ML assessment

**1. Description**

This writeup documents my code and approaches to the two exercises, I chose question 1 (computer vision) and question 5 (server side authentication) as I was more familiar with the topics. Both questions are developed using python.

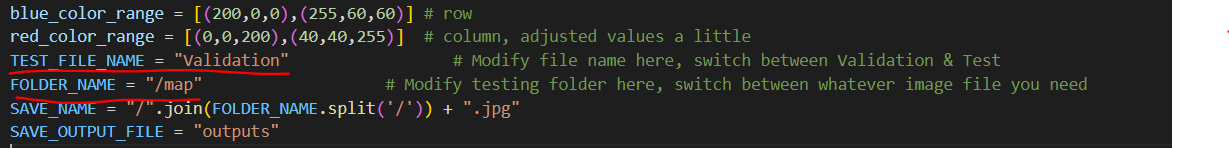
**2. Question 1 (Computer Vision)**

This was a real fun one 😊

For this question I used the **OpenCV** library to identify the blue and red dots. The algorithm I used was as follows:

1. For each image, count the number of blue and red dots (in that order) and store it as an entry in a dictionary in this format *Filename: [# of bluedots, # of red dots]*. This is done by applying a mask on the color ranges, and then counting the contours CORRESPONDING TO THAT MASK.
2. The dictionary is then sorted by its values and then saved in another dictionary
3. OpenCV has a built in concatenation function (cv2.vconcat) that I can use to combine the images together, it only took in the arranged images in sorted order as an argument. The max width of the image was determined by calculating the MAX value of the number of red dots in any entry, which corresponds to the last column of the image
4. We generate and then save the image to an outputs folder

To test it out, all you have to do is change the code in this portion



The folder structure should be something like this:

Root/

├── question-1.py

├── Test/

│ ├── fumo/

│ │ ├── picture1.jpg

│ │ ├── picture2.jpg

│ │ ├── …

│ ├── galaxy/

│ │ ├── picture1.jpg

│ │ ├── picture2.jpg

│ │ ├── …

│ └── simple/…

├── Validation/

│ ├── map/

│ │ ├── picture1.jpg

│ │ ├── picture2.jpg

│ │ ├── …

│ ├── saturn/

│ │ ├── picture1.jpg

│ │ ├── picture2.jpg

│ │ ├── …

│ └── xray/…

└── output/

│ ├── fumo.jpg

│ ├── galaxy.jpg

└── └── …

With the outputs stored in output folder, you can view some of the results there!

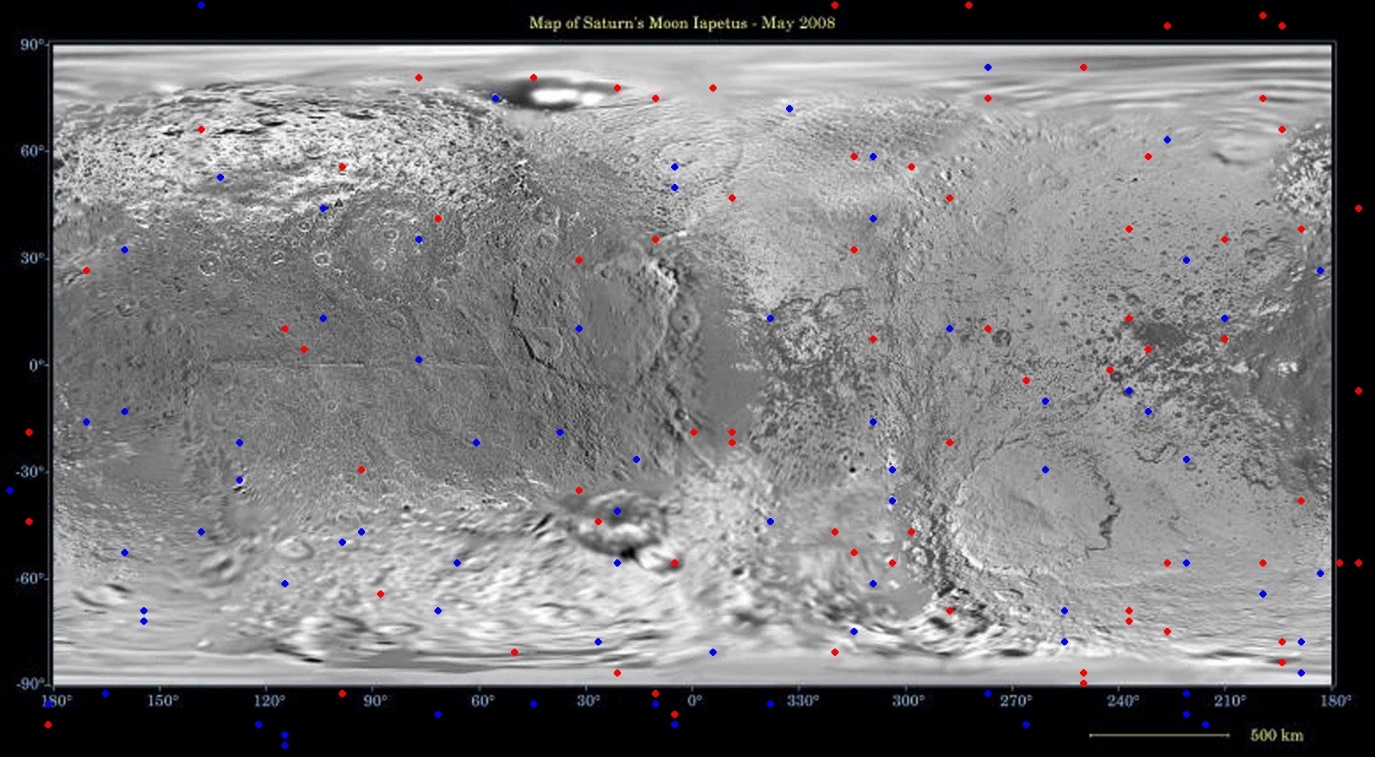


Fumo

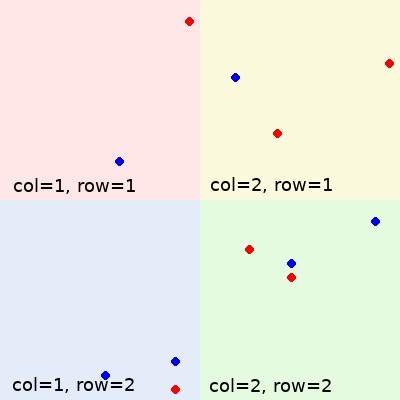


galaxy

map



Saturn



Simple



Xray

**3. Question 5 (server side authentication)**

This question was also done in python, while I have some experience in golang for setting up servers, I think having python demonstrate a simple authentication request would be best.

I used the python *http.server* library to handle the requests and POSTMAN to generate the requests.

I configured authentication to use a POST request, where the user will parse in the login, password and location in the request body in JSON format. Understandably the user would usually send location in the header of the request, but for this exercise simplicity I opted to put it into the request body.

Here is a sample request body: {"login":"sgriley","password":"chin","location":"singapore"}

How it works is as follows:

1. We set up the server by running python question-5.py, this defaults the server to run on localhost at port **8080**.
2. We can then start to send our requests via postman with this configuration:

Graphical user interface, application

Description automatically generated

1. There is a login.txt and a passwords.txt, which will store all our logins and password in text format. The handler will check against this list when verifying authentication
2. We first check the country code by checking with the ISO3166 guidelines on the actual location of the country to its corresponding country code, I converted the code to lowercase since usernames shouldn’t caps-specific.
3. When a user enters a wrong country code, we return a **400 forbidden request**. The user should enter a correct country code
4. When a user enters a login ID that does not exist in the database, we return a **404 not found error**
5. To hide the password, I used the hashlib library on python, which just hashes a password with an additional salt added to it for extra protection. The password is then also checked against the database for an existing record and will also throw a **404 not found error** otherwise.

Here are some sample test cases

1. {"login":"sgriley","password":"chin","location":"singapore"}

Successful Login!

A screenshot of a computer

Description automatically generated

1. {"login":"riley","password":"chin","location":"singapore"}

bad request, country code error

Graphical user interface, text, application, email

Description automatically generated

1. {"login":"sgdamien","password":"chin","location":"singapore"}

Not found, user doesnt exist

Graphical user interface, text, application, email

Description automatically generated

1. {"login":"sgriley","password":"ching","location":"singapore"}

Not found, user doesnt exists

Graphical user interface, text, application, email

Description automatically generated