SDM
Federated Learning
(Lab Assignment)

Ricardo Silva Peres

NOVA University of Lisbon

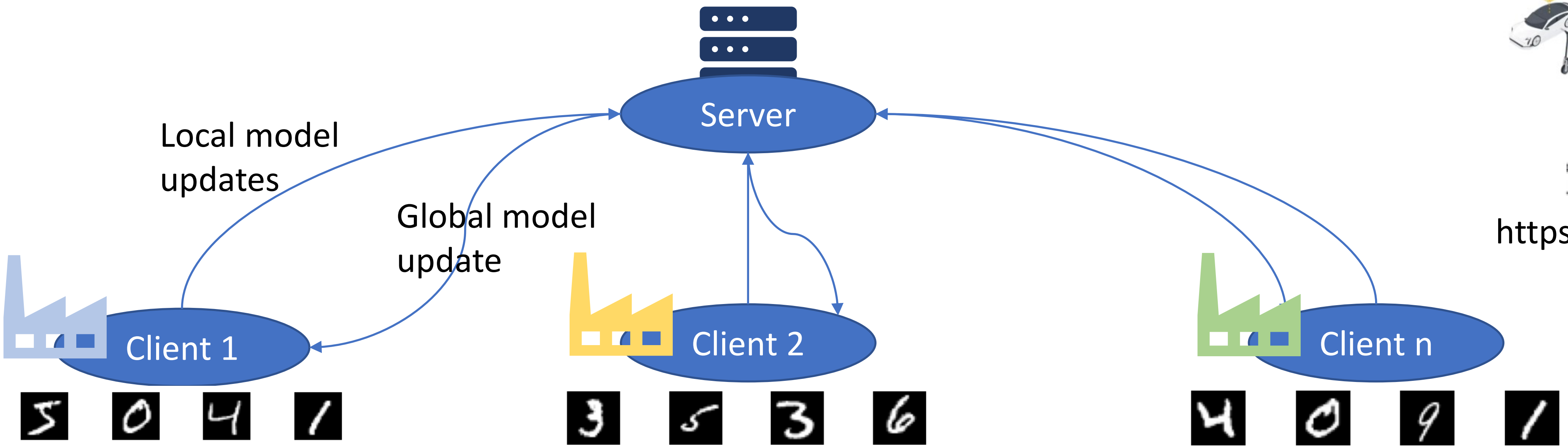
School of Science and Technology

ricardo.peres@uninova.pt

Deadline: 16th of June 23:59 (via Moodle)

Objective

Develop a simple MNIST **image classifier** based on a **Federated Learning** approach (using the *Flower Framework* and *Tensorflow* for Python).

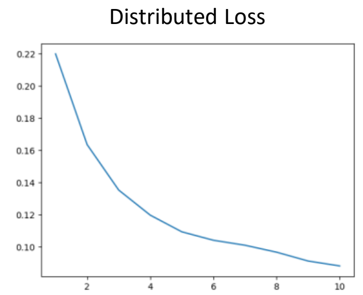


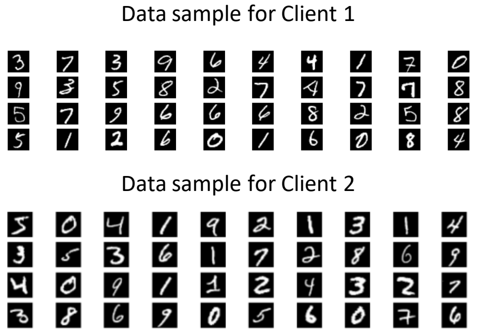
Data remains locally at the client

https://flower.dev/

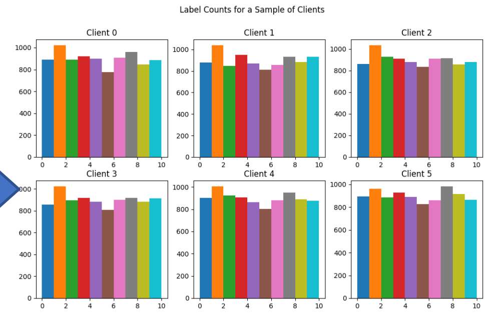
Evaluation Criteria

- Correct implementation of the server node [4]
- Correct implementation of the client nodes [6]
- Successful training of the image classifier using Federated Learning for 5 rounds [4]
- Free Choice Features (must implement at least 2) [6] :
  - Visualize the evolution of the distributed loss metric during the training procedure;
  - Explore and implement a different aggregation strategy for the server (other than FedAvg)
  - Visualize the different partial datasets from the local clients using Matplotlib or an equivalent library.
  - Visualize the label counts for a given sample of clients


Distributed Loss

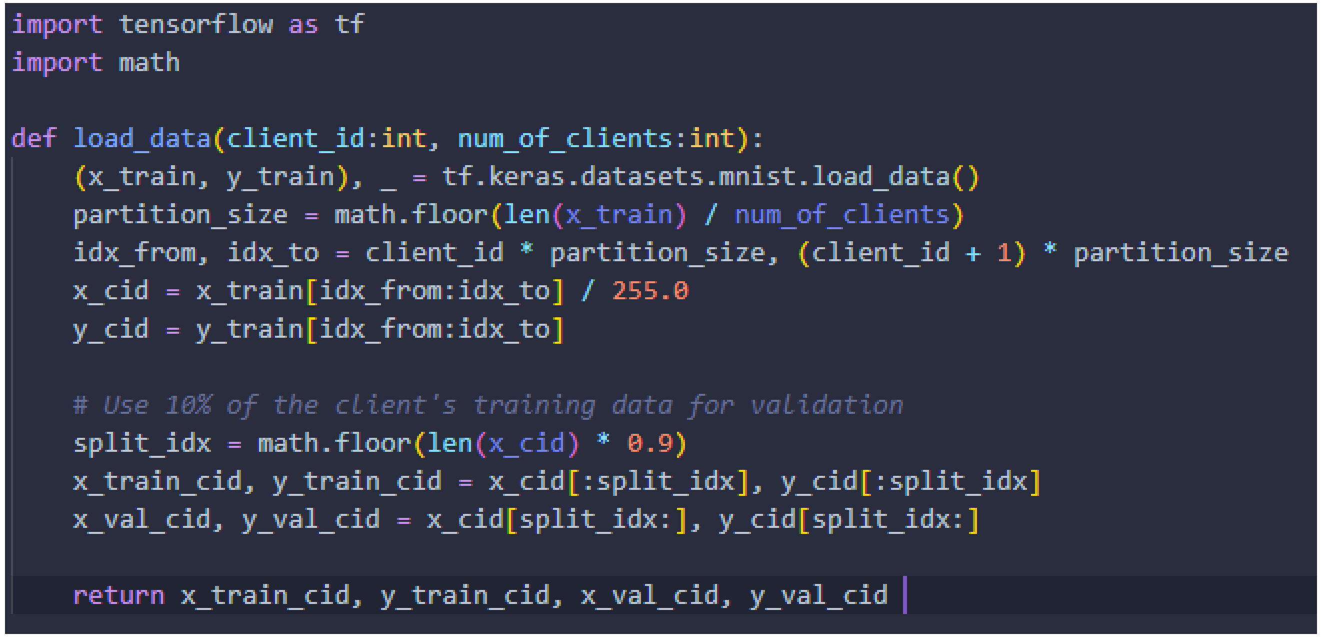

Data sample for Client 1

Data sample for Client 2


Label Counts for a Sample of Clients

Implementation Guidelines (1)

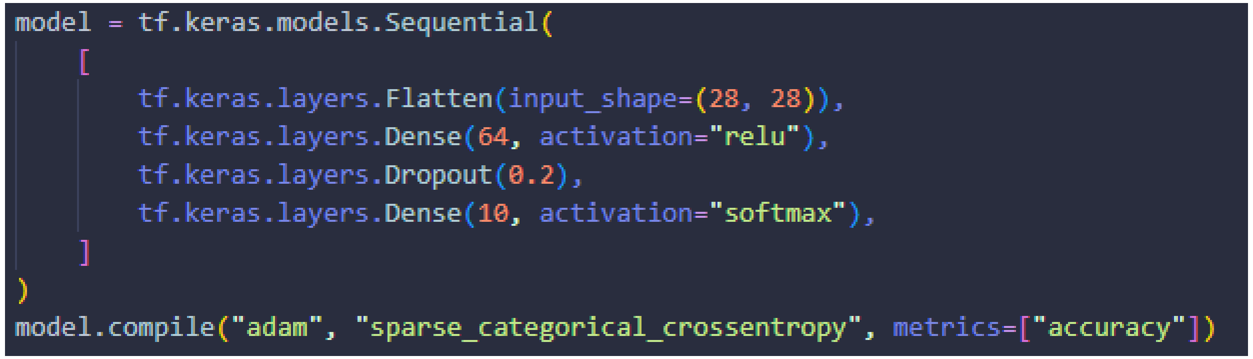
- The *load_data* function (provided in CLIP) facilitates the generation of a federated version of the MNIST dataset for a particular user, partitioned based on the total number of clients.



```python
import tensorflow as tf
import math

def load_data(client_id:int, num_of_clients:int):
    (x_train, y_train), _ = tf.keras.datasets.mnist.load_data()
    partition_size = math.floor(len(x_train) / num_of_clients)
    idx_from, idx_to = client_id * partition_size, (client_id + 1) * partition_size
    x_cid = x_train[idx_from:idx_to] / 255.0
    y_cid = y_train[idx_from:idx_to]

    # Use 10% of the client's training data for validation
    split_idx = math.floor(len(x_cid) * 0.9)
    x_train_cid, y_train_cid = x_cid[:split_idx], y_cid[:split_idx]
    x_val_cid, y_val_cid = x_cid[split_idx:], y_cid[split_idx:]

    return x_train_cid, y_train_cid, x_val_cid, y_val_cid
```

Implementation Guidelines (2)

- For the implementation of the client and server scripts, the official flower documentation should be followed: https://flower.dev/docs/quickstart-tensorflow.html
- For the model, experiment with the following architecture:

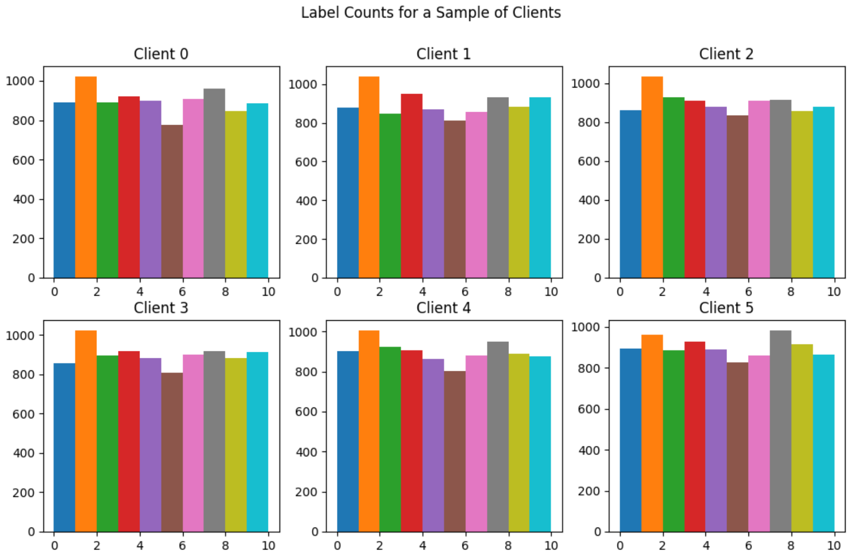


```
model = tf.keras.models.Sequential(
    [
        tf.keras.layers.Flatten(input_shape=(28, 28)),
        tf.keras.layers.Dense(64, activation="relu"),
        tf.keras.layers.Dropout(0.2),
        tf.keras.layers.Dense(10, activation="softmax"),
    ]
)
model.compile("adam", "sparse_categorical_crossentropy", metrics=["accuracy"])
```

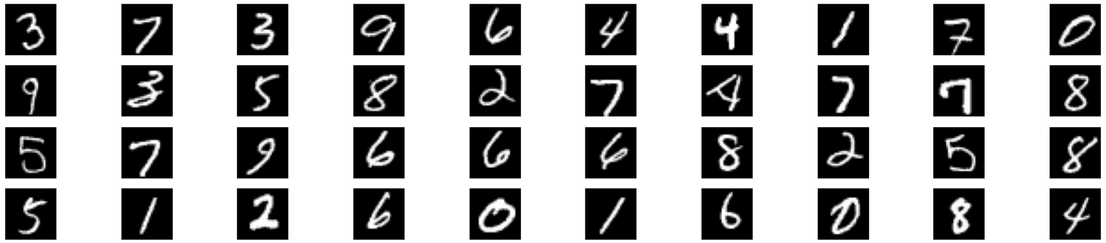
Implementation Guidelines (3)

- For the free choice features, the link below contains some examples regarding the different visualization types. Please note that the base dataset format/object is different, so **it must be adapted**. Nevertheless, the visualization part with Matplotlib is similar.

https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification

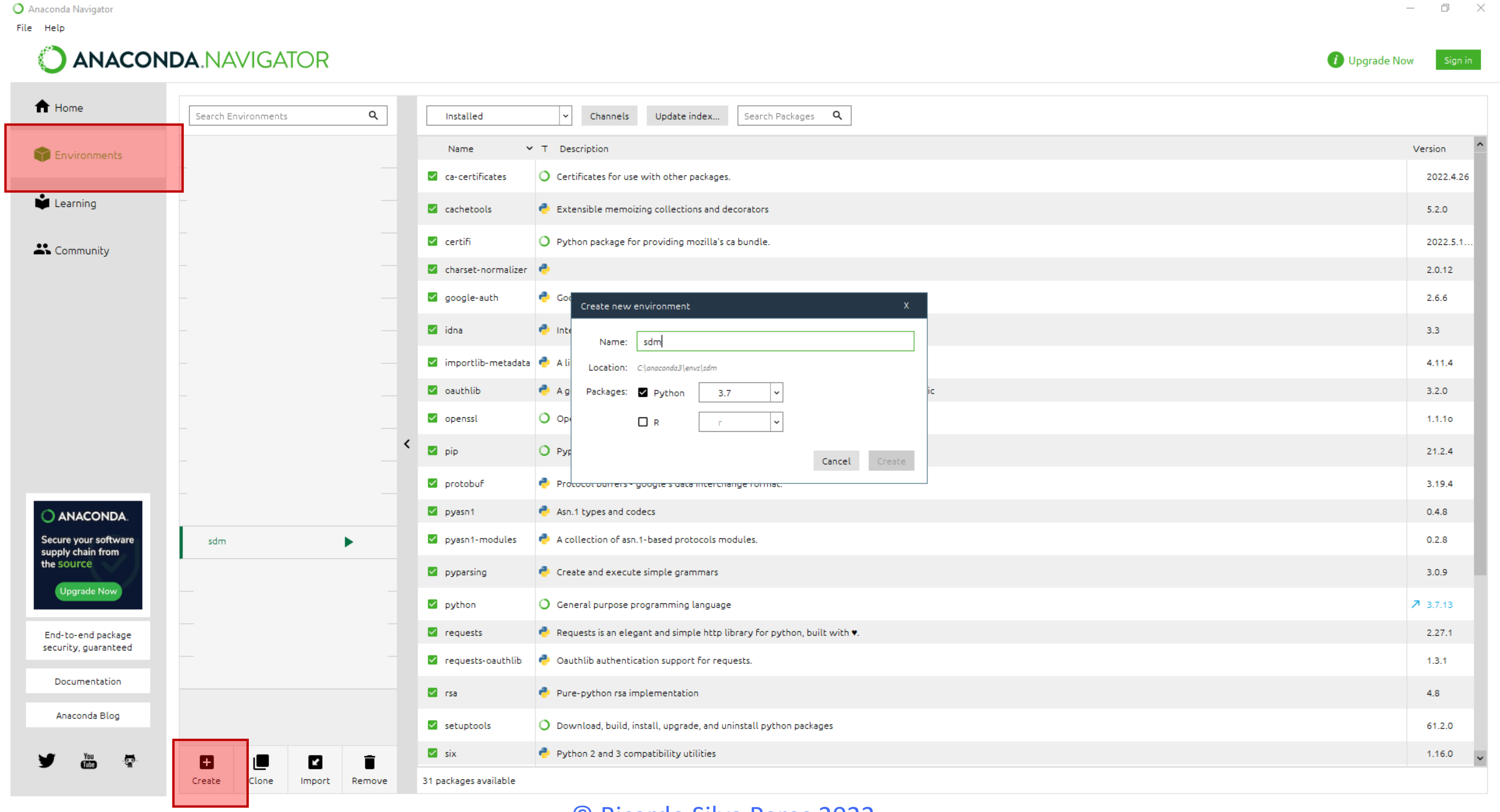


Data sample for Client 1



Setting up the development environment (1)

- If you don't have a python environment already configured, we suggest the usage of the Anaconda Distribution: https://www.anaconda.com/. Start by creating a new environment for SDM:



Create new environment

Name: sdm

Location: C:\anaconda3\envs\sdm

Packages: ☑ Python 3.7

□ R r

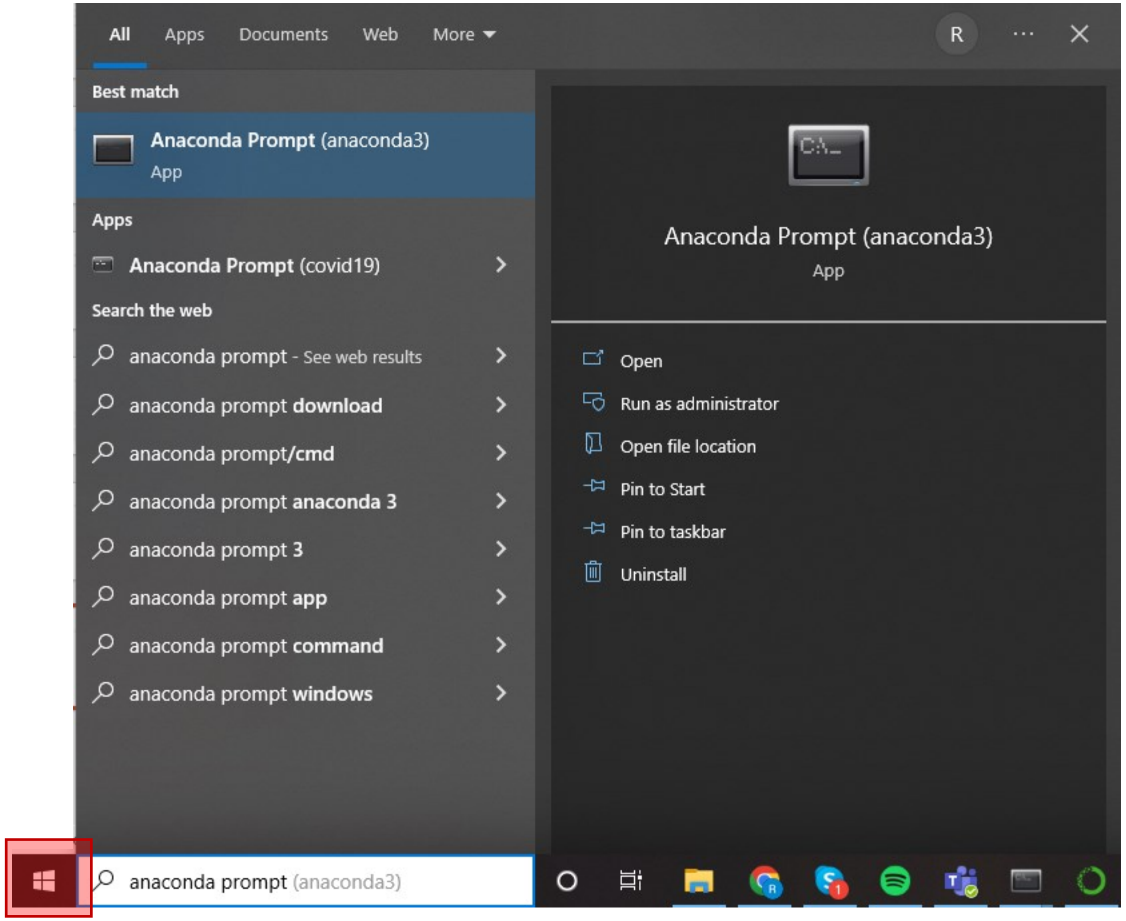
Cancel Create

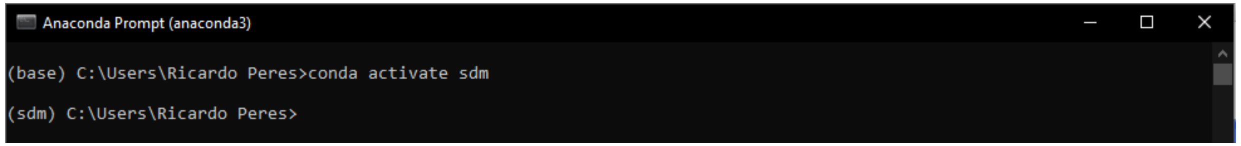
	Name	T	Description	Version
☑	ca-certificates		Certificates for use with other packages.	2022.4.26
☑	cachetools		Extensible memoizing collections and decorators	5.2.0
☑	certifi		Python package for providing mozilla's ca bundle.	2022.5.1...
☑	charset-normalizer			2.0.12
☑	google-auth		Goo	2.6.6
☑	idna		Inte	3.3
☑	importlib-metadata		A li	4.11.4
☑	oauthlib		A g ic	3.2.0
☑	openssl		Op	1.1.1o
☑	pip		Pyp	21.2.4
☑	protobuf		Protocol buffers - google's data interchange format.	3.19.4
☑	pyasn1		Asn.1 types and codecs	0.4.8
☑	pyasn1-modules		A collection of asn.1-based protocols modules.	0.2.8
☑	pyparsing		Create and execute simple grammars	3.0.9
☑	python		General purpose programming language	↗ 3.7.13
☑	requests		Requests is an elegant and simple http library for python, built with ♥.	2.27.1
☑	requests-oauthlib		Oauthlib authentication support for requests.	1.3.1
☑	rsa		Pure-python rsa implementation	4.8
☑	setuptools		Download, build, install, upgrade, and uninstall python packages	61.2.0
☑	six		Python 2 and 3 compatibility utilities	1.16.0

31 packages available

Setting up the development environment (2)

- Open the Anaconda Prompt and run the command ***conda activate sdm***

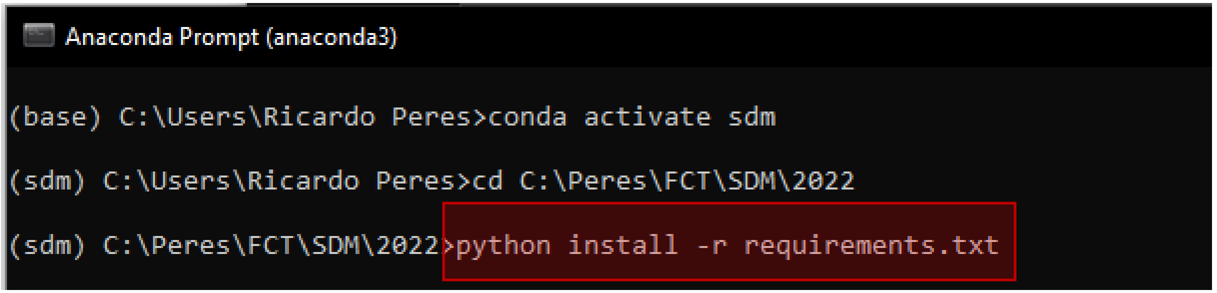




```
(base) C:\Users\Ricardo Peres>conda activate sdm

(sdm) C:\Users\Ricardo Peres>
```

Notice how the active environment changes from *base* to the newly created *sdm* env.

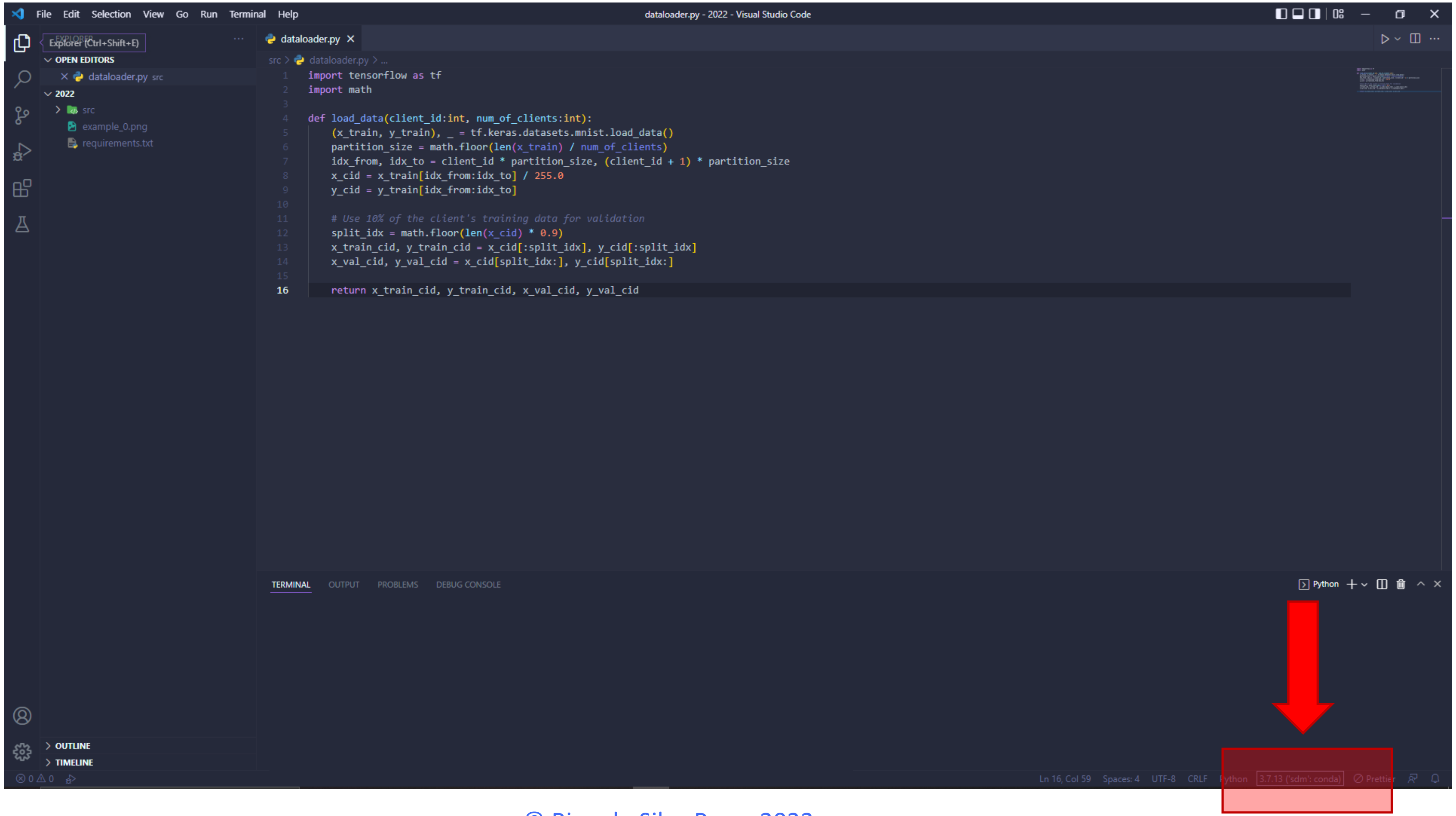
From here, navigate to your project directory (using the *cd* command) and run the command below to install the dependencies listed in the *requirements.txt* file (provided in CLIP).



```
(base) C:\Users\Ricardo Peres>conda activate sdm

(sdm) C:\Users\Ricardo Peres>cd C:\Peres\FCT\SDM\2022

(sdm) C:\Peres\FCT\SDM\2022>python install -r requirements.txt
```

Setting up the development environment (3)

- The suggested IDE is **VS Code**. The SDM env can be selected in the bottom right corner, or by pressing CTRL + Shift + P and searching for “Select Interpreter”



```python
import tensorflow as tf
import math

def load_data(client_id:int, num_of_clients:int):
    (x_train, y_train), _ = tf.keras.datasets.mnist.load_data()
    partition_size = math.floor(len(x_train) / num_of_clients)
    idx_from, idx_to = client_id * partition_size, (client_id + 1) * partition_size
    x_cid = x_train[idx_from:idx_to] / 255.0
    y_cid = y_train[idx_from:idx_to]

    # Use 10% of the client's training data for validation
    split_idx = math.floor(len(x_cid) * 0.9)
    x_train_cid, y_train_cid = x_cid[:split_idx], y_cid[:split_idx]
    x_val_cid, y_val_cid = x_cid[split_idx:], y_cid[split_idx:]

    return x_train_cid, y_train_cid, x_val_cid, y_val_cid
```

Ln 16, Col 59 Spaces: 4 UTF-8 CRLF Python 3.7.13 ('sdm': conda) Prettier