Capstone:
Pick Your Deployment Method

13.03.2025

Bruce Walker, UCSD MLE/AI Bootcamp

Chosen Method: Containerized API

Options Considered

1) **Hosted Web Service** - A website would be provided where an end-user could upload network traffic files for processing. The prediction would be performed in the cloud with results being written back out to files (records identified by a UID provided in the original file). The end-user would then retrieve the results files within a specified timeframe before the files were purged. (Uploaded files would be purged as soon as predictions are done.)
2) **Hosted API** - API interface would be provided to access a hosted prediction service. End-user would write code to interface with the API. network traffic records would be submitted to the API individually or in batches. Predictions would be made and results returned via the API interface.
3) **Containerized API** - API interface would be created and hosted via a container (Docker, Kubernetes, etc.). End-user can choose whether to host the container locally or in the cloud. Usage procedure would be the same as the Hosted API option but the API's hosting location is chosen by the end-user.

Factors Affecting Deployment Method Choice

1) **Bandwidth** - Transferring millions or billions of records to/from a cloud-hosted solution in a timely manner would require significant bandwidth in the network pipeline between the end-user and the hosted solution.
2) **File Management** - Having the store/purge file management for the hosted web service could lead to problems. If an end-user does not retrieve their files in a timely manner they may wind up needing to re-upload files for processing thus wasting resources. Also, the storage capacity needs could very significantly depending on how heavily the system is being used.

3) **Security** - Though the information required in the input data file would be fairly anonymous there is still the potential for data that is sent to and/or stored on a cloud host to be accessed and analyzed for unintended – possibly malicious – purposes.
4) **Timing** - Network intrusion/attack analysis could be a time-critical need. The quicker the predictions can be retrieved and analyzed, the more likely an attack/intrusion can be stopped before additional/lasting damage can be done. By allowing the end-user to host the API locally they can eliminate any delay that might be caused by having to send/retrieve the data to/from a cloud host.

Model Lifecycle Management

- **Versioning** - The API will include functions to download trained model versions (updated code as well as updated trained model) from a GitHub repository.
    - New versions will be added to the repository when they are proven to have improved performance and/or efficiency.
    - If the end-user is not using the latest version, the API will alert them that an updated version is available.
    - New container downloads will be pre-configured to use the latest version from the GitHub repository.
- **Customization** - The API will also have a function so the end-user can create a trained model based on their own training data using whichever version of the model they have chosen. That model will be able to be saved within the container for reuse. If not too complicated, I will include functions for the end-user to maintain multiple versions of their self-trained models.