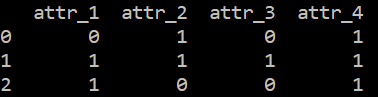
**Friday 2/21/2020**

Today, I decided to get back on focusing on the multi-label train part. The next step I focused on was setting up a Pandas data frame function that could give me the “tags” column, where I would have annotations separate by space rather than binary 1s and 0s I had in place. (For reference, I had a screenshot of the “tags” column in journal report 17.) I kept searching for possibly related functions, and I found that the .apply() method was the most relevant at the time. The .apply() function allowed me to make a new column/row based on the currently existing rows/columns respectively. I wanted to use the positions of the 1s and 0s on each row so that I could get the “tags” column.

I made a sample data frame for testing purposes (see the screenshot below):



Then, I figured out the function I needed to insert into the .apply() method, which was the following:

def add\_atr\_col(pand\_row):

col\_names = list(real\_df.columns)

curr\_row = list(pand\_row)

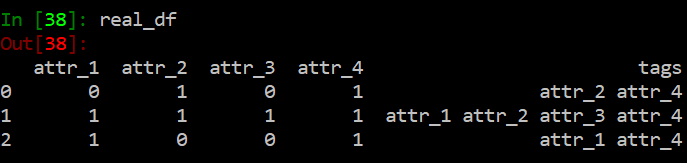
attr\_list = [col\_names[i] for i in range(len(curr\_row)) if curr\_row[i] == 1]

return ‘’.join(attr\_list)

The call to create the “tags” column looked like this:

real\_df[‘tags’] = real\_df.apply(add\_atr\_col, axis = 1)

Figuring out this function took me a while, but in the end, it did exactly what it was supposed to do:



I then thought all I needed was to apply my function to the “all.csv” data frame where the real annotations were stored. However, before that, I wanted to consider if I had all the variables I wanted to have. So far, what I had is the following: “earthquake, flooding, fire, hurricane, normal, bridge, building\_damage, lava, roads, utilities, snow, low\_vegetation, high\_vegetation, river.” I debated in my head for a little while whether to keep the “snow” column. One reason to get rid of it was that the vast majority of my images did not have snow in them. However, I also thought about how snow storms might be another important category to classify.

**Monday 2/24/2020**

I navigated to the “all.csv” file, and kept thinking about whether to get rid of the “snow” category. After a discussion with my partner, we agreed to cancel that possibility. The disasters our current neural network could classify were sufficient and perhaps adding more categories would complicate the learning process of the multi-label neural network.

I also thought about whether to train one giant multi-label network that could classify all features at once (inclusive of the disaster type) or to train a multi-label neural network for each disaster category (essentially multiple multi-label neural networks in total.) I originally firmly decided on the former idea, but then I considered how our current neural network, though only capable of identifying the disaster type, was very skilled at getting the disaster type correct. When I trained it, I was lucky to achieve an incredibly low error rate. My thinking of luck also came from the fact that when my research partner also trained a neural network of the same type, the lowest error rate it could achieve was 30%. With a multi-label neural network, however, since there are more possible classifications, training it on both the disaster type and the details makes getting an error rate anywhere near the one I received for our current neural network an exponentially lower chance.

**Wednesday 2/26/2020**

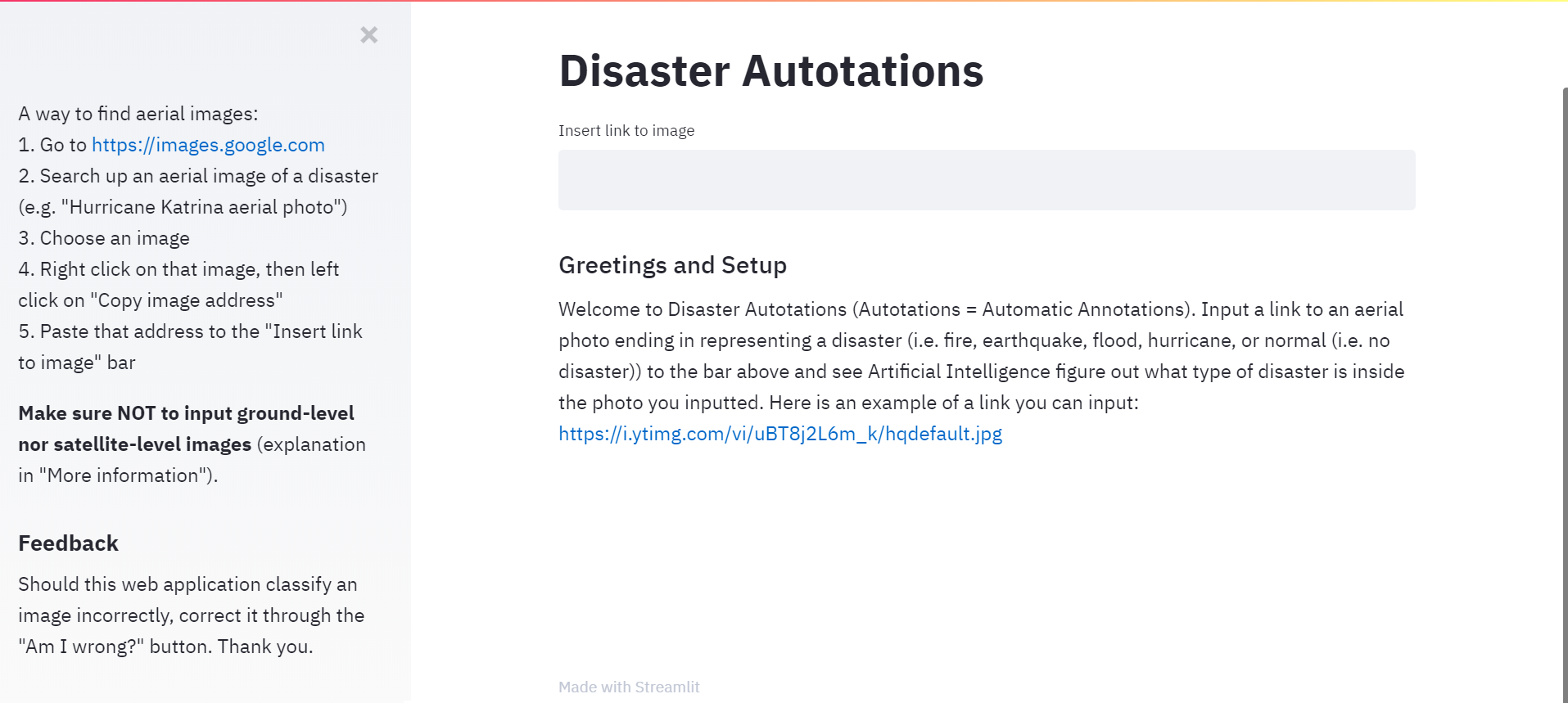
I noticed that in the “all.csv” file, where all the multi-label annotations were held, the image annotations for “fire59.jpg” through “fire99.jpg” were missing. I went into the “fire\_annotations.csv” file, with the intention of copying the “fire59.jpg” through “fire99.jpg” annotations from Google Sheets and pasting them into the “all.csv” file, where the combined annotations for all disaster types were found.

Then, Dr. Gabor came over to check on my user interface. I told him while I did not make any progress on my user interface, there were maybe two ways I could get rid of the white space at the top. One way was to embed HTML code in the “st.markdown(‘[insert HTML here]’, unsafe\_allow\_html = True)” method. Another way was to change the fundamental source code of Streamlit. Dr. Gabor commented that none of these ways were in the proper direction of his advice to me earlier. He then explained to me how one could insert Javascript code inside of an HTML file, and use that code to get rid of the white space at the top. I then asked him where exactly I could find that HTML file. He then asked whether I was editing the HTML code myself or if I was using another application to my web page. I told him it was the latter. We then explored the Streamlit documentation to see if there was a more proper way of inserting HTML tags/attributes into my Streamlit code. Unfortunately, we found nothing useful. We then checked the community forums. There was still no luck. Dr. Gabor then suggested to me that I could post a question about getting rid of the white space at the top myself within the Streamlit community. However, because I had other tasks to focus on (i.e. training a multi-label neural network), Dr. Gabor decided to postpone updates of my user interface.

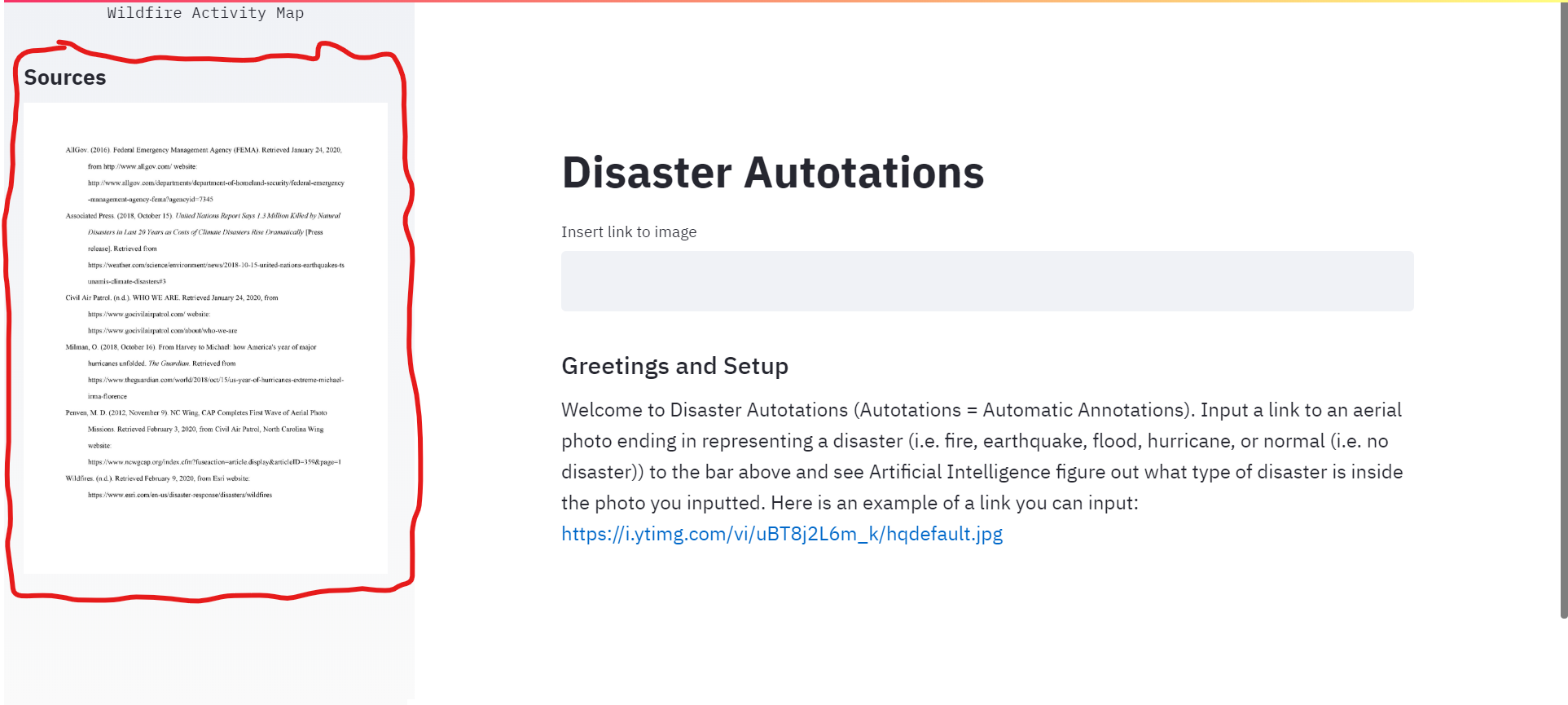
In hindsight, I think the experience going through the Streamlit community and documentation was exciting, fun, and overall, very educational.

**Thursday 2/27/2020**

Considering our discussion about the white space to the sides of our web application, my partner implemented a sidebar with the extraneous information I originally hid under the “More information” checkbox. Here was how it looked:



Clearly, there was still a significant amount of whitespace present to the left-hand side and right-hand sides. In addition, the sidebar made my sources too small to see, as seen circled from the below screenshot:



Then again, I could move the sources from the sidebar to the main page. However, since I did not like the idea of a sidebar for extraneous information (and instead, I would use a sidebar to link to other pages. In this case, I could have the sidebar contain a hyperlink to “<https://streamlit2.sites.tjhsst.edu/about>,” where I would have the content for the extraneous information.), I moved back the extraneous information under the “More information” checkbox. I thought it was very considerate of my partner to comment out the code where I had the checkbox and the extraneous information under it. My partner, while he liked having the sidebar, agreed to maintain the checkbox instead of a sidebar.

I may consider a sidebar for extraneous information, however, if there is a way to adjust the sidebar size, which seemed not possible within Streamlit’s capabilities.

My partner realized why “fire59.jpg” through “fire99.jpg” for the fire annotations were missing: those images seemed to have “.jpeg” extensions instead of “.jpg” extensions. So, in turn, either the script to combine the annotations together deleted the “.jpeg” annotations or Connor might have deleted them himself. Connor could not recall which of the two events occurred. I decided to double-check on this matter tomorrow: Friday, 2/28/2020.