Samuel Bernal

Rao

INFO 4160

14 November 2021

Mini Project 2 – Notes

**What was the fun part and what were the challenges you faced?**

My favorite part of the project was the reiterative testing of the whole API and website model. Once I got the basic structure down of displaying web content and manipulating web inputs, I enjoyed troubleshooting the errors I had with website crashes or missing values in my Flask app.

After developing my app and uploading my classification model, I tried running the URL and my website immediately crashed. The error I had in my app was that my model for some reason had an array index of 22 inputs even though I only had 21 feature inputs total. I fixed this by indexing my imported dataframe from column zero.

Another problem I ran into was how to reset the feature dataframe for consecutive tests and how to create a database without crashing the app for each run. This was tricky at first, but I used an array of zeros to reset my feature input dataframe for each run, and I used if statements to only create a database if there is not already a database file in the directory.

**What did you learn from this mini project overall?**

Overall, I believe a developed a greater skill in strategically planning my model and API creation. For this Mini Project 2, I did not just head in and start throwing together the models and APIs, but I took the time to write some pseudo code and the workflow for how I would approach the process. By writing and imagining pseudo code first, I was able to solve the project problem rationally and logically instead of being lost and confused between the multiple files. After being able to put the whole project together, I was also able to appreciate all the query, predictive modeling, and HTML skills that I have learned. When they were all combined, I realized how satisfying and cool the IoT process can be when all the parts work properly.

**Which algorithm did you choose and how or why did you come to that conclusion?**

I decided to use the Random Forest Classifier due to the multiple classification outputs of the project. There were also 21 total feature engineered inputs, so I thought it would be useful to use an ensemble classifier which would create multiple trees and randomly test out the combinations of different features and training data. The multiple outputs and inputs of this project made it seem to me that the Random Forest Classifier would perform well in interpreting and modeling the data.

**How did you go about testing your project at different stages of development?**

The two main stages I split my project into was the Flask app testing and the SQLdb storage. For the flask app, I mostly used Jupyter notebook to simulate all the array indexing, list creations, and model inputs. By testing all my functions within Juptyer, I was able to make sure that the code worked properly when developed in the flask app. I would then run the server to see what errors or issues I had to fix.

For the data storage, I had to use the SQLdb viewer to see if my table and column names were created properly. Every time I submitted data on the server, I would refresh the database tool to make sure that my inputs were committing to storage in my db file.

By having immediate feedback on my code outputs for both Jupyter and Flask, it was easier for my to debug my code and to find errors within my project.