GridAI: Cloud-Based Machine/Deep Learning For Power Grid Data Analytics

sdmay21-23

Faculty Advisor & Client:

Dr. Gelli Ravikumar

Team Members:

Abir Mojumder

Karthik Prakash

Justin Merkel

Patrick Wenzel

Abhilash Tripathy

Project Purpose

Problem Statement:

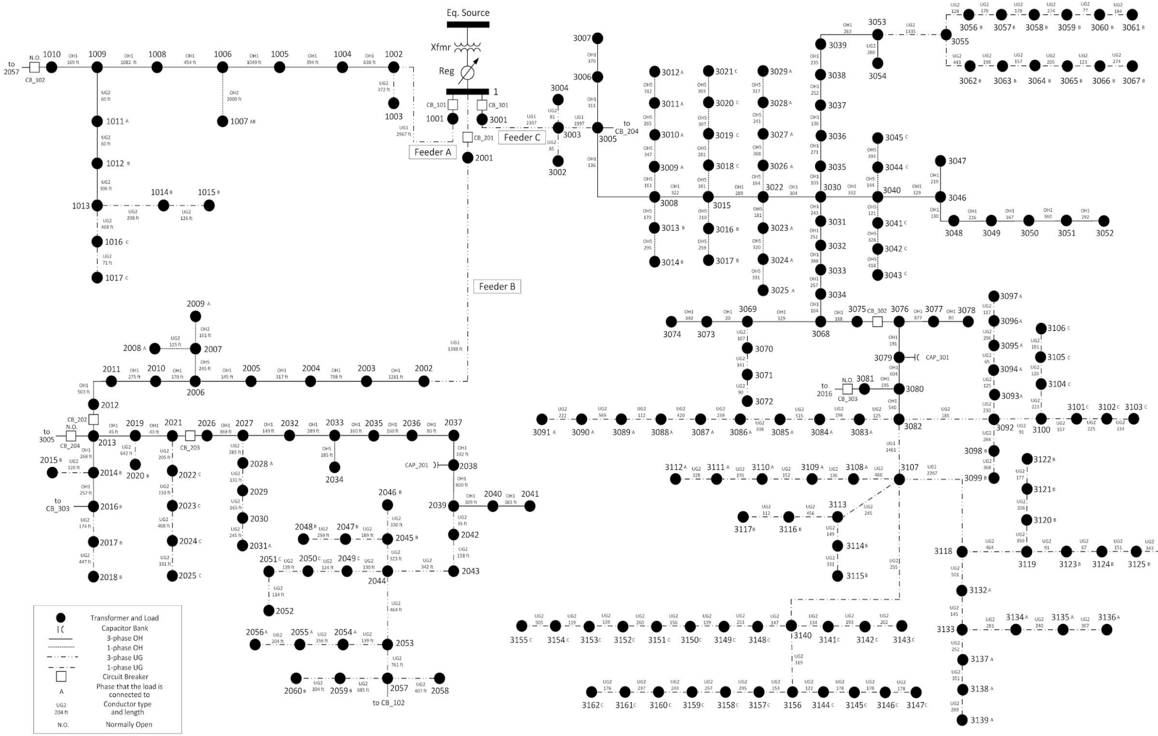
- Power grids are complex and critical infrastructure which leaves them vulnerable to instability and attack.

Solution Approach:

- Develop a web application that implements a Machine Learning model to analyze power grid data and detect anomalies in power usage.

Project Context

- Use Machine Learning on a simulated power grid to provide analytics and anomaly detection
  - Every node has some power output data associated
  - Static electrical properties
  - Location and connections in network



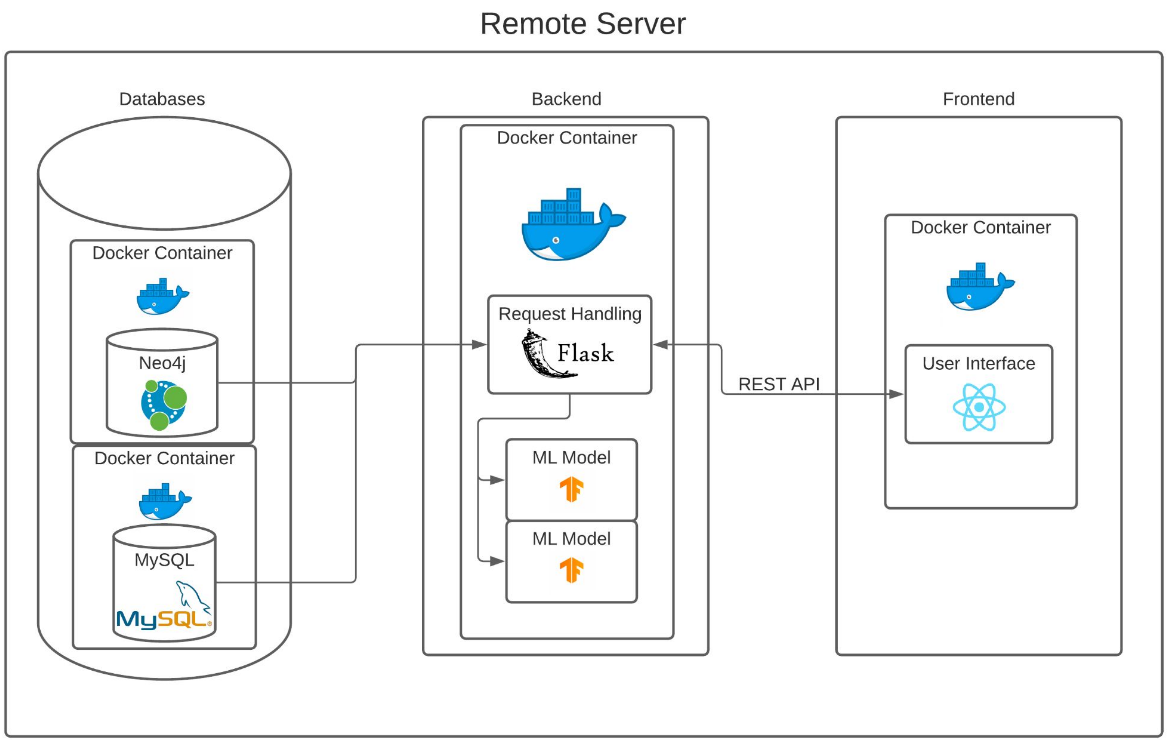
Project Functional Requirements

- Machine Learning Requirements. The ML models should:
  - Use the most recent kWh value from the grid in the predictions.
  - Predict the next kWh output for each node in the grid.
  - Predict the probability of each anomaly class.
  - Use convolutional layers for deep learning.
- Front-end Requirements. The front-end should:
  - Receive data from the back-end
  - Visualize data on a dashboard:
    - Graph-based visualization
    - Geographical representation of the power grid
    - Charts for each node's history and predictions
    - Tabular data showing anomaly status for every node
  - Interface directly with the back-end
- Back-end Requirements The back-end should:
  - The server-side application will handle all data communication with the databases
  - All data processing, including ML analysis, will occur on the back-end
  - Provide real-time data to front-end

Project Non-Functional Requirements

- Clear documentation
  - Allows future teams to improve on the baseline
- Maintainability
  - Modular coding and Docker containers
- Scalability
  - ML models are generalized predictors for nodes.
- Response time
  - Lightweight front-end to accommodate response rate of work heavy back-end

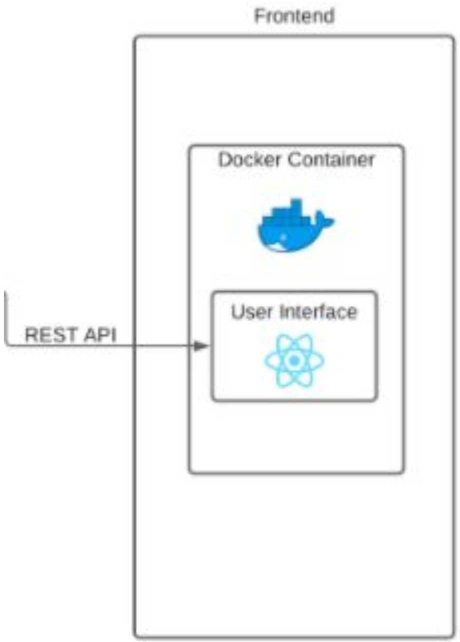
High Level Design



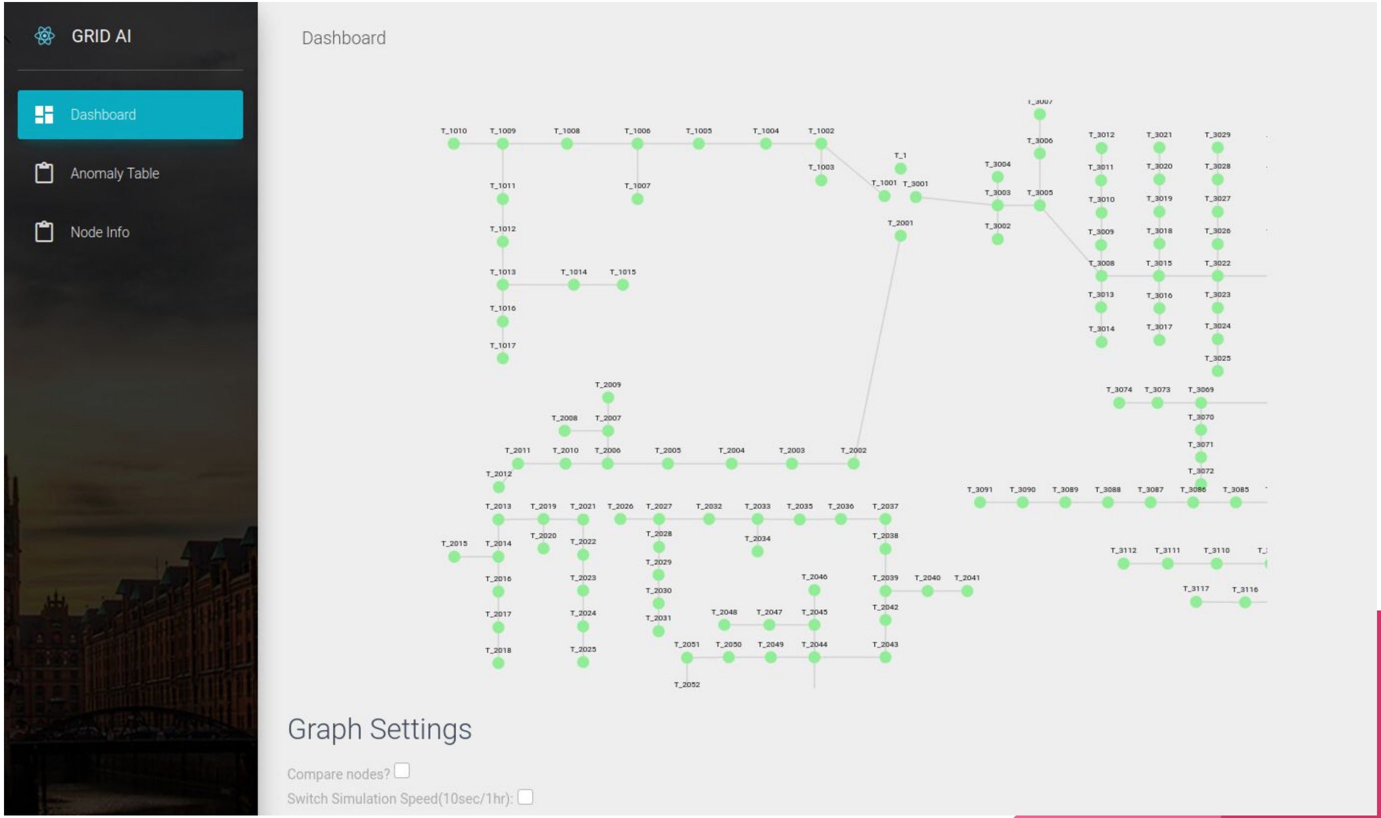
Front-end Design/Implementation

- Main Requirements:
  - Communicates with back-end
  - Accurate visualizations of data
  - Multiple kinds of visualizations
  - Clean-looking and easy to navigate

- One functional module
  - ReactJS frontend



Dashboard - Home Screen

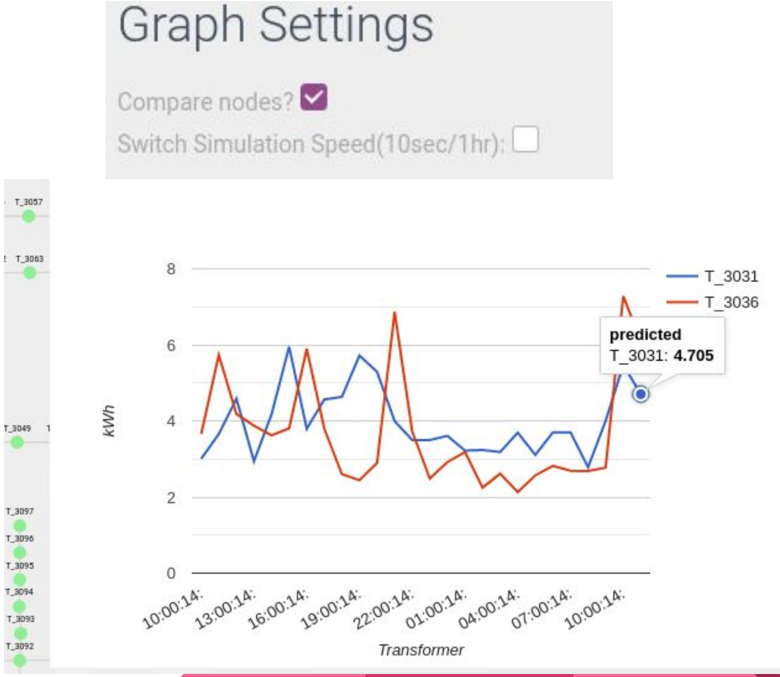


React D3 Graph

- Nodes with Links
  - Use Rest API to get coordinates and links between transformers
  - Plot and link nodes
  - Visualize 240-Node grid from schematic.
- Interactive graph
  - Drag and zoom to adjust view
  - Click on nodes to display information

Time-series data display and Comparison

- 24-hr History Data
  - Click a node to display past 24-hrs kWh readings
  - Predicted kWh value shown based on ML prediction models



- Graph Settings
  - Compare Nodes option allows comparison of time-series history of 2 selected nodes
  - Option to switch simulation update speed - 1 hour(realtime) or 10 seconds

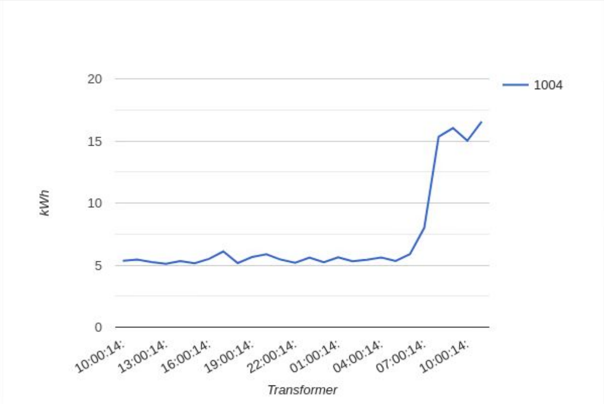
Detailed Node Information

- Display Node Properties : Primary/Secondary voltage, Time running, Phase type, Resistance, Reactance, etc. (Properties vary based on phase type)

Please enter a Bus number

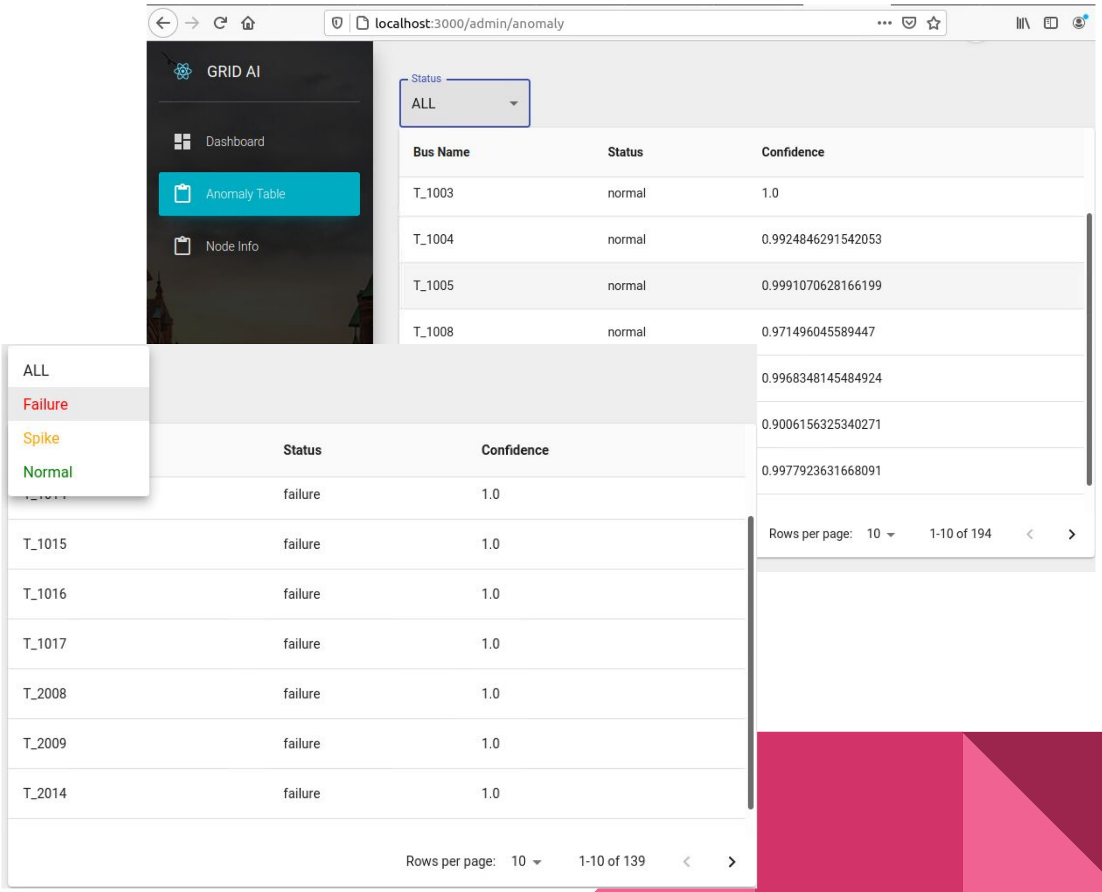
1004 Go

Bus Name	Type	Current Value	Previous Value	Primary voltage rating (kV)	Secondary voltage rating (kV)	kVA rating (kVA)	%R	%X	Time
T_1004	Three Phase	15.011	5.233	13.8	0.208	75	2.27	1.91	4 months, 30 days, 10 hours



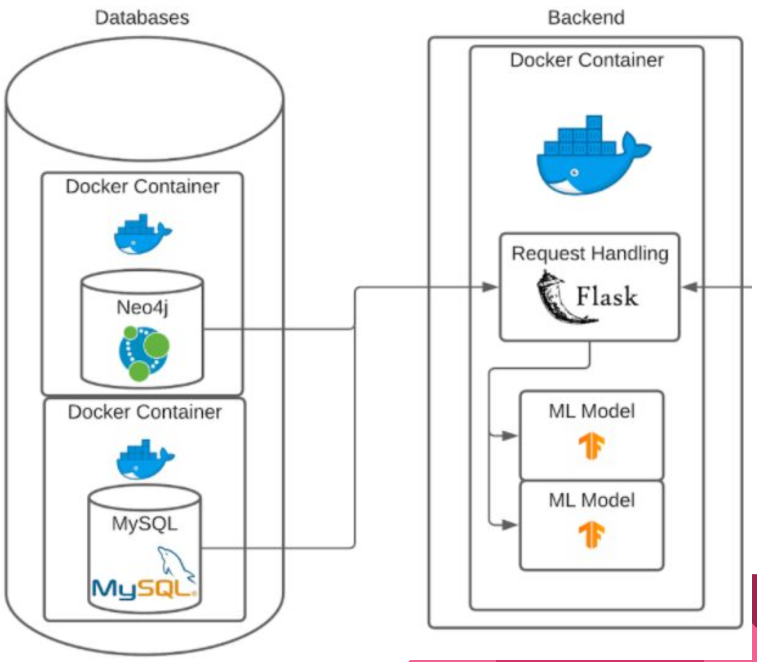
Anomaly Data

- To view the predicted status of the node.
- Display the confidence level of the prediction in a table format.
- View the nodes by their Status “Normal”, “Spike”, or “Failure.”



Back-end Design/Implementation

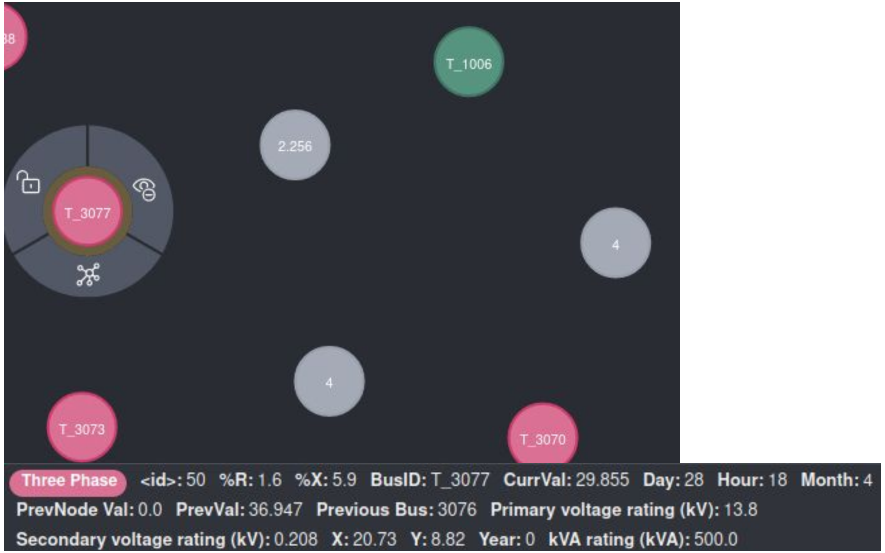
- Main Requirements:
    - Data processing
    - Supply real-time data
- Three functional modules
    - Neo4j database
    - MySQL database
    - REST API



Neo4j Database

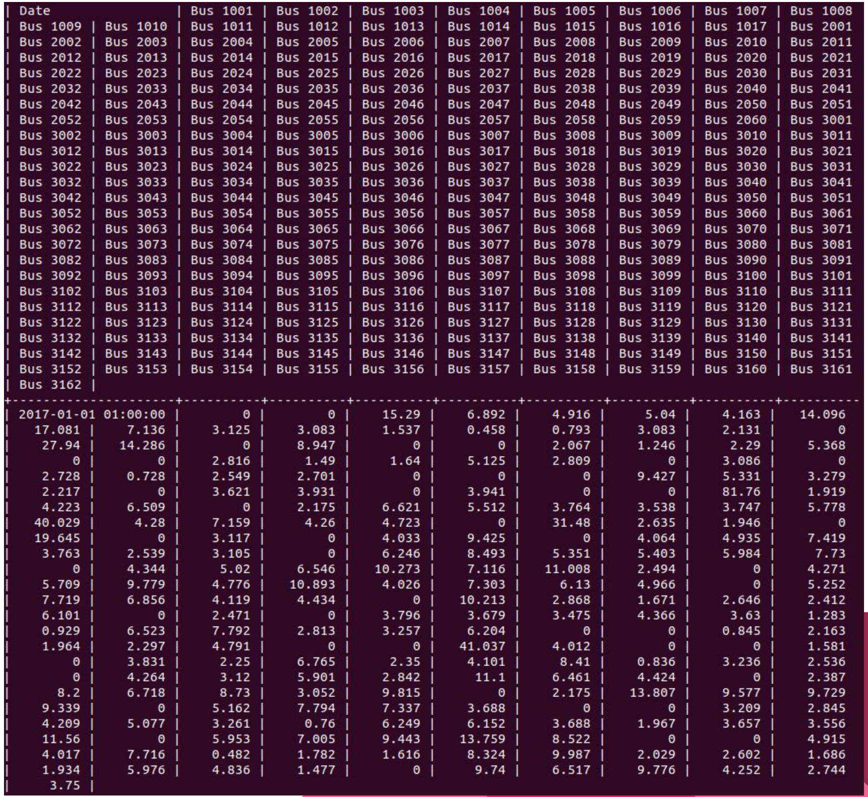
- Graph-based database
  - Fast query response times
  - Practical power grid depiction

- Represent transformers in power grid
  - Store transformer features for ML
  - Up-to-date kWh output properties



MySQL Database

- Developer-friendly
  - Previous experience with SQL
  - Well-documented
  - Simple integration with server-side application
- Scalable
  - Preserve record of time-series data
  - Future-proof for larger dataset



```
| Date                | Bus 1001 | Bus 1002 | Bus 1003 | Bus 1004 | Bus 1005 | Bus 1006 | Bus 1007 | Bus 1008
| Bus 1009 | Bus 1010 | Bus 1011 | Bus 1012 | Bus 1013 | Bus 1014 | Bus 1015 | Bus 1016 | Bus 1017 | Bus 2001
| Bus 2002 | Bus 2003 | Bus 2004 | Bus 2005 | Bus 2006 | Bus 2007 | Bus 2008 | Bus 2009 | Bus 2010 | Bus 2011
| Bus 2012 | Bus 2013 | Bus 2014 | Bus 2015 | Bus 2016 | Bus 2017 | Bus 2018 | Bus 2019 | Bus 2020 | Bus 2021
| Bus 2022 | Bus 2023 | Bus 2024 | Bus 2025 | Bus 2026 | Bus 2027 | Bus 2028 | Bus 2029 | Bus 2030 | Bus 2031
| Bus 2032 | Bus 2033 | Bus 2034 | Bus 2035 | Bus 2036 | Bus 2037 | Bus 2038 | Bus 2039 | Bus 2040 | Bus 2041
| Bus 2042 | Bus 2043 | Bus 2044 | Bus 2045 | Bus 2046 | Bus 2047 | Bus 2048 | Bus 2049 | Bus 2050 | Bus 2051
| Bus 2052 | Bus 2053 | Bus 2054 | Bus 2055 | Bus 2056 | Bus 2057 | Bus 2058 | Bus 2059 | Bus 2060 | Bus 3001
| Bus 3002 | Bus 3003 | Bus 3004 | Bus 3005 | Bus 3006 | Bus 3007 | Bus 3008 | Bus 3009 | Bus 3010 | Bus 3011
| Bus 3012 | Bus 3013 | Bus 3014 | Bus 3015 | Bus 3016 | Bus 3017 | Bus 3018 | Bus 3019 | Bus 3020 | Bus 3021
| Bus 3022 | Bus 3023 | Bus 3024 | Bus 3025 | Bus 3026 | Bus 3027 | Bus 3028 | Bus 3029 | Bus 3030 | Bus 3031
| Bus 3032 | Bus 3033 | Bus 3034 | Bus 3035 | Bus 3036 | Bus 3037 | Bus 3038 | Bus 3039 | Bus 3040 | Bus 3041
| Bus 3042 | Bus 3043 | Bus 3044 | Bus 3045 | Bus 3046 | Bus 3047 | Bus 3048 | Bus 3049 | Bus 3050 | Bus 3051
| Bus 3052 | Bus 3053 | Bus 3054 | Bus 3055 | Bus 3056 | Bus 3057 | Bus 3058 | Bus 3059 | Bus 3060 | Bus 3061
| Bus 3062 | Bus 3063 | Bus 3064 | Bus 3065 | Bus 3066 | Bus 3067 | Bus 3068 | Bus 3069 | Bus 3070 | Bus 3071
| Bus 3072 | Bus 3073 | Bus 3074 | Bus 3075 | Bus 3076 | Bus 3077 | Bus 3078 | Bus 3079 | Bus 3080 | Bus 3081
| Bus 3082 | Bus 3083 | Bus 3084 | Bus 3085 | Bus 3086 | Bus 3087 | Bus 3088 | Bus 3089 | Bus 3090 | Bus 3091
| Bus 3092 | Bus 3093 | Bus 3094 | Bus 3095 | Bus 3096 | Bus 3097 | Bus 3098 | Bus 3099 | Bus 3100 | Bus 3101
| Bus 3102 | Bus 3103 | Bus 3104 | Bus 3105 | Bus 3106 | Bus 3107 | Bus 3108 | Bus 3109 | Bus 3110 | Bus 3111
| Bus 3112 | Bus 3113 | Bus 3114 | Bus 3115 | Bus 3116 | Bus 3117 | Bus 3118 | Bus 3119 | Bus 3120 | Bus 3121
| Bus 3122 | Bus 3123 | Bus 3124 | Bus 3125 | Bus 3126 | Bus 3127 | Bus 3128 | Bus 3129 | Bus 3130 | Bus 3131
| Bus 3132 | Bus 3133 | Bus 3134 | Bus 3135 | Bus 3136 | Bus 3137 | Bus 3138 | Bus 3139 | Bus 3140 | Bus 3141
| Bus 3142 | Bus 3143 | Bus 3144 | Bus 3145 | Bus 3146 | Bus 3147 | Bus 3148 | Bus 3149 | Bus 3150 | Bus 3151
| Bus 3152 | Bus 3153 | Bus 3154 | Bus 3155 | Bus 3156 | Bus 3157 | Bus 3158 | Bus 3159 | Bus 3160 | Bus 3161
| Bus 3162 |
+---------------------+----------+----------+----------+----------+----------+----------+----------+----------
| 2017-01-01 01:00:00 |        0 |        0 |    15.29 |    6.892 |    4.916 |     5.04 |    4.163 |   14.096
|   17.081 |    7.136 |    3.125 |    3.083 |    1.537 |    0.458 |    0.793 |    3.083 |    2.131 |        0
|    27.94 |   14.286 |        0 |    8.947 |        0 |        0 |    2.067 |    1.246 |     2.29 |    5.368
|        0 |        0 |    2.816 |     1.49 |     1.64 |    5.125 |    2.809 |        0 |    3.086 |        0
|    2.728 |    0.728 |    2.549 |    2.701 |        0 |        0 |        0 |    9.427 |    5.331 |    3.279
|    2.217 |        0 |    3.621 |    3.931 |        0 |    3.941 |        0 |        0 |    81.76 |    1.919
|    4.223 |    6.509 |        0 |    2.175 |    6.621 |    5.512 |    3.764 |    3.538 |    3.747 |    5.778
|   40.029 |     4.28 |    7.159 |     4.26 |    4.723 |        0 |    31.48 |    2.635 |    1.946 |        0
|   19.645 |        0 |    3.117 |        0 |    4.033 |    9.425 |        0 |    4.064 |    4.935 |    7.419
|    3.763 |    2.539 |    3.105 |        0 |    6.246 |    8.493 |    5.351 |    5.403 |    5.984 |     7.73
|        0 |    4.344 |     5.02 |    6.546 |   10.273 |    7.116 |   11.008 |    2.494 |        0 |    4.271
|    5.709 |    9.779 |    4.776 |   10.893 |    4.026 |    7.303 |     6.13 |    4.966 |        0 |    5.252
|    7.719 |    6.856 |    4.119 |    4.434 |        0 |   10.213 |    2.868 |    1.671 |    2.646 |    2.412
|    6.101 |        0 |    2.471 |        0 |    3.796 |    3.679 |    3.475 |    4.366 |     3.63 |    1.283
|    0.929 |    6.523 |    7.792 |    2.813 |    3.257 |    6.204 |        0 |        0 |    0.845 |    2.163
|    1.964 |    2.297 |    4.791 |        0 |        0 |   41.037 |    4.012 |        0 |        0 |    1.581
|        0 |    3.831 |     2.25 |    6.765 |     2.35 |    4.101 |     8.41 |    0.836 |    3.236 |    2.536
|        0 |    4.264 |     3.12 |    5.901 |    2.842 |     11.1 |    6.461 |    4.424 |        0 |    2.387
|      8.2 |    6.718 |     8.73 |    3.052 |    9.815 |        0 |    2.175 |   13.807 |    9.577 |    9.729
|    9.339 |        0 |    5.162 |    7.794 |    7.337 |    3.688 |        0 |        0 |    3.209 |    2.845
|    4.209 |    5.077 |    3.261 |     0.76 |    6.249 |    6.152 |    3.688 |    1.967 |    3.657 |    3.556
|    11.56 |        0 |    5.953 |    7.005 |    9.443 |   13.759 |    8.522 |        0 |        0 |    4.915
|    4.017 |    7.716 |    0.482 |    1.782 |    1.616 |    8.324 |    9.987 |    2.029 |    2.602 |    1.686
|    1.934 |    5.976 |    4.836 |    1.477 |        0 |     9.74 |    6.517 |    9.776 |    4.252 |    2.744
|     3.75 |
```

REST API

- Flask framework
  - Lightweight
  - Relatively easy learning curve

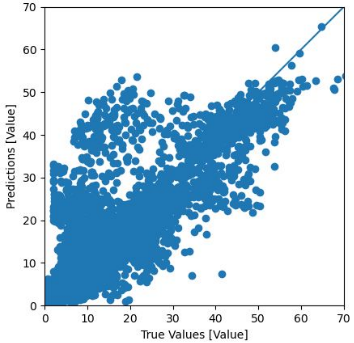
- Handle all data processing tasks
  - Provide access to necessary queries
  - Returns JSON formatted data
  - Implements ML models
  - Update Neo4j with time-series data

ML Design/Implementation

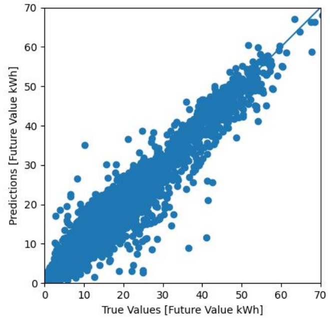
- Two key requirements
  - The ML models must predict the next kWh output for each node in the grid.
  - The ML models must predict the probability of each anomaly class.
- Two types of models
  - One that predicts a continuous value
  - One that classifies a datapoint
- Requires separate algorithms for each type.
- The unique aspects of the three transformer types require their own version of the models

Linear Regression

- Linear Regression will output a continuous value
- Our implementation of Linear Regression
  - Multiple fully connected relu activated layers to add non-linearity
  - MAE loss function over MSE loss in order to limit the impact of outliers
    - Scalable and Generalized models (NFR)
- Feature Set
  - Static Transformer Data (Resistivities, Voltage Rating, etc)
  - Timestamp (insights on power usage by month and hour)
  - Previous Transformer in the line (Locality)
  - Previous and Current hour's data (Power Trend Context)

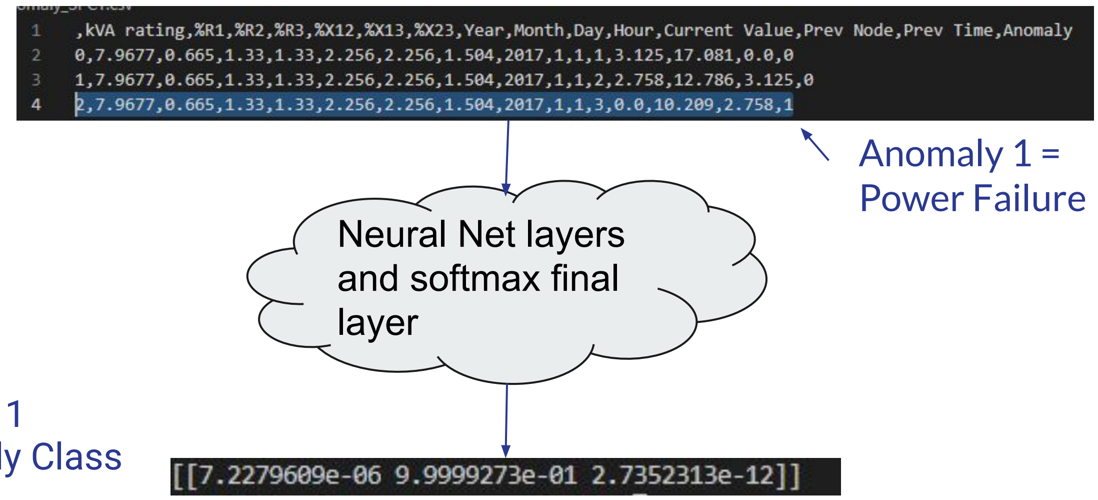


Non-linearity and context features



Logistic Regression

- A continuous value does not tell us if there's an anomaly
- 3 classes of data
  - No Anomaly
  - Power Spike
  - Power Failure
- Softmax for K = 3
  - $$\sigma(\mathbf{z})_i = \frac{e^{-\beta z_i}}{\sum_{j=1}^{K} e^{-\beta z_j}} \text{ for } i = 1, \ldots, K.$$
  - Returns 3 values summing to 1
  - Probabilities for each Anomaly Class
- Can use the same features



[Normal, Power Failure, Power Spike]

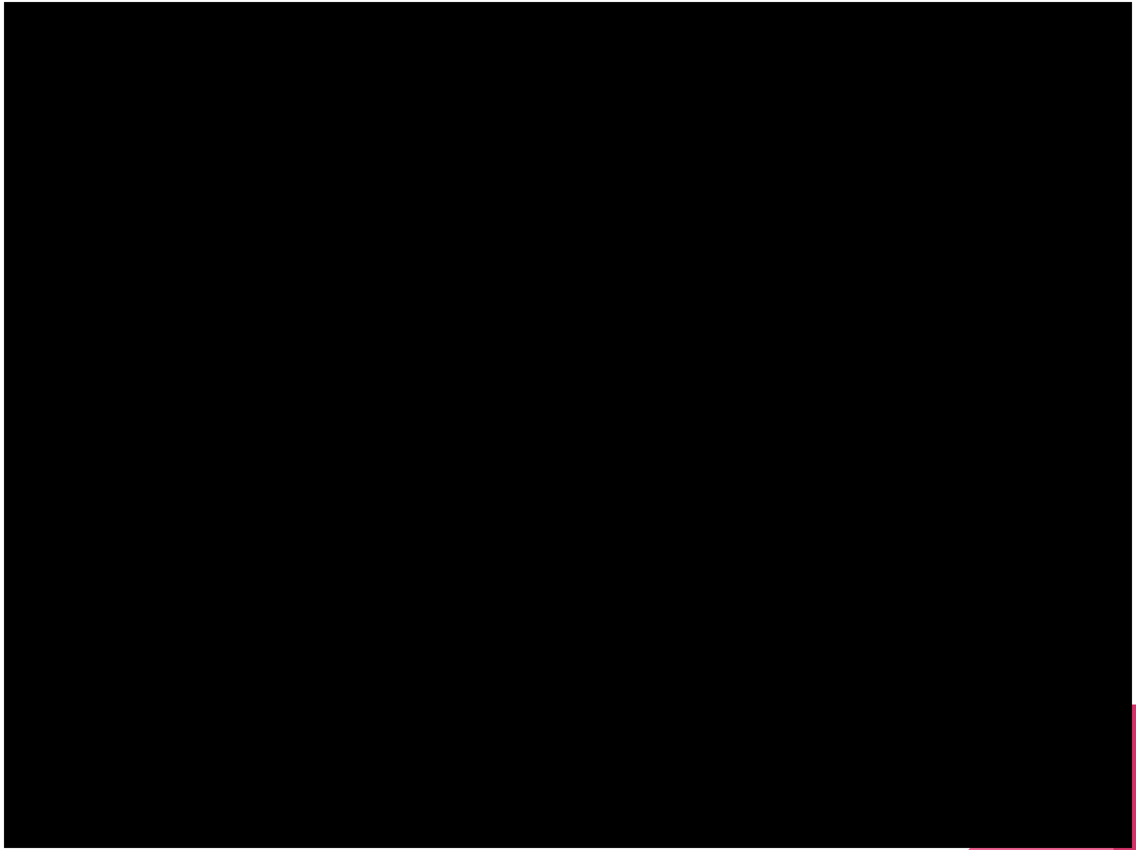
Testing/Testing Results

- ML Models
  - The average deviation of the original data is 6.97 kWh.
  - With the DNN Linear Regression model, this is 1.25 kWh
  - Correctly Classifying 96% of the dataset with the Logistic Models
- Backend
  - Validated database queries manually
  - Verified functionality of endpoints with Postman
  - Identified bottleneck at startup due to database initialization
- Frontend
  - Manually tested every component of the UI
  - Null/undefined errors were common when data was not loaded correctly.
  - Use of asynchronous functions to resolve data loading issues before rendering.

Engineering Standards

- IEEE/ISO/IEC 12207-2017: Software life cycle processes
  - Requirements Definition
  - Architecture Definition
  - Design Definition
  - Implementation
  - Integration

Demo



The GridAI Team

Patrick Wenzel

Karthik Prakash

Justin Merkel

Abir Mojumder

Abhilash Tripathy