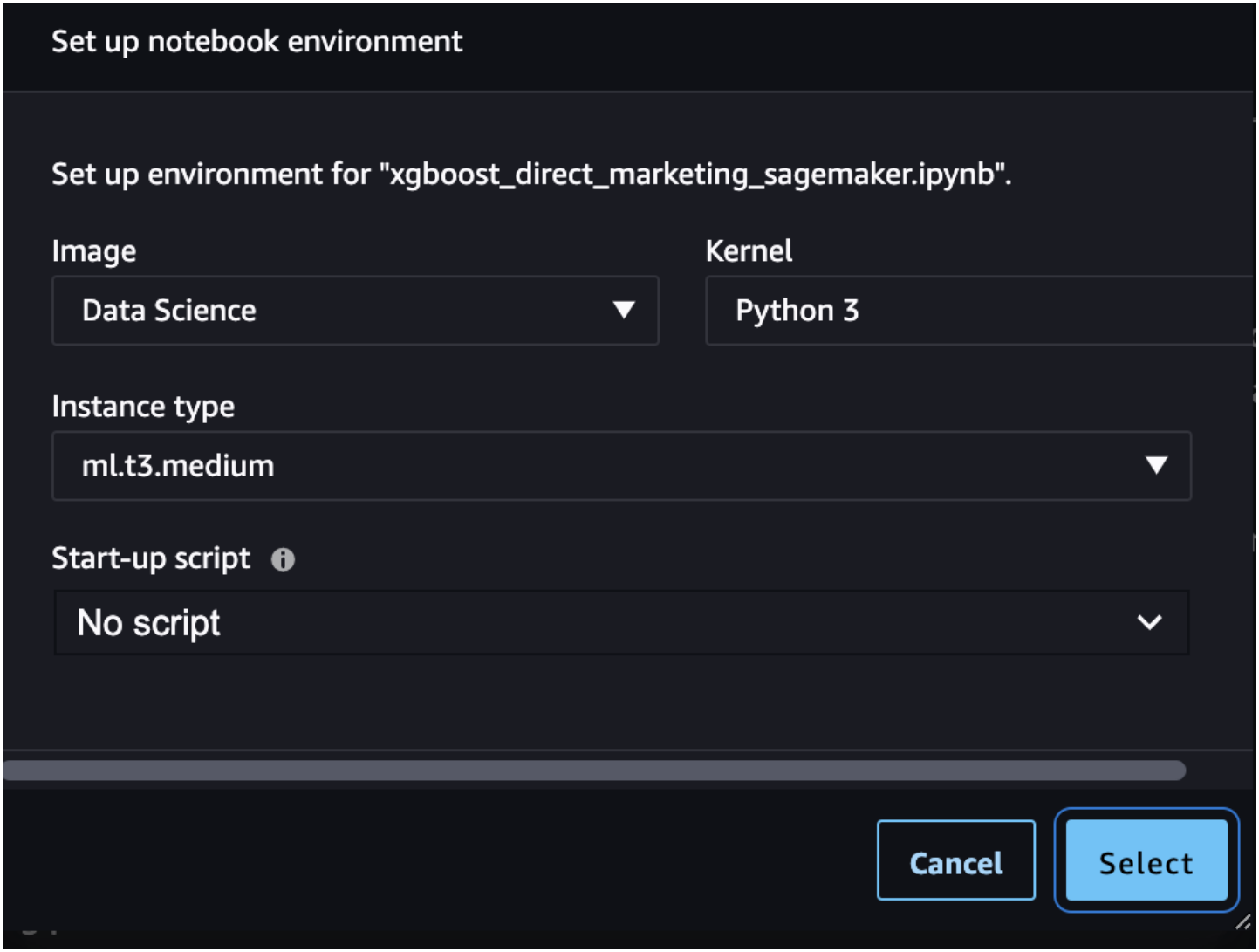
Option 2: Numpy and Pandas

⚠ DO NOT perform this part if you have already executed the feature engineering with Amazon Data Wrangler.

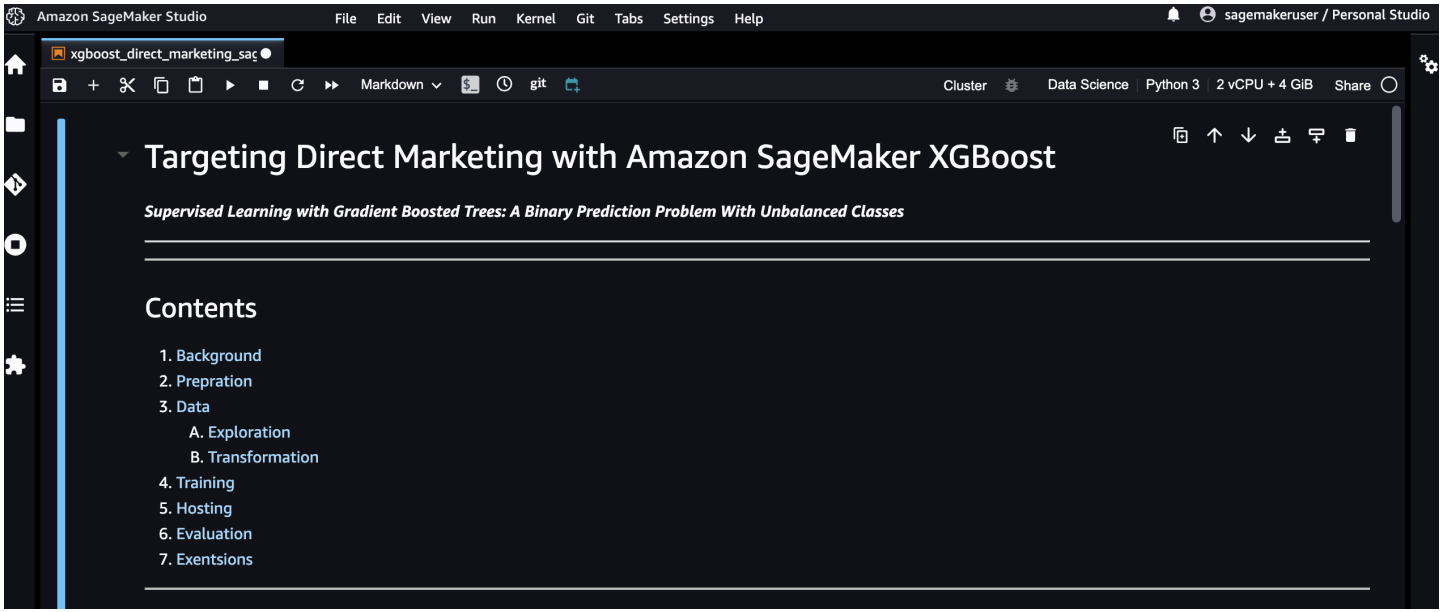
Data Preparation

In this step you will use your Amazon SageMaker Studio notebook to preprocess the data that you need to train your machine learning model.

1. Click on this “amazon-sagemaker-immersion-day” folder and then double click on the `xgboost_direct_marketing_sagemaker.ipynb` notebook.
2. If you are prompted to choose a Kernel, choose the “Python 3 (Data Science)” kernel and click “Select”.



3. You will then have the notebook opened. You can verify the Kernel CPU and Memory states on the top right of the notebook.



4. Execute the first four cells by pressing **Shift+Enter** in each of the cells. While the code runs, an * appears between the square brackets as pictured in the first screenshot to the right. After a few seconds, the code execution will complete, the * will be replaced with the number 1.

This code will import some libraries and define a few environment variables in your Jupyter notebook environment.

```
# cell 03
import sagemaker_datawrangler       # For interactive data prep widget
import numpy as np                  # For matrix operations and numerical processing
import pandas as pd                 # For munging tabular data
import matplotlib.pyplot as plt     # For charts and visualizations
from IPython.display import Image   # For displaying images in the notebook
from IPython.display import display # For displaying outputs in the notebook
from time import gmtime, strftime   # For labeling SageMaker models, endpoints, etc.
import sys                          # For writing outputs to notebook
import math                         # For ceiling function
import json                         # For parsing hosting outputs
import os                           # For manipulating filepath names
import sagemaker
import zipfile  # Amazon SageMaker's Python SDK provides many helper functions
```

5. Now, Let's download the dataset by running the 5th cell.

Data

Let's start by downloading the direct marketing dataset from the sample data s3 bucket.

[Moro et al., 2014] S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014

```
# cell 05
!wget https://sagemaker-sample-data-us-west-2.s3-us-west-2.amazonaws.com/autopilot/direct_marketing/bank-additional.zip

with zipfile.ZipFile('bank-additional.zip', 'r') as zip_ref:
    zip_ref.extractall('.')
```

6. In the next cell you will load the dataset into a pandas dataframe and utilize the `sagemaker_datawrangler` library to explore the data and understand the data distribution for each feature:

```
# cell 06
data = pd.read_csv('./bank-additional/bank-additional-full.csv')
pd.set_option('display.max_columns', 500)     # Make sure we can see all of the columns
pd.set_option('display.max_rows', 20)       # Keep the output on one page
data
```

View Pandas table | Use all of the rows in your dataset to compute the insights. Using the entire dataset might increase the time it takes to generate the insights.

The following table shows the first 1000 rows of your dataset. By default, Data Wrangler computes the insights for the first 10,000 rows of your dataset. Learn more.

age	job	marital	education	default
10+ Categories	10+ Categories	4 Categories	8 Categories	2 Categories
45	unknown	married	unknown	unknown
44	unknown	single	basic.9y	unknown
38	blue-collar	divorced	unknown	unknown
57	blue-collar	married	unknown	unknown
59	technician	married	unknown	unknown
34	blue-collar	married	unknown	unknown
38	technician	married	unknown	unknown
55	unknown	married	university.degree	unknown
42	blue-collar	married	unknown	unknown
31	blue-collar	married	basic.6y	unknown
47	unknown	married	unknown	unknown
28	unknown	single	unknown	unknown
53	admin	single	university.degree	unknown

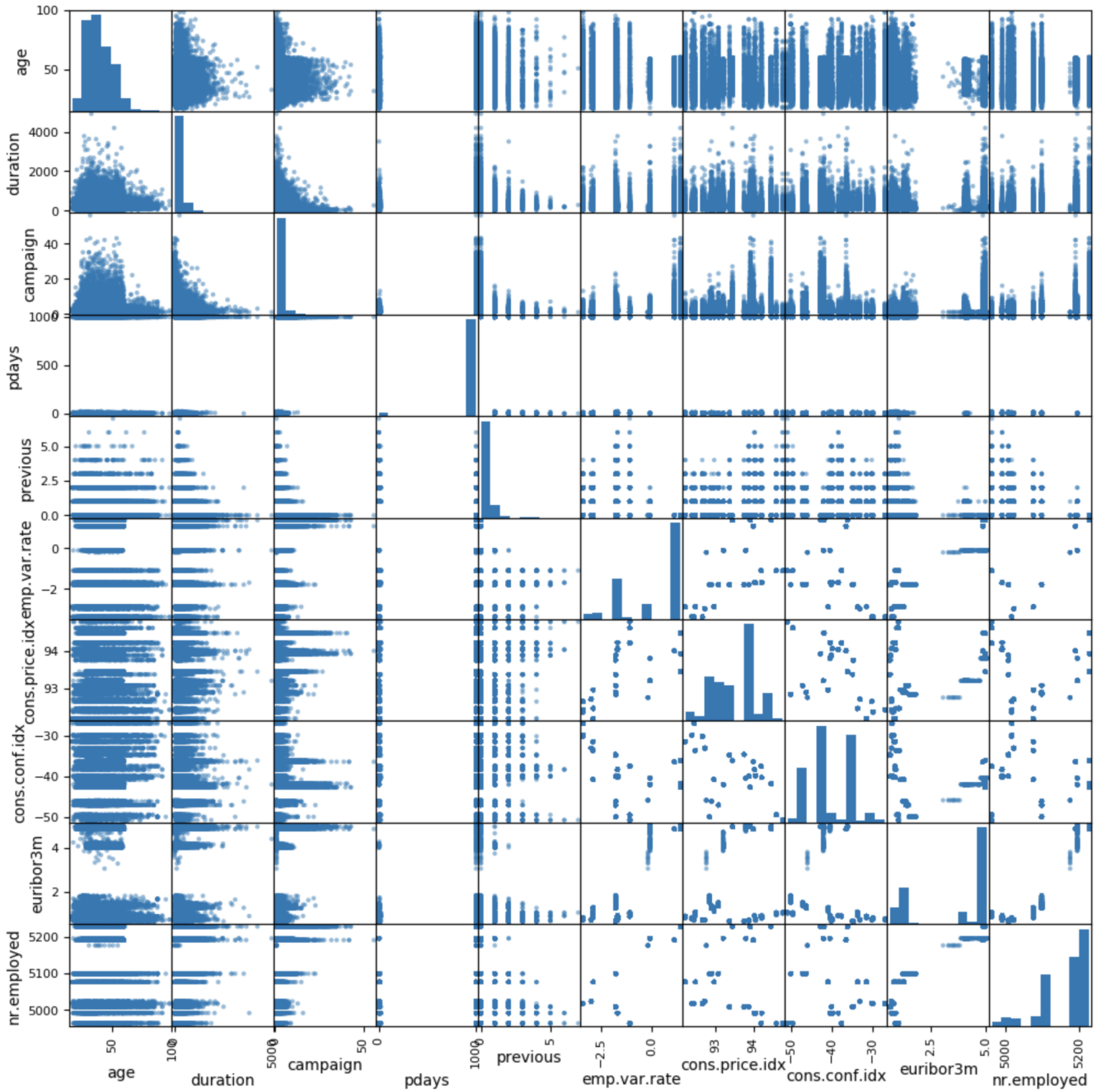
21 columns · 1,001 rows

7. Now let's look at how our features relate to the target that we are attempting to predict.

```
[6]: for column in data.select_dtypes(include=['object']).columns:
         if column != 'y':
             display(pd.crosstab(index=data[column], columns=data['y'], normalize='columns'))

     for column in data.select_dtypes(exclude=['object']).columns:
         print(column)
         hist = data[[column, 'y']].hist(by='y', bins=30)
         plt.show()
```

8. Now let's look at how our features relate to one another



9. Now we will convert all the categorical variables using one hot encoding

```
# cell 09

# Note: These transformations can be done through the graphical widget that we generated above. The data prep widget will automatically generate code for transformations that you do.
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
```

```
# cell 10
model_data = pd.get_dummies(data)                                                                  # Convert categorical variables to sets of indicators
```

10. We will drop few features which are not required

```
# cell 11
model_data = model_data.drop(['duration', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'], axis=1)
```

11. We will split our data set into 3 channels: train, test, validation set:

```
# cell 12
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))])   # Randomly sort the data ther
```

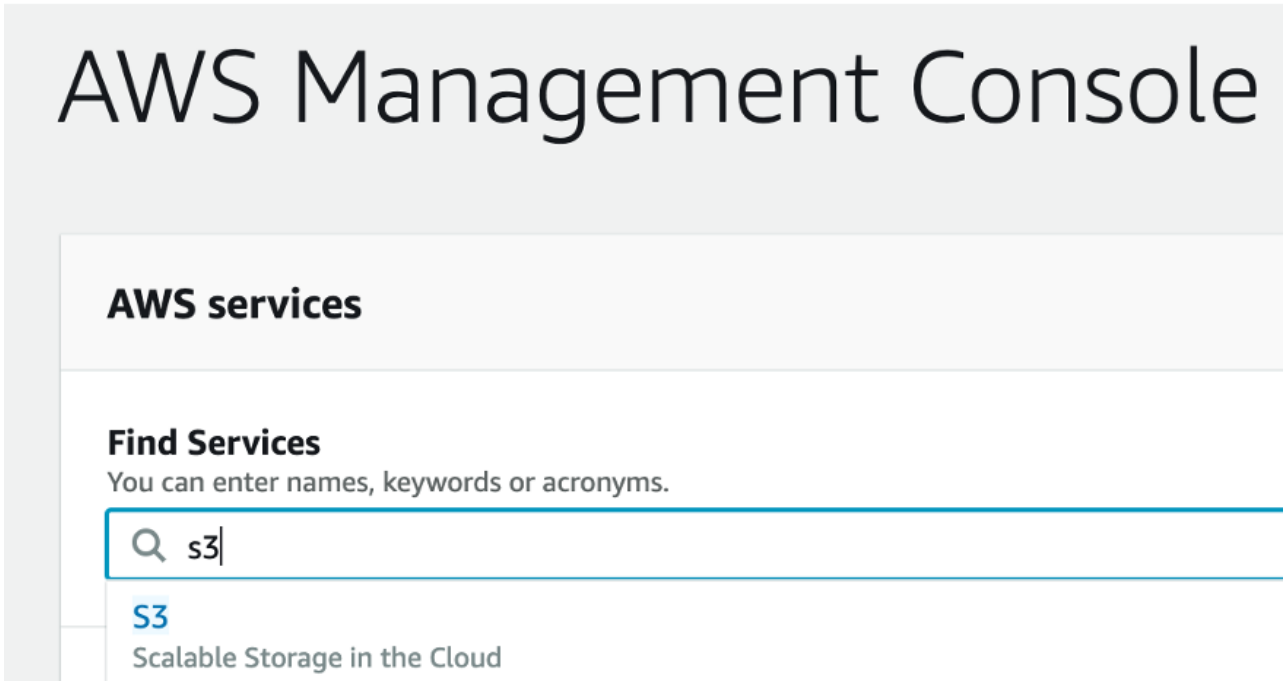
12. Amazon SageMaker XGBoost algorithm expects data to be in libSVM or CSV format (without header) and the first column must be the target variable. So in this step we transform the data accordingly

```
# cell 13
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)
```

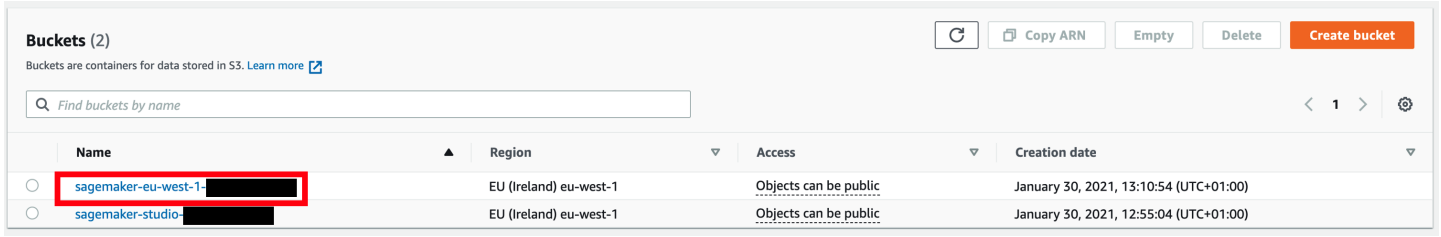
13. Now we will upload the final data into Amazon S3 bucket.

```
# cell 14
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')
```

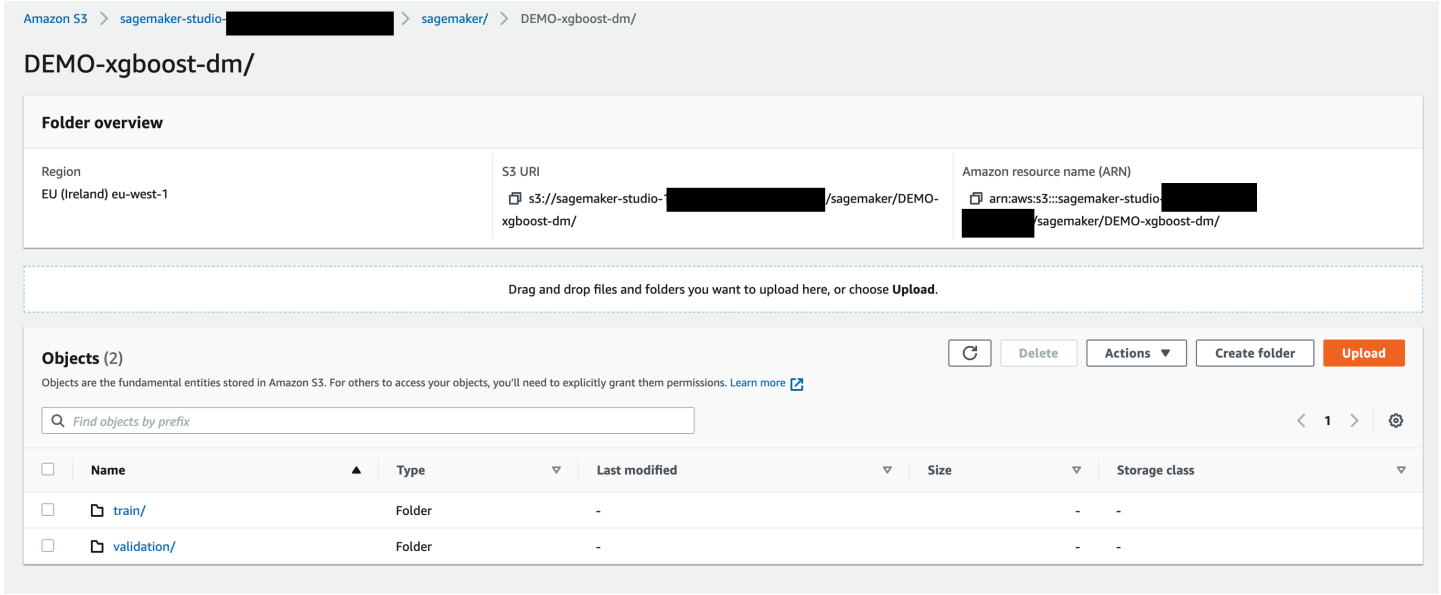
14. Now check the S3 bucket through the console to make sure you have uploaded the train.csv and validation.csv. In the AWS Console Main Page, type “S3”.



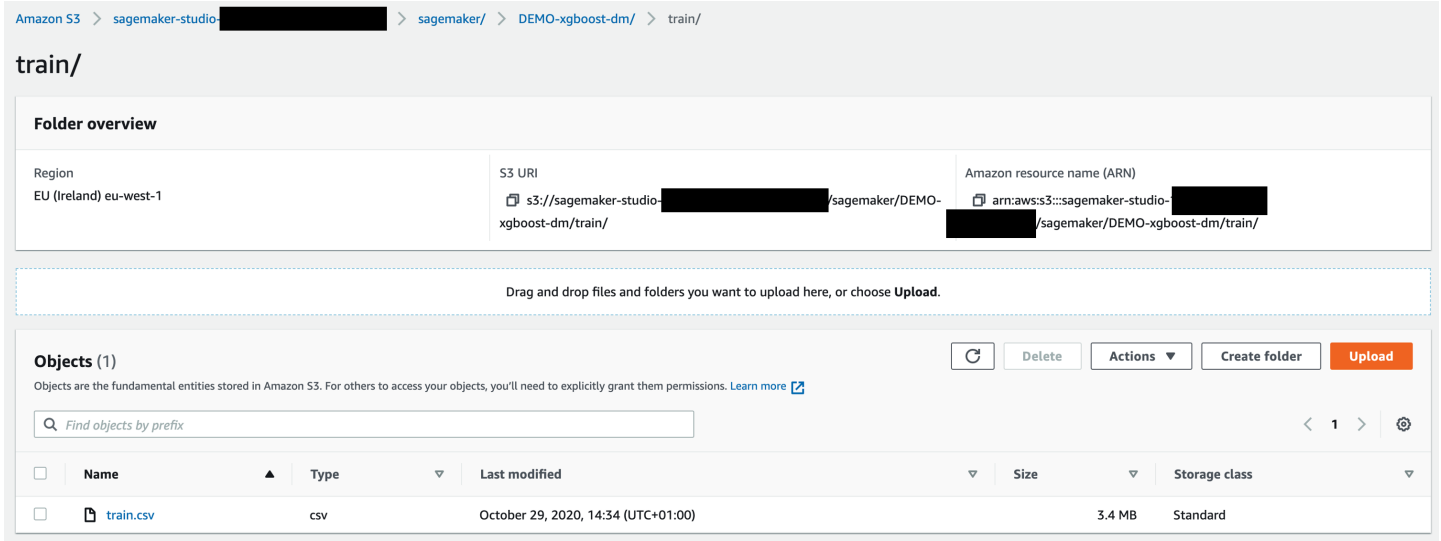
15. Click on your bucket name. If two buckets are present, take the one including your region name ("eu-west-1" as an example in the image below).

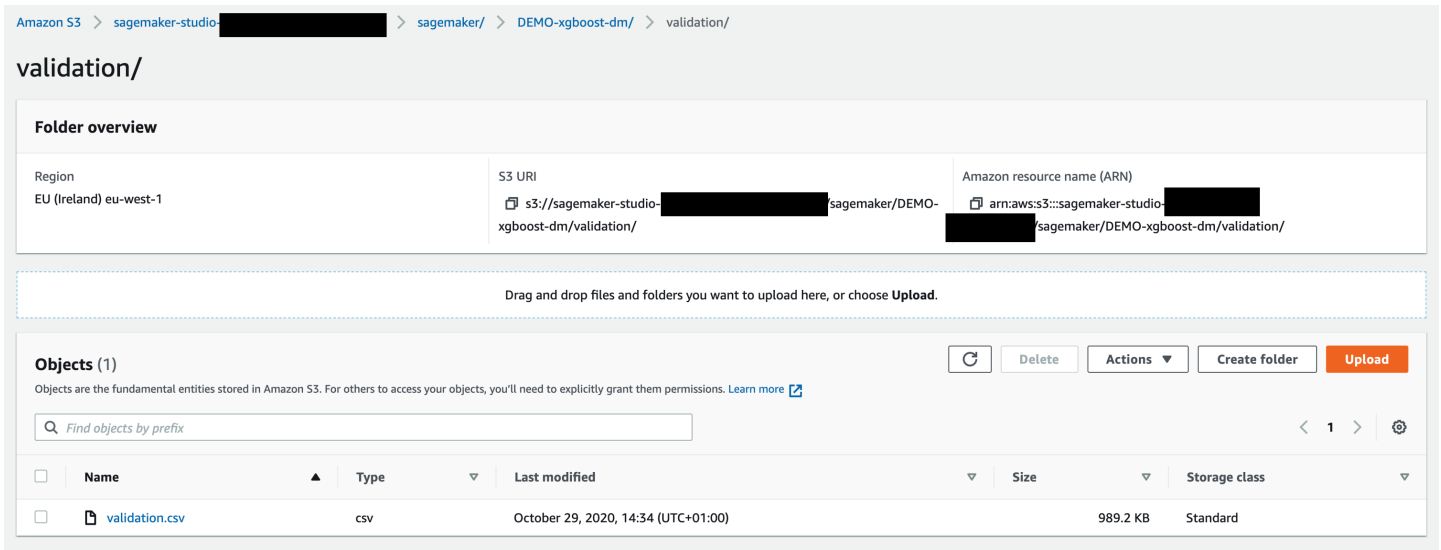


16. Click on "sagemaker/" and then on "DEMO-xgboost-dm/". You will see a "train/" and "validation/" folder.



17. You can have a look inside of each folder to make sure that the “train.csv” file is there as well as the “validation.csv” file.





Congratulations!! You have successfully prepared the data to train an XGBoost model.

Conclusion

In this lab you have walked through the process of required environment setup and data engineering to clean and prepare your data for model building and training. In the next lab you will learn how to Train , Tune and Deploy an XGBoost model using SageMaker's Built-in XGBoost algorithm.