?

`Bayesian` was used in the tuning job.