EE / CprE / SE 492 – sdmay21-23
Grid AI
Week 3 Report

2/23/21 – 3/1/21
Client: Dr. Ravikumar Gelli
Advisor: Dr. Ravikumar Gelli

Team Members:

Justin Merkel — *ML Developer, Backend Developer*
Patrick Wenzel — *Frontend Developer*
Abhilash Tripathy — *Frontend Developer*
Karthik Prakash — *Backend Developer*
Abir Mojumder — Frontend/Backend Developer

Weekly Summary

For the past week, the main objective for the frontend team was to be able to get ReactJS and the material-ui-dashboard into one Docker container, the Flask portion into another Docker container, and then be able to run a docker-compose.yml file to be able to run both at the same time. We were able to successfully complete that and now our goal is to be able to add routes in the Flask container that the frontend Docker container can successfully use. The frontend design is starting to populate with information from the backend. We are currently able to visualize the 240 node system, and plan to design layouts to display predictions/current value for a node.

For the backend team, one of the main objectives was to create models for all 240 nodes in the system which we were successfully able to do. We also wanted to get a connection set up between the frontend and our Neo4j database to display the static data that we are storing in the nodes. As said before, we were able to do this and receive requests from the frontend and be able to send all 240 nodes' data to the frontend. Furthermore, data for the neo4j was formatted for easier integration with ML models (through flask routing).

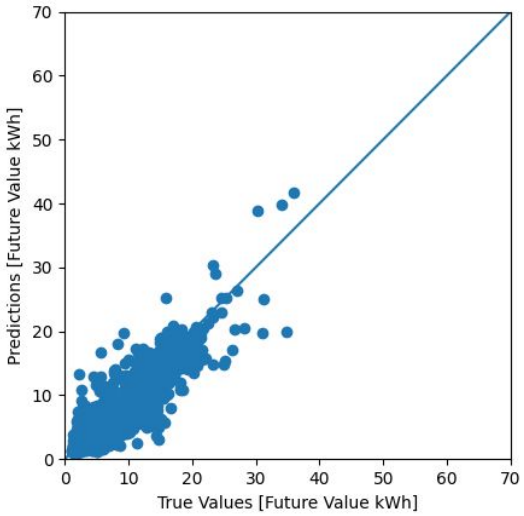
As a whole, the team is also working on our PIRM presentation as we have one on Thursday, March 4th.

Past Week Accomplishments

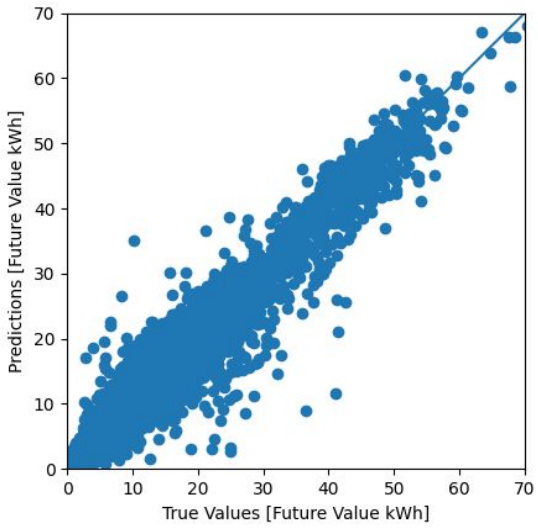
- Getting Docker containers for Flask and ReactJS/material-ui-dashboard working on the frontend and being able to run our application with a docker-compose.yml file.

```
(base) ubuntu@ubuntu-vm:~/Desktop/sdmay21-23/GridAI-Dashboard$ sudo docker-compose up
[sudo] password for ubuntu:
Starting gridai-dashboard_api_1 ... done
Starting gridai-dashboard_client_1 ... done
Attaching to gridai-dashboard_api_1, gridai-dashboard_client_1
api_1      |  * Serving Flask app "app.py" (lazy loading)
api_1      |  * Environment: development
api_1      |  * Debug mode: on
api_1      |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
api_1      |  * Restarting with stat
api_1      |  * Debugger is active!
api_1      |  * Debugger PIN: 308-220-137
client_1   |
client_1   | > material-dashboard-react@1.9.0 start
client_1   | > react-scripts start
client_1   |
client_1   | i ｢wds｣: Project is running at http://172.31.0.3/
client_1   | i ｢wds｣: webpack output is served from /material-dashboard-react
client_1   | i ｢wds｣: Content not from webpack is served from /public
client_1   | i ｢wds｣: 404s will fallback to /material-dashboard-react/
client_1   | Starting the development server...
client_1   |
client_1   | Compiled successfully!
client_1   |
client_1   | You can now view material-dashboard-react in the browser.
client_1   |
client_1   |   Local:            http://localhost:3000/material-dashboard-react
client_1   |   On Your Network:  http://172.31.0.3:3000/material-dashboard-react
client_1   |
client_1   | Note that the development build is not optimized.
client_1   | To create a production build, use npm run build.
client_1   |
```

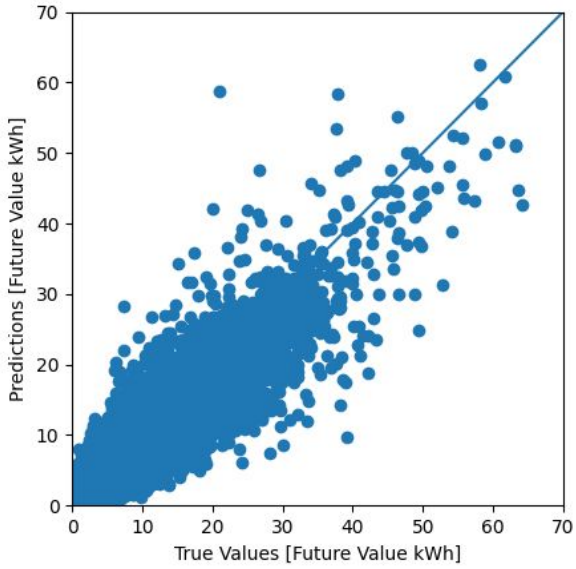
- Justin- Expanded the linear regression models so that there is one linear regression model per transformer type. This means that there are models for all 240 nodes in the system for power consumption. Additionally, I discovered how to easily save and load models into python for the backend.
    - Single Phase



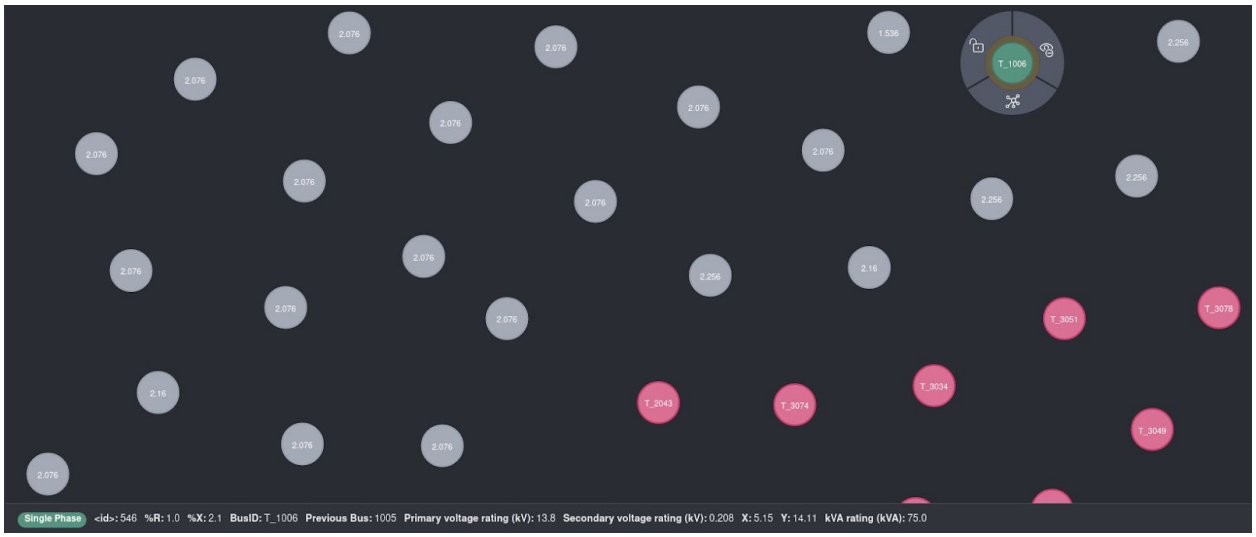
    - Three-Phase



- Single Phase Center Tapped

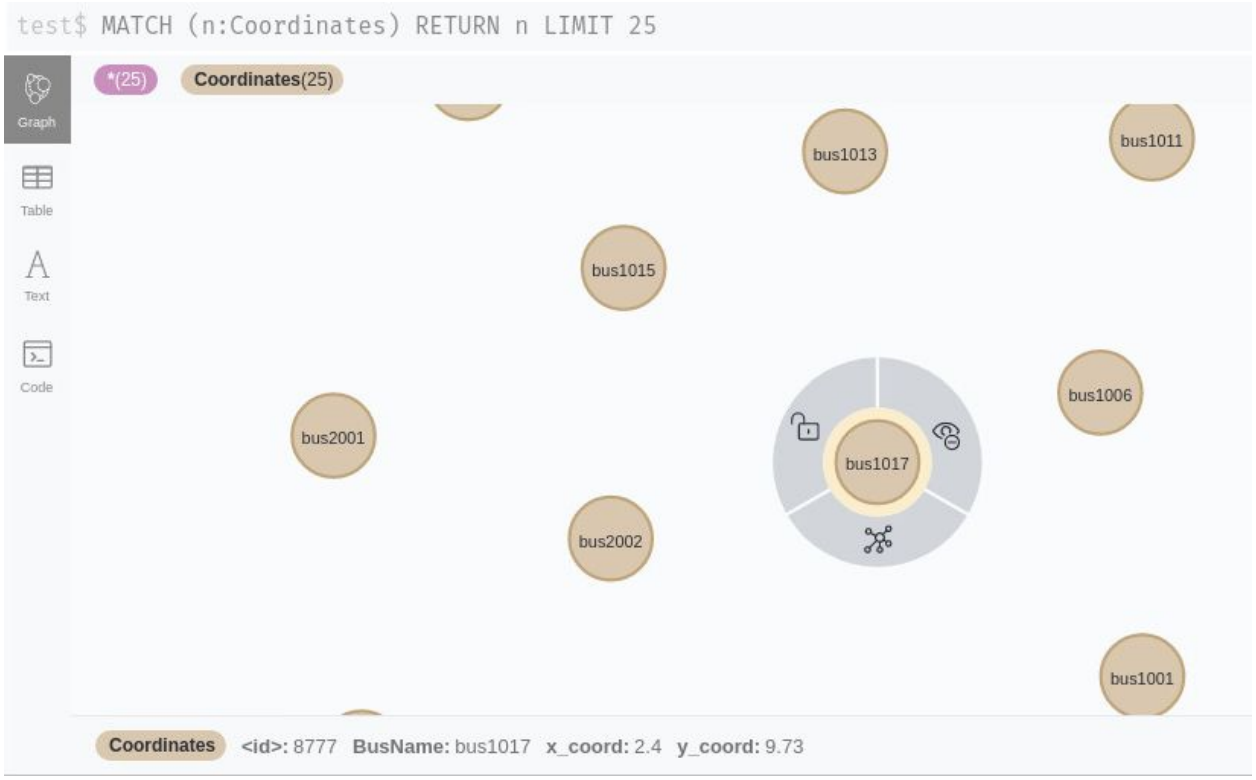


- Successfully formatted all of the static node data into Neo4j
  - Node data for all transformers in the network are loaded
  - Each Node represents the Bus from a transformer and contains bus number, voltage ratings, power rating, location of node, etc. Every Node is labeled as the type of transformer as well (Single-Phase, Single-Phase Center Tapped, and Three-Phase)



Abir - Node visualization in React

- Neo4j database uploaded with 240 nodes' x and y coordinates for visualization of the power grid in the webapp.



- Data flow: neo4j -> flask -> React
- The coordinates for the nodes show where they are placed relative(position) to each other. (not distance however).

Populated bus data in tabular form:

BUS NAME	VALUE
BUS 10	7.136
BUS 10	7.077999999999999
BUS 10	5.9910000000000005
BUS 10	7.03
BUS 10	6.042999999999999
BUS 10	6.075
BUS 10	5.706
BUS 10	5.809
BUS 10	5.836
BUS 10	6.409
BUS 10	6.457999999999999
BUS 10	6.3629999999999995
BUS 10	5.682
BUS 10	8.957
BUS 10	6.855
BUS 10	10.261
BUS 10	9.954
BUS 10	8.571
BUS 10	7.638999999999999
BUS 10	7.444
BUS 10	9.074
BUS 10	9.46
BUS 10	6.916
BUS 10	6.425
BUS 10	5.704

Pending Issues

- Creating routes that work between the two Docker containers for the frontend that will route between the different dashboard pages.
- Configure time-series data in database to allow backend querying
    - Potentially setup MySQL instance for this
- Create endpoints for pulling specific data from Neo4j
- ML model complete integration with backend.
- Get line information from neo4j to display connections between nodes.
- Design homepage, design how to display data for each node (userfriendly).

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Patrick Wenzel	Set up the frontend so that the Flask api routing had its own Docker container and the ReactJS client had its own Docker container. Created a docker-compose.yml file that runs both of those Docker containers at the same time. Also set up a test route to verify that the two Docker containers can communicate with each other.	7	20
Justin Merkel	Expanded the number of models to meet all 240 nodes in the system. Helped Backend understand ML database requirements.	7	23
Abir Mojumder	React frontend now uses d3 Component to render the 240 nodes grid.	8	20
Karthik Prakash	Formatted Neo4j data to easily integrate with ML models	8	20
Abhilash Tripathy	Making endpoints on the frontend that connect with api on backend populating a table on the frontend displaying the bus data and their values	8	20

Plans for Coming Week

- Patrick - Be able to actually route the different pages in Flask so that the dashboard routes that way instead of through the NodeJS api it came with.
- Justin-Work on implementing a Logistic regression model that will be able to detect anomaly chance.

- Abir - Use line information from neo4j node relationships to display Bus line connections on the frontend.
- Karthik - Import more data in a reasonable format to Neo4j (node relationships and time-series), potentially setup MySQL docker instance for time-series data, and work on backend endpoints
- Abhilash - Consolidating multiple data insights from the backend to a user-friendly format in the frontend.