

DSCI 525 - Web and Cloud Computing

Milestone 2: Your team is planning to migrate to the cloud. AWS gave 400(100 each) to your team to support this. As part of this initiative, your team needs to set up a server in the cloud, a collaborative environment for your team, and later move your data to the cloud. After that, your team can wrangle the data in preparation for machine learning.

Milestone 2 checklist

You will have mainly 2 tasks. Here is the checklist...

- Task 1: To set up a collaborative environment
 - Setup your EC2 instance with JupyterHub.
 - Install all necessary things needed in your UNIX server (Amazon EC2 instance).
 - Set up your S3 bucket.
 - Move the data that you wrangled in your last milestone to S3.
- Task 2: Wrangle the data in preparation for machine learning
 - Get the data from S3 in your notebook and make data ready for machine learning.


Everything in this milestone is to be completed in sequential order. But if you want to divide the tasks, you can ask a team member to work on Task 2 (4. Get the data that we wrangled in our first milestone. and 6. Wrangle the data in preparation for machine learning) locally on their laptop while the other members set up the infrastructure (EC2, S3) and TLJH in the cloud. This way, you can move quick.

Outside of Milestone: I strongly recommend you spin up your own instance and experiment with the s3 bucket in doing something (there are many things that we learned and practical work from video series) to get comfortable with AWS. But we won't be looking at it for a grading purpose.

Keep in mind:

- All services you use are in region `us-west-2` .
- Don't store anything in these servers or storage that represents your identity as a student (like your student ID number) .
- Use only default VPC and subnet.
- No IP addresses are visible when you provide the screenshot.
- You do proper budgeting so that you don't run out of credits.

- We want one single notebook for grading, and it's up to your discretion on how you do it. ***So only one person in your group needs to spin up instance and a t2.large is of decent size.***
- Please stop the instance when not in use. This can save you some bucks, but it's again up to you and how you budget your money. Maybe stop it if you or your team won't use it for the next 5 hours?
- Your AWS lab will shut down after 4 hours (also your services). When you start it again (after it is stopped), your AWS credentials (***access key, secret, and session token***) will change, and you want to update your credentials file with the new one.
- If you don't want your lab to shut down, click on the start lab again before 4 hours ends. This will fill up your time to 4 hours.
- Say something went wrong and you want to spin up another EC2 instance, then it will be good to terminate the previous one.
- We will be choosing the storage to be **Delete on Termination** (that is the default option), which means that stored data in your instance will be lost upon termination. Make sure you save any data to S3 and download the notebooks to your laptop so that next time you have your jupyterHub in a different instance, you can upload your notebook there.
- Wherever I ask for screenshots, you can provide the screenshot location in your GitHub.

NOTE: Everything you want for this notebook is discussed in lecture 3 and lecture 4. So you can follow the same order of things shown in lecture 4. You can also refer to the videos linked in each section under image  to see a demo on how to do it. In addition, I will put a link to the lecture note section that applies to each question in this milestone so that you don't get lost.

1. Setup your EC2 instance

- Name of the instance to be **mds-your_groupnumber** . For example if my group is 14 I would name it **mds-14** .
- AMI to be **Ubuntu server 22.04 LTS (HVM)** .
- Instance type to be **t2.large** .
- Architecture to be **64-bit(x86)** .
- Storage to be **30 GB** .
- Make sure you install TLJH in your instance, by giving instructions in **User Data** .

Check [this](#) section in lecture notes for more details on setting up EC2 instance with TLJH.

rubric={correctness:20}

Please provide here the GitHub path to your screenshot for grading.



2. Setup the server

rubric={correctness:20}

2.1) Add your team members to EC2 instance.

2.2) Setup a common data folder to download data, and this folder should be accessible by all users in the JupyterHub.

2.3) Install and configure AWS CLI.

Check following sections in lecture notes for more details;

- [Logging into EC2s](#)
- [Setting up a common space in EC2](#)
- [AWS CLI](#)

Please provide here the GitHub path to your screenshot for grading.

Make sure you mask the IP address refer [here](#).



3. Setup your JupyterHub

rubric={correctness:20}

Please provide here the GitHub path to your screenshot for grading.

I want to see all the group members here in this screenshot



Check [this](#) section in lecture notes for more details on setting up jupyterHub.

4. Get the data what we wrangled in our first milestone.

You have to install the packages that are needed. Refer this TLJH [document](#). Refer `pip` section.

Don't forget to add option -E. This way, all packages that you install will be available to other users in your JupyterHub. These packages you must install and install other packages needed for your wrangling.

```
sudo -E pip install pandas
sudo -E pip install pyarrow
sudo -E pip install s3fs
```

Check [this](#) section in lecture notes for more details on installing packages in TLJH.

As in the last milestone, we looked at transferring the data from Python to R, and we have different solutions. To stay consistent, I uploaded the parquet file (which I combined and converted in the last milestone), which we can use moving forward. Here in this section, I am getting that file using API; (the section is prepopulated for you)

Remember, from now on, we are going to run this notebook in the TLJH that is set up from previous steps. You can upload your notebook there using the upload button in jupyter. After that, you are going to run the following code to get the data downloaded to your EC2 instance and then you move to questions 5 and 6.

```
In [1]: import re
import os
import glob
import zipfile
import requests
from urllib.request import urlopen
import json
import pandas as pd
```

Remember here we gave the folder that we created in Step 2.2 as we made it available for all the users in a group.

```
In [ ]: # DO NOT RUN this block

# # Necessary metadata
# article_id = 14226968 # this is the unique identifier of the article on figshare
# url = f"https://api.figshare.com/v2/articles/{article_id}"
# headers = {"Content-Type": "application/json"}
# output_directory = "/srv/data/"
#
# response = requests.get(url, headers=headers)
# data = json.loads(response.text) # this contains all the articles data, but we only need the files
# files = data["files"] # this is just the data about the files, not the articles
#
# files_to_dl = ["combined_model_data_part1.parquet.zip"] ## Please download the files
# for file in files:
#     if file["name"] in files_to_dl:
#         os.makedirs(output_directory, exist_ok=True)
```

```
#         urlretrieve(file["download_url"], output_directory + file["name"])
#
# with zipfile.ZipFile(os.path.join(output_directory, "combined_model_data_p
#         f.extractall(output_directory)
```

5. Setup your S3 bucket and move data

rubric={correctness:15}

Check [this](#) section in lecture notes for more details on setting up S3 bucket and ***making it public***.

5.1) Create a bucket and name should be mds-s3-groupnumber-studentname. Replace groupnumber with your "group number", and studentname with your "student name". For example, if my group is 14 and my name is "Gittu George", then I would name it "mds-s3-14-gittu". (Note: For the purpose of this milestone only one student need to create this.)

5.2) Create your first folder called "output".

5.3) Move the "observed_daily_rainfall_SYD.csv" file from the Milestone1 data folder to your s3 bucket from your local computer.

5.4) Moving the parquet file we downloaded(combined_model_data_parti.parquet) in step 4 to S3 using the cli what we installed in step 2.3.

Check [this](#) section in lecture notes for more details on moving data to S3.

Please provide here the GitHub path to your screenshot for grading.

Make sure it has 3 objects and see red **public** symbol in your bucket.



6. Wrangle the data in preparation for machine learning

rubric={correctness:20}

IMPORTANT: Don't forget to deal with the credentials before you start this section. Check details in lecture notes [here](#)

Check [this](#) section in lecture notes for more details on how to read data from S3.

Our data currently covers all of NSW, but say that our client wants us to create a machine learning model to predict rainfall over Sydney only. There's a bit of wrangling that needs to be done for that:

1. We need to query our data for only the rows that contain information covering Sydney
2. We need to wrangle our data into a format suitable for training a machine learning model. That will require pivoting, resampling, grouping, etc.

To train an ML algorithm we need it to look like this:

	model-1_rainfall	model-2_rainfall	model-3_rainfall	...	observed_rainfall
0	0.12	0.43	0.35	...	0.31
1	1.22	0.91	1.68	...	1.34
2	0.68	0.29	0.41	...	0.57

6.1) Get the data from s3 (`combined_model_data_parti.parquet` and `observed_daily_rainfall_SYD.csv`)

6.2) First query for Sydney data and then drop the lat and lon columns (we don't need them).

```
syd_lat = -33.86
syd_lon = 151.21
```

Expected shape `(1150049, 2)`.

6.3) Save this processed file to s3 for later use:

Save as a csv file `ml_data_SYD.csv` to `s3://mds-s3-xxx/output/` expected shape `(46020, 26)` - This includes all the models as columns and also adding additional column `Observed` loaded from `observed_daily_rainfall_SYD.csv` from s3.

Set constant and read files

```
In [2]: os.environ["AWS_SHARED_CREDENTIALS_FILE"] = "/srv/keys/credentials"
```

```
In [3]: models_url = 's3://mds-s3-15-lennon/combined_model_data_parti.parquet'
observed_url = 's3://mds-s3-15-lennon/observed_daily_rainfall_SYD.csv'
output_url = 's3://mds-s3-15-lennon/output/ml_data_SYD.csv'
```

```
In [4]: df_models = pd.read_parquet(models_url)
df_observed = pd.read_csv(observed_url)
```

```
In [5]: df_models.head()
```

```
Out[5]:
```

	time	lat_min	lat_max	lon_min	lon_max	rain (mm/day)	model
0	1889-01-01 12:00:00	-36.25	-35.0	140.625	142.5	3.293256e-13	ACCESS-CM2
1	1889-01-02 12:00:00	-36.25	-35.0	140.625	142.5	0.000000e+00	ACCESS-CM2
2	1889-01-03 12:00:00	-36.25	-35.0	140.625	142.5	0.000000e+00	ACCESS-CM2
3	1889-01-04 12:00:00	-36.25	-35.0	140.625	142.5	0.000000e+00	ACCESS-CM2
4	1889-01-05 12:00:00	-36.25	-35.0	140.625	142.5	1.047658e-02	ACCESS-CM2

```
In [6]: df_models.shape
```

```
Out[6]: (62513863, 7)
```

Filter by coordinates and set index

```
In [7]: syd_lat = -33.86
syd_lon = 151.21

df_models_syd = df_models.query('lat_min <= @syd_lat & lat_max >= @syd_lat &
df_models_syd = df_models_syd.drop(['lat_min', 'lat_max', 'lon_min', 'lon_max'])

df_models_syd.head()
```

```
Out[7]:
```

	time	rain (mm/day)	model
552240	1889-01-01 12:00:00	0.040427	ACCESS-CM2
552241	1889-01-02 12:00:00	0.073777	ACCESS-CM2
552242	1889-01-03 12:00:00	0.232656	ACCESS-CM2
552243	1889-01-04 12:00:00	0.911319	ACCESS-CM2
552244	1889-01-05 12:00:00	0.698013	ACCESS-CM2

```
In [11]: df_models_syd = df_models_syd.set_index('time')
```

```
In [13]: df_models_syd.head()
```

```
Out[13]:
```

	rain (mm/day)	model
time		
1889-01-01 12:00:00	0.040427	ACCESS-CM2
1889-01-02 12:00:00	0.073777	ACCESS-CM2
1889-01-03 12:00:00	0.232656	ACCESS-CM2
1889-01-04 12:00:00	0.911319	ACCESS-CM2
1889-01-05 12:00:00	0.698013	ACCESS-CM2

Pivot by Model and Resample

```
In [22]: df_models_pivot = df_models_syd.pivot(columns='model', values='rain (mm/day)')
```

```
In [36]: df_models_daily = df_models_pivot.resample('1D').sum()
```

```
In [37]: df_models_daily.head()
```

```
Out[37]:
```

	model	ACCESS-CM2	ACCESS-ESM1-5	AWI-ESM-1-1-LR	BCC-CSM2-MR	BCC-ESM1	CMCC-CM2-HR4	CMCC-CM2-SR5	C
	time								
	1889-01-01	0.040427	1.814552	35.579336	4.268112e+00	1.107466e-03	11.410537	3.322009e-08	2.6
	1889-01-02	0.073777	0.303965	4.596520	1.190141e+00	1.015323e-04	4.014984	1.312700e+00	0.9
	1889-01-03	0.232656	0.019976	5.927467	1.003845e-09	1.760345e-05	9.660565	9.103720e+00	0.4
	1889-01-04	0.911319	13.623777	8.029624	8.225225e-02	1.808932e-01	3.951528	1.317160e+01	0.3
	1889-01-05	0.698013	0.021048	2.132686	2.496841e+00	4.708019e-09	2.766362	1.822940e+01	0.3

5 rows × 25 columns

```
In [35]: df_models_daily.shape
```

```
Out[35]: (46020, 25)
```

Set time index for observed data

```
In [18]: df_observed['time'] = pd.to_datetime(df_observed['time'])
df_observed = df_observed.set_index('time')
df_observed.head()
```


Out[18]:

rain (mm/day)	
time	
1889-01-01	0.006612
1889-01-02	0.090422
1889-01-03	1.401452
1889-01-04	14.869798
1889-01-05	0.467628

In [34]: `df_observed.shape`

Out[34]: (46020, 1)

Merge models and observed together

In [38]: `df_merged = pd.merge(df_models_daily, df_observed, left_index=True, right_in`

In [42]: `df_final = df_merged.rename(columns={'rain (mm/day)' : 'observed'})`

In [48]: `df_final.head()`

Out[48]:

	ACCESS-CM2	ACCESS-ESM1-5	AWI-ESM-1-1-LR	BCC-CSM2-MR	BCC-ESM1	CMCC-CM2-HR4	CMCC-CM2-SR5	CI E
--	------------	---------------	----------------	-------------	----------	--------------	--------------	------

time								
1889-01-01	0.040427	1.814552	35.579336	4.268112e+00	1.107466e-03	11.410537	3.322009e-08	2.66
1889-01-02	0.073777	0.303965	4.596520	1.190141e+00	1.015323e-04	4.014984	1.312700e+00	0.94
1889-01-03	0.232656	0.019976	5.927467	1.003845e-09	1.760345e-05	9.660565	9.103720e+00	0.43
1889-01-04	0.911319	13.623777	8.029624	8.225225e-02	1.808932e-01	3.951528	1.317160e+01	0.36
1889-01-05	0.698013	0.021048	2.132686	2.496841e+00	4.708019e-09	2.766362	1.822940e+01	0.33

5 rows × 26 columns

In [50]: `df_final.shape`

Out[50]: (46020, 26)

Write result to S3

```
In [52]: df_final.to_csv(output_url)

In [53]: validate_df = pd.read_csv(output_url, index_col='time', parse_dates=['time'])

In [54]: validate_df.head()
```

Out[54]:

	ACCESS- CM2	ACCESS- ESM1-5	AWI-ESM- 1-1-LR	BCC-CSM2- MR	BCC- ESM1	CMCC- CM2-HR4	CMCC-CM2- SR5	CI E
time								
1889-01-01	0.040427	1.814552	35.579336	4.268112e+00	1.107466e-03	11.410537	3.322009e-08	2.66
1889-01-02	0.073777	0.303965	4.596520	1.190141e+00	1.015323e-04	4.014984	1.312700e+00	0.94
1889-01-03	0.232656	0.019976	5.927467	1.003845e-09	1.760345e-05	9.660565	9.103720e+00	0.43
1889-01-04	0.911319	13.623777	8.029624	8.225225e-02	1.808932e-01	3.951528	1.317160e+01	0.36
1889-01-05	0.698013	0.021048	2.132686	2.496841e+00	4.708019e-09	2.766362	1.822940e+01	0.33

5 rows × 26 columns