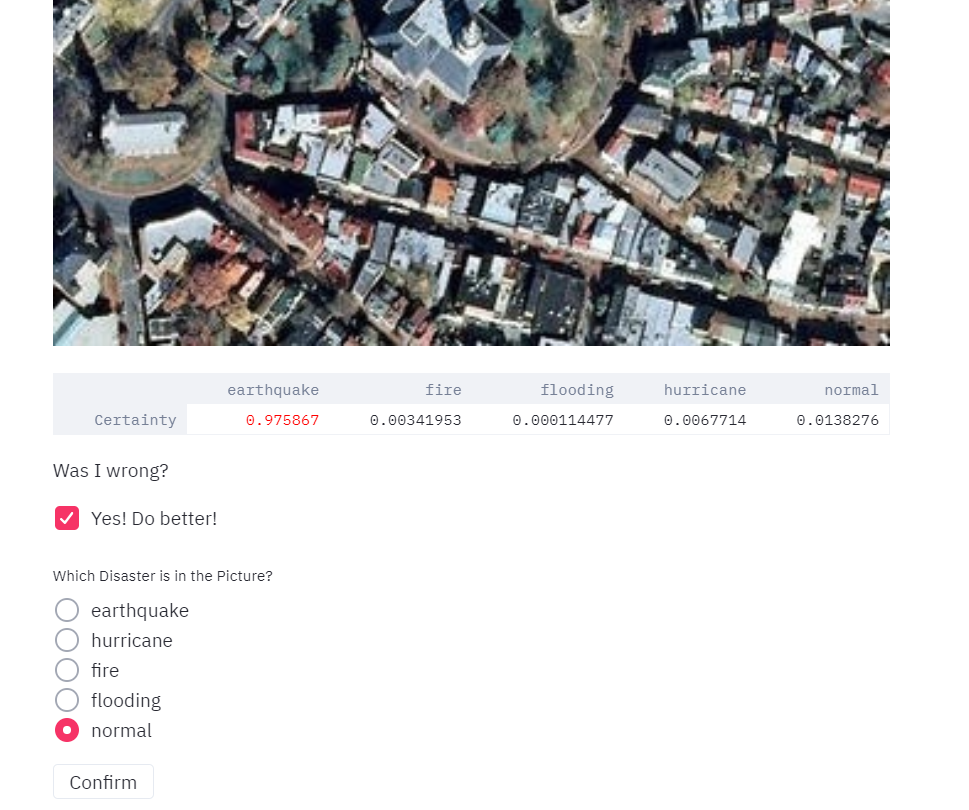
**Friday 12/6/2019**

Today, I had a conversation with Mr. White. I first thanked him for working out making our Streamlit website compatible with Director. Then we talked about recommendations as to where to start looking for other sources of disaster images. He suggested that perhaps it is a good idea to get into contact with local responders such as the fire department and with the federal government for disaster response (FEMA). I planned on contacting departments like FEMA via email, and Mr. White commented that that would be a good idea. I think emails are better since the responder has as much time as he or she needs and the text of information I am trying to communicate is all right there in front of the screen for reference.

**Monday 12/9/2019**

Of course, contacting FEMA and other organizations was not the top priority (but I was glad I did talk about it with Mr. White last Friday), but for now my next milestone is to make our correction feature so that after the user hits the “confirm” button (as seen in the image above),

the internal training dataset would get updated, and the neural network retrains so that it may improve its accuracy. Of course, we would preserve the best trained model in the case that our neural network model suddenly drops in accuracy, in which case we would not have to worry. I illustrated the above example to show what I meant by “confirm,” since the neural network classified the above normal aerial photo of a city as having an earthquake inside of it, hence the need to use my correction feature, which is only partially completed and the other half is what I just talked about.

When we trained a neural network under fast.ai, we had to make sure the dataset to our different disaster images were organized by folder along with ensuring the paths to reach those folders. We would have no problem replicating this structure in Director itself so that a script in Director itself can handle the retraining of the network. However, the problem with this is that fast.ai also relies on a GPU to re-train the network, and Director only computes on a CPU. Therefore, we wanted to find a way to connect Google Drive with Director since obviously Google Colab lends a free GPU to its user. The way my partner and I found was a Python library called PyDrive. My partner found a well-explained online tutorial on how to connect a local Python script to manage Google Drive files remotely. He was able to successfully connect a local Python script on his machine to Google Drive, and I wanted to make sure I was able to replicate his results, but instead of just utilizing the code he already made (which was actually just a copy / paste with a few edits to match the setup / path to certain essential folders on Google Drive (i.e. a folder for the best trained model, for the last trained model, and the dataset) from the actual tutorial he sent me), I also wanted to gain some familiarity with PyDrive as well.

So, following through the tutorial, one of the first instructions is to set up my first project on “Google Cloud Platform.” Apparently, when I attempted to start a new project, like the instructions in the tutorial said, I was not able to since I did not select an organization for my project, even though the instructions said it did not matter if you worked under an organization or not.

I decided to hold off familiarity with PyDrive until later. I had no idea how to get myself out of that issue. The following diagram may be helpful for fully understanding our next milestone in picture:

**Wednesday 12/11/2019**

The solution to the problem of not being able to start my own project was to connect to Connor’s existing Google Cloud Platform project by using a credentials .json file. However, I was faced with initial error as my code attempted to connect to his project. More specifically, my computer, at a local level, was not able to access the folder that contained the best model, last trained model, and dataset folders. After pondering for a while, I thought maybe it was because since the folders were not shared with my Google Drive, that’s why the code was not working. Once Connor did share the files, I tried the code once more a day later (so on Thursday, 12/12/2019) and it finally worked and was able to retrieve an “export.pkl” from one of the Google Drive folders specified in the code.