- First, submit a link to your final project.
- Second, imagine you are a consultant providing an overview of the application you built to a client. Include advantages and disadvantages of the application, areas of improvement, and new technologies moving forward.

The original concept that led to this project was the desire to be able to determine the accuracy of complimentary sensors designed to detect similar phenomena absent a source of ground truth. This capability has industry applications in use cases featuring ambiguous information such as in field tests and contested environments. While a robust community of practice exists for determining accuracy of sensors in laboratory environments in which ground truth is known, a source of ground truth is commonly accepted as a necessity for being able to determine accuracy. My hope is that, through Bayesian Reasoning, complimentary sensors associated with each other through coordinated analysis can indicate the propensity for the sensors to provide information which agrees with each other, and thereby a proxy for ground truth can be established and the presumed accuracy of the sensor can therefore be reported.

This project is little more than a stepping stone towards a proof of concept. It's not even a complete first step towards a workable product, and in fact that was in some ways the most difficult aspect of it for me. When I originally conceived of the project, I was going to have multiple microservices, all running and monitored and evaluated in real time, interacting with a dedicated relational database, using at least one specific algorithm handpicked for the intended use case. Instead, all I was able to demonstrate was a static data use case of the algorithm without live services to monitor the data from.

That's not to say that this has no practical application. On the contrary, the Naive Bayes algorithm demonstrated in this project is the first missing piece in the larger implementation for me, and the other initial pieces, i.e. the harvester simulations, analysis, and database services, are all relatively well understood by me already. Now that I have the algorithm running in this limited use case, fleshing out the functionality around it should just be an exercise in patience and determination rather than one requiring novel solutions.

Even after the simulated services are incorporated into the implementation, there will still be an extensive backlog of additional features to deploy. First, fine tuning algorithm performance and expanding test cases with varying sets of simulated harvesters. Second, replacing the simulated harvesters with genuine datasets, either static or simulated-real-time. Third, adjusting the number of simulated harvesters (my expectation is that the ability of an algorithm to determine comparable accuracy of various sensor systems will increase with the number of sensor systems to compare values against). Those are just the first changes that immediately come to mind from a data quality/validity perspective - practically speaking there would first need to be an exploration and planning phase to determine what specific changes to make after the initial proof of concept is completed.

In addition to those data-oriented fixes, the project here could benefit from any number of additional software engineering improvements. First, the deployment as it currently sits is quite manually intensive and prone to reliability issues. I didn't take a test-driven development approach, so even the functional requirements are in question to me; that is, I'm unsure how to validate whether or not the code actually technically works. Sure, I receive a value when I run the model, but that's hardly sufficient for the broader use case I had targeted for this project. Incorporating a more procedurally robust development process than the one I used here for this initial exploration towards a proof of concept will result in a much more traceable, professional proof of concept which can be further iterated on in the future.

Additionally, the technology used in this initial version of the project is relatively simple. A more mature version built for continuous operation, integration and deployment would (in addition to the tests mentioned above) be deployed in a production environment cluster of services using Kubernetes, whereby each change in the production codebase would reset the node running the code and allow a new & improved instance of the scripts to begin running with minimal manual oversight. Error logging in the script, picked up through an add on like Prom-tail, would benefit the metrics being collected by Prometheus and thereby allow more informed visualizations and alerts by Grafana.

One of the things I appreciate the most about this project has been how the tempering of my expectations within the time constraints of a single semester has allowed me to more accurately assess the amount of work required to eventually demonstrate the mature version of this concept, i.e. one which uses real sensor data to provide real evaluations in real time to real end users. All of the various features I'd like to incorporate into the design couldn't possibly be accomplished in a single semester, especially considering I had no prior familiarity with the algorithm itself. That being said, this project has allowed me to come to the conclusion that this effort is at least worth trying up until experimental trials could determine the actual efficacy of the methodology in determining coordinated ground truth against complimentary sensors. I'm now looking forward to continuing to develop the idea at work - I already know exactly the dataset I can train and test this model on, and this project has provided me with enough info that I can more readily coordinate resources and funding approval to make this proof of concept more of a reality.