**Setting environment with Pycharm and Conda**

Ctrl-alt-s

Project – interpreter – select conda

Now we must install the dependencies or libraries

Tools -> Sync Python Requirements

**Way to install modules without pycharm**

pip list this command list all modules installed in the actual environment

pip freeze > requirements.txt this command get the list of modules in the actual environment and build a txt file

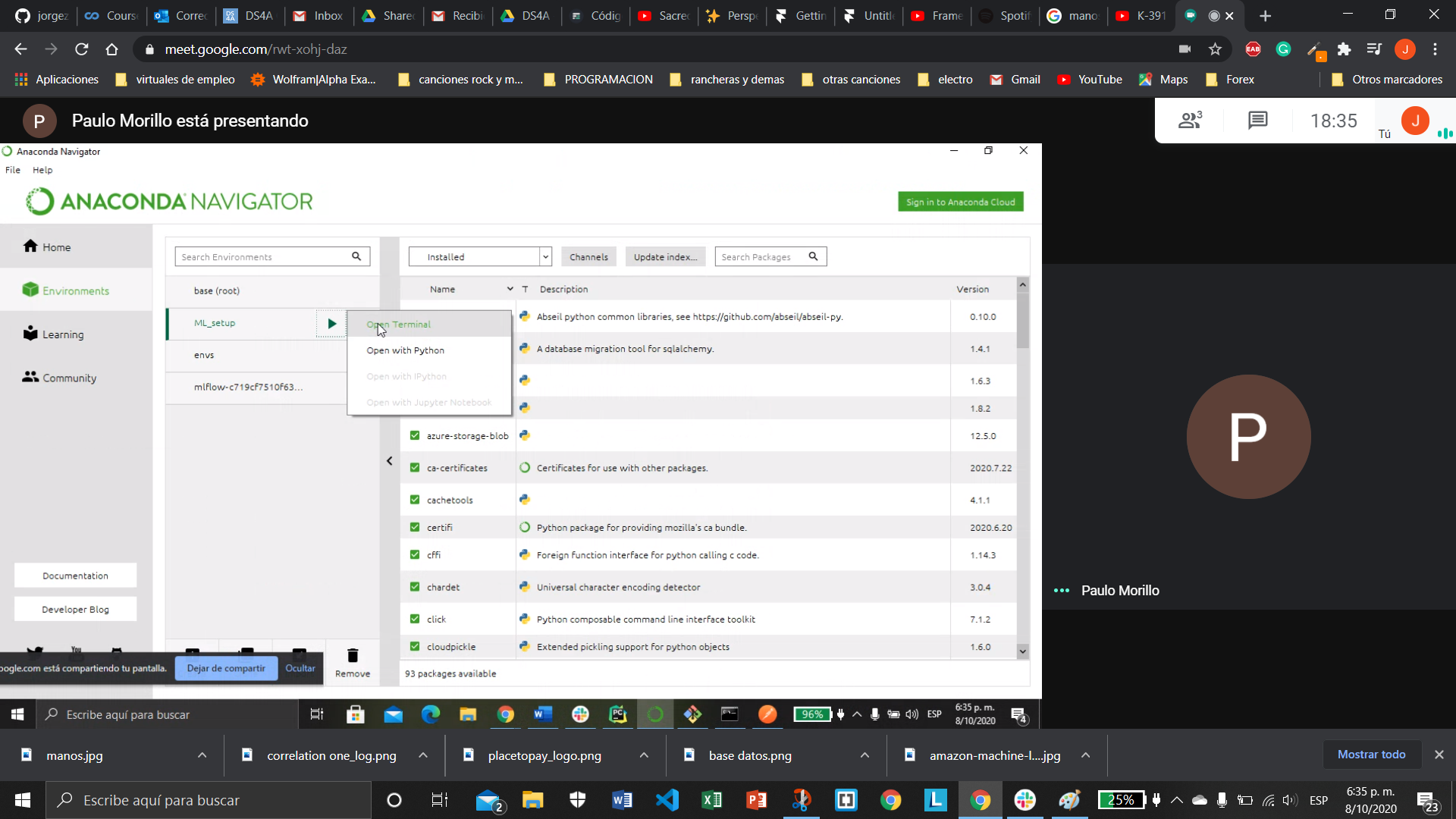
pip install -r requirements.txt way to install the modules in new environment

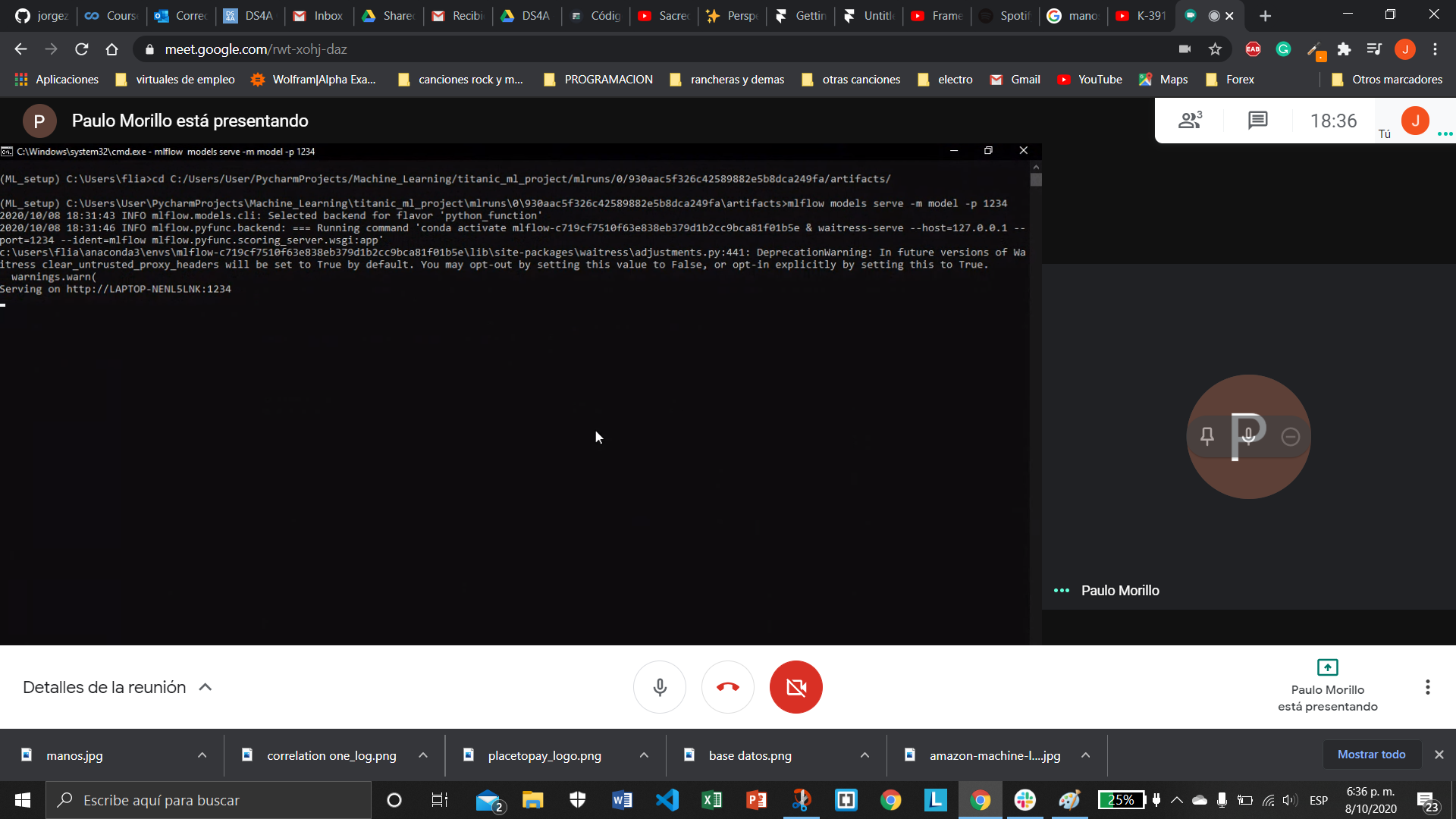
Possible mistakes:

Anaconda doesn’t run pycharm as administrador and if we are using OS module we are going to have troubles. To solve this, close pycharm and reopened as admin

Scikit-learn module couldn’t be installed directly with pip install, This is so weird but I remove the version and pip install the same version of the requirements file. (No reason about it)

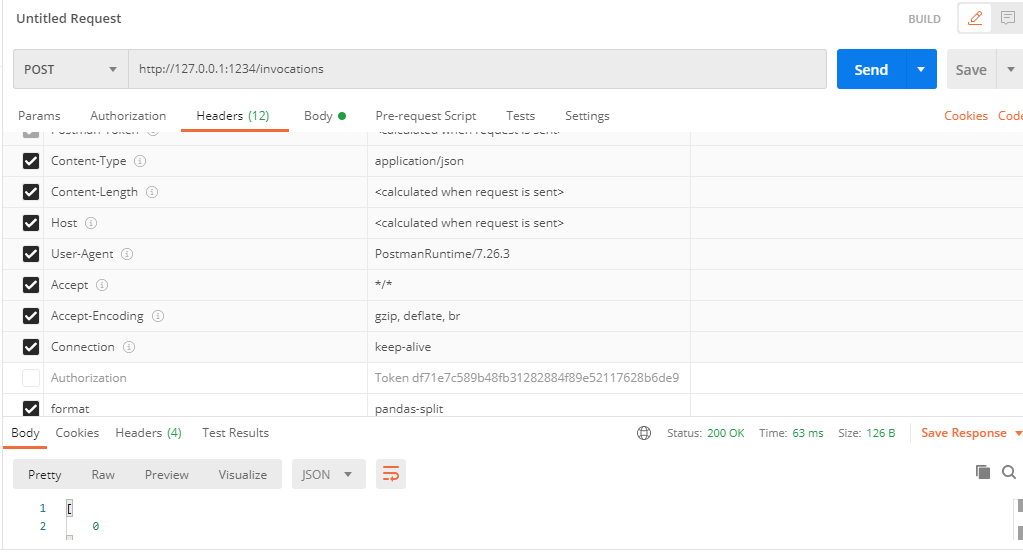
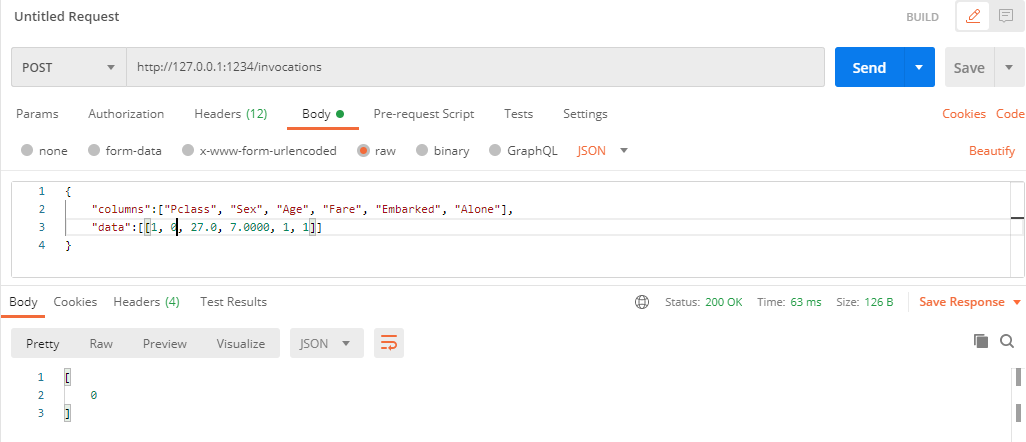
To serve mlflow for me was better opening the terminal with conda. And trying to reach the model folder with cd.





Pclass Sex Age Fare Embarked Survived Alone

curl -X POST -H "Content-Type:application/json; format=pandas-split" --data '{"columns":["Pclass", "Sex", "Age", "Fare", "Embarked", "Alone"],"data":[[3, 0, 27.0, 7.0000, 1, 1]]}' http://127.0.0.1:1234/invocations



Yaml is a superset of JSON

With pyyaml module

We can import functions, classes, dictionaries, list, variables, all that we import is going to be an object (for this reason we can have all data types). However, we should be load data with safe\_load to avoid security troubles.

Like a little overview for a yaml file the next example has a dictionary with item hola and a list as value

hola:

- mundo  
 - yaml

When we import that file, we are going to have {“hola”: [“mundo”, “yaml”]}

Installing typer

pip install typer[all]

Typer is a library which permits to get data from CLI. Instead sys.arg they only need to declare the arguments in the main function (as we did it with C)

For example this is the hello world:

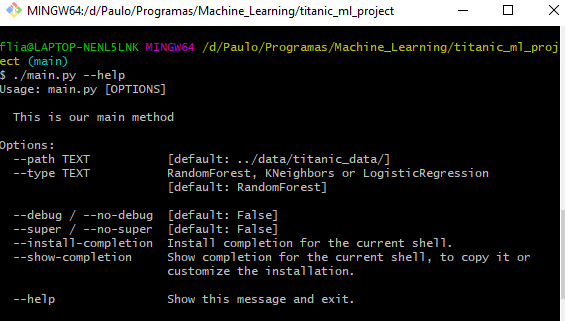
def main(name: str, lastname: str, formal: bool = False):  
 *""" This is our main method """* print(name, lastname, formal)  
  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 typer.run(main)

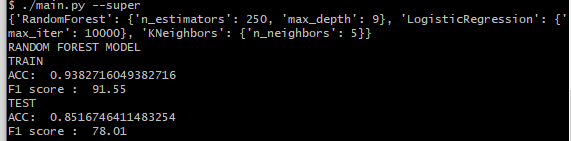
$ ./main.py Paulo Morillo --formal

Paulo Morillo True

As you can see, we have 3 arguments name as a str, lastname as str too, and formal as a Boolean with a default value equal to False.

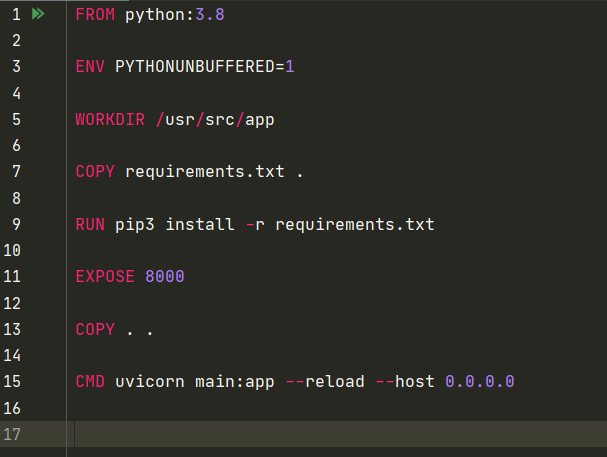
To run this program





How to create a Dockerfile

We have to create a dockerfile to use docker-compose in our server, so that we can host our program keeping the different versions and modules that we need to run our program.



From: here we are going to use the version of python that we want in our image

Env: here we can put all the environment variables, and pythonunbuffered is one of them

Workidr: is where the app is going to run, also we are telling that all the copy is going to go in here

Run: to run a command inside the filesystem image

Expose: listen port

Cmd: run a command inside the container

How to create a Container



Version: version of docker to use

Services: services that our container is gonna have

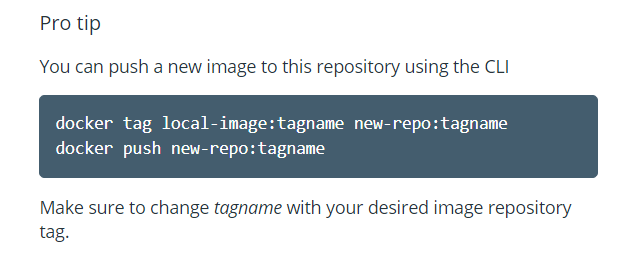
* Web: this is the name of our service
* Build: we said . because this automatically is going to build the image in the actual directory
* Ports: ports where the container is gonna be opened
* Volumes: this says that everything we are going to change in our project this changes are gonna be reproduced in the /usr/src/app

To create a container we need to run “docker-compose up” and this command will create the Dockerfile which is in the same directory

Once we have the container run we can run it from the desktop app of docker and then use this url ‘http://localhost:8001/docs’ on there you will can interact with the api

Dockerhub

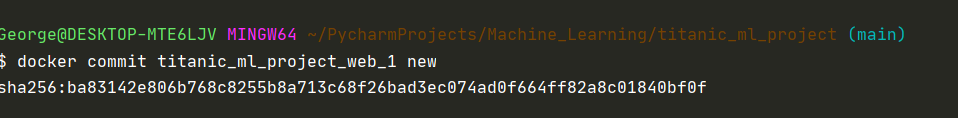
First of all we have to have a count on dockerhub, then create a new repository, after that you can use the **hole** name of your repo as new-repo



The push can take some time

To save the changes we have to put:

* docker commit image\_changed new\_name



After this again to the tag and the push