Below Ground: Enhancing Electrical Grid Analysis through OpenDSS, Interactive Mapping, and Machine Learning

Chakraborty, Deepankar; Fetahaj, Latif; Jannath, Syeda; Saddler, Orlando[1]

[1]The City College of New York, **CSc 59867, Senior Design II**

Abstract

This paper discusses the creation of an easy-to-use program that simplifies load flow analysis for students and electrical professionals. The project's goal was to make a tool that doesn't rely on private company methods, which are often locked behind confidentiality agreements. Our approach uses machine learning to predict critical issues in power grids, like transmission line failures that could cause blackouts. We focused on a key value known as Voltage Per Unit (VPU) to monitor the health of the grid. Using OpenDSS, a popular electrical system simulation tool, along with Python, we crafted an interactive application that maps out an electrical grid and allows users to change values and see the effects in real-time. This paper walks through the project's development process, the machine learning models we used, the challenges we overcame, and the interactive map we built. We also look at what we achieved compared to what was initially planned and suggest next steps for the project.

Keywords: Load Flow Analysis, OpenDSS, Machine Learning, Power Grid, Electrical Grid Simulation, Neural Networks, Random Forest Algorithm, User Interface, Grid Stability.

Table of Contents:

I. Introduction:

For the Below Ground project, we were instructed by Professor Mohamed to create a program that would make load flow analysis autonomous for the likes of students and professionals alike. The thought process behind a project like this is to make something for people who do not have access to a specific company's approach to load flow analysis. Load flow analysis is used to help engineers and students alike to view changes within bus systems and electrical grids. It should also be noted that load flow analysis is a computation, and every electrical company out there has their own specific approach and formula for achieving the desired results. To look at how a specific company goes about this process students and professors alike usually must sign an NDA or Non-Disclosure Agreement. This information is very sensitive and therefore very hard to get access to, and with this in mind the professor thought a machine learning based model would be of great value to people who cannot get a power company's approval.

In this paper we will be discussing our journey to get to the finished product that we have today. Starting with the project's inception, we'll explore the inspiring goals set by our leaders, Yusef and Kirn, guiding our journey and achievements. From here, we will go into some of the background information that is needed to understand what we are trying to accomplish. After this we will get into our initial approach to achieving the desired results and discuss the difficulties that came along the way. With that we will finally be able to divulge what we were able to do and show how the model performed. Finally, we can discuss what was accomplished compared to what our leaders initially wanted and what will happen with the project in the future.

As stated previously, the thought behind the project is to essentially develop a model that will replicate load flow analysis. More specifically the model is going to be used to try to predict potential disastrous results within a power grid such as the failure of transmission lines which can lead to a blackout. To determine if there is a failure or not we need to focus on a specific value known as the Voltage Per Unit (VPU) or Per Unit Voltage, the terms are interchangeable. Most apparatus within a power grid such as generators, transformers, and lines tend to have a similar VPU. If any of them within a system have a value out of the acceptable range of .95 and 1.05 it is indicative of potential damage to the grid. Along with these initial requirements our project leaders also exclaimed that they wanted the project to be interactive and user friendly. They figured the best way to accomplish this would be to create a map that would show the many components within a grid system based on their real-world coordinates. From the map the user would be able to manipulate the values at specific components and see changes in the map.

II. Project Overview and Interaction with OpenDSS

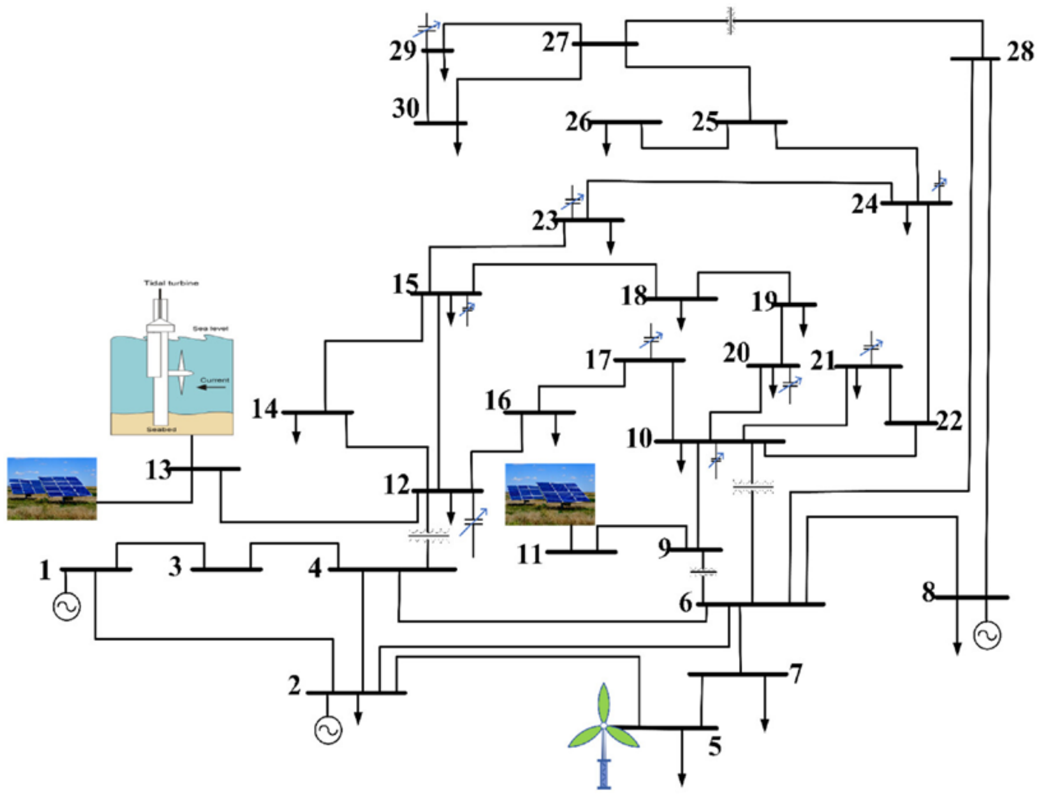


Figure: IEEE 30 bus system

For us to make load flow analysis possible we were instructed to download a program known as OpenDSS. OpenDSS or the Open Distribution System Simulator is a comprehensive electrical system tool for electric utility systems. The program supports nearly all analyses commonly performed for utility distribution systems planning and analysis. For the sake of our project we will feed the program files based on IEEE Bus Systems and the program will do load flow analysis on the bus system. IEEE Bus Systems are test cases used by IEEE to replicate an American Electrical Power System. The bus system contains buses, which are nodes where several transmission lines are connected which may include loads and generators. A load is an electrical device that consumes electrical energy and converts it into another form. Of the several bus cases that are included within the files that come with OpenDSS the one of great interest to our project leaders is the IEEE 9500 bus example. Throughout the project any code we write will be tested on cases with fewer buses but will ultimately be written with the IEEE 9500 bus in mind.

II. B: Interaction with OpenDSS

OpenDSS is a tool primarily used by Electrical Engineering students and professionals, so for us Computer Science students we needed to find a way to interact with the program without directly using it. This would be done in python with the use of the import win32com.client, this enables us to create an instance of the OpenDSS COM object (dssObj) which will allow subsequent interaction with the OpenDSS engine. The OpenDSS engine allows interaction with the OpenDSS software from within a python script. The object allows subsequent interaction with the OpenDSS engine and access to its methods and properties. From here we will create three objects that allow us to access the different properties of OpenDSS that we need for the sake of this project. The first object is dssText, this provides methods and properties for working with text-based commands in OpenDSS. The second object is dssCircuit, which allows interaction with the active circuit loaded into the python script. This will be the primary way we manipulate values within the program. The third object is dssSolution, which provides methods and properties for solving power flows and accessing solution-related information. This object will need to be used after every function created that manipulates the circuit in any way. Any python program that will interact with OpenDSS will require a file path that represents the file the IEEE Bus System we want to manipulate and objects that allow us to use the many features of OpenDSS.

To see changes in the VPU value that we spoke of earlier we are going to need to manipulate the circuit. The primary way we can go about seeing changes is by manipulating the real and reactive power of each load within the bus system. This can be done by manipulating the properties dssCircuit.Loads.kw and dssCircuit.Loads.kvar. It was suggested by our project leaders that we should be able to manipulate these values by a percentage to see how subsequent increases and decreases could destabilize the whole circuit. Our project leaders also wanted to see how changing the source of power for a particular load could change the VPU, with this they decided they wanted the ability to change any given load that they see on the map into a PVSystem. A PVSystem is a photovoltaic system or solar panel-based system. In theory, based on the environment in which the bus system is located and the specifications for the solar panel set by the user we should be able to see real changes in the VPU and therefore the map.

III. Development of the Map and GUI Application

To abstract the entire process and make it easier for the user to make changes on the grid, such as installing a PVSystem, or changing the load on a bus, we needed to build an interactive map. The goal was to provide the ability to click on each plotted load and change the values of real(kW) and reactive(kvar) power. Moreover, the map would also allow multiple buses, select certain areas, or filter by components such as transformers. However, we ran into problems when we tried to achieve this level of map interactivity.

A. Challenges in Map Development and Learning Curve

Initially, we researched ArcGIS and ArcPy. ArcGIS is a geographic information system (GIS) software that is best suited for mapping, spatial analysis (patterns and relationships in geographic data), and data management. On the other hand, ArcPy is a Python package that helps python developers to integrate and automate workflows in ArcGIS. Even though these are great tools, we faced sign-in issues with our school accounts, and pricing wasn't suited for our budget. Moreover, the browser option of ArcGIS didn't fulfill the map interactivity goals.

In the second try, we turned to Python libraries, such as GeoPandas, Plotly, and Folium. GeoPandas is an open-source Python library that extends the popular data manipulation library, Pandas, to provide support with spatial or geospatial data, in analysis, manipulation, and visualization of the data. On the other hand, Plotly, is a plotting library used for creating interactive and visually appealing plots in maps, which lets users explore data through zooming, panning, and hovering. Lastly, Folium is a library designed for creating leaflet maps that support markers, pop-ups, and other visualizations. Together these libraries are very powerful in creating and representing useful information on the map, but still didn't offer what we needed, dynamic data updates and direct user edits on map points.

An approach that would solve our needs was to create a web-based GUI. Creating a full-stack application using HTML, JavaScript, Python, and Django/Flask, would enable full interactivity for the map to click on points, edit buses, or select different regions on the map. However, this would require a lot of developing time which we didn't have . Therefore, we had to look for an intermediary solution.

B. Implementation of GUI Using PyQt

Given the tight timeline of the project, we decided to use Python only, as the programming language. Some key benefits of this approach were:

a) **Consistency in the codebase:** which leads to better collaboration and maintainability.

b) **Reduced learning curve:** as a team, Python is our strength.

c) **Better integration:** When using a single language, libraries and frameworks are more likely to work seamlessly together.

d) **Improved collaboration:** Meetings, code reviews, and knowledge sharing was more efficient.

To build the intermediary solution, we used Folium for basic map interactions, and PyQt for GUI. PyQt is a library built for the Qt application framework which is used to develop applications with a graphical user interface. Some of the Qt functionalities that we used for developing the map were buttons, text boxes, sliders, resizable layout and responsive user interfaces, etc.

C. Interactive Map and GUI Features

The full functional map is shown in the picture below.

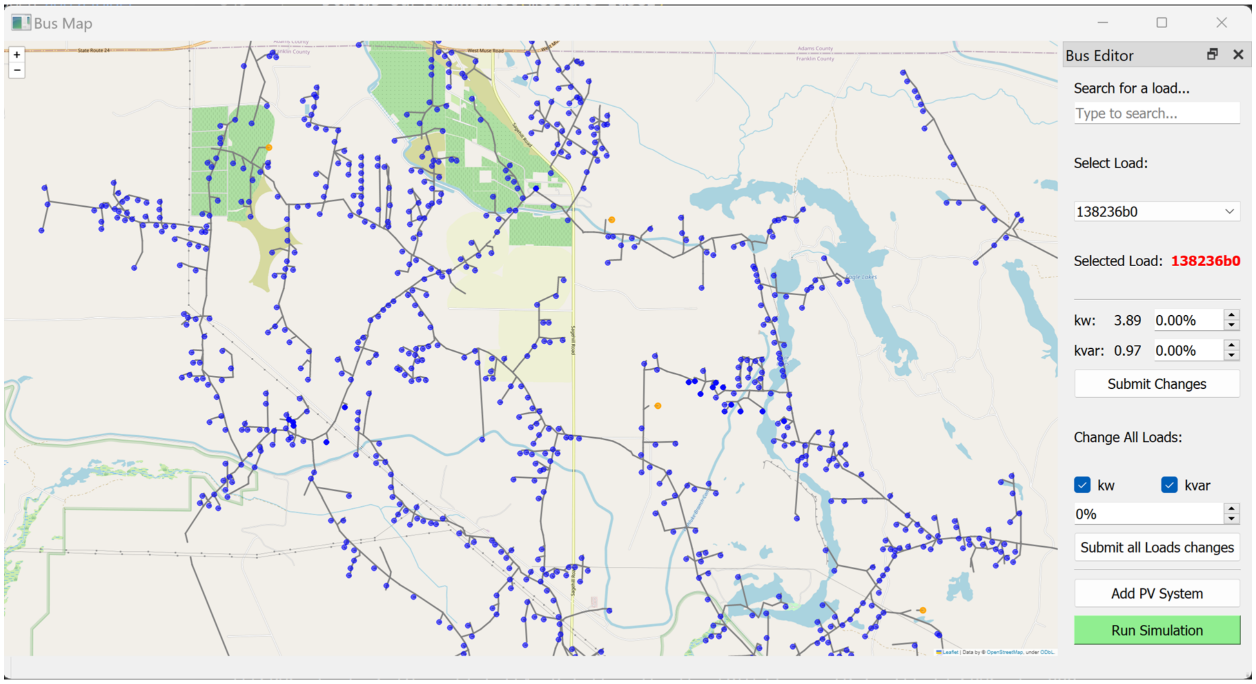
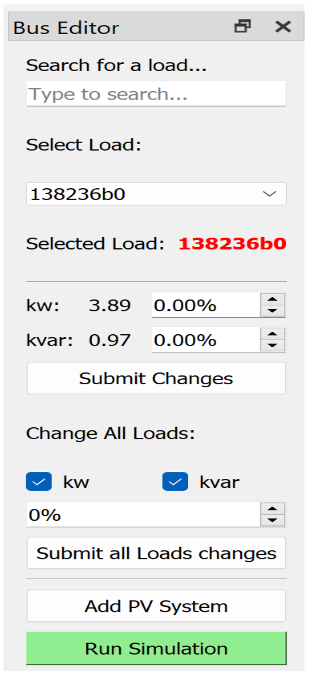


Fig1. Map of the IEEE 9500 system with its functionalities.

On the left side of the map, is the IEEE 9500 bus system with all its transmission lines, buses, loads, and generators. On the right side is the Bus Editor panel. In the map, users can slide, zoom-in, zoom-out, click loads and see information such as PUV, kvar and kW values. Fig2 shows the bus editor panel(left) with its functionalities(right).



Bus Editor

Search for a load...

Type to search...

Select Load:

138236b0

Selected Load: 138236b0

kw:	3.89	0.00%
kvar:	0.97	0.00%

Submit Changes

Change All Loads:

✓ kw ✓ kvar

0%

Submit all Loads changes

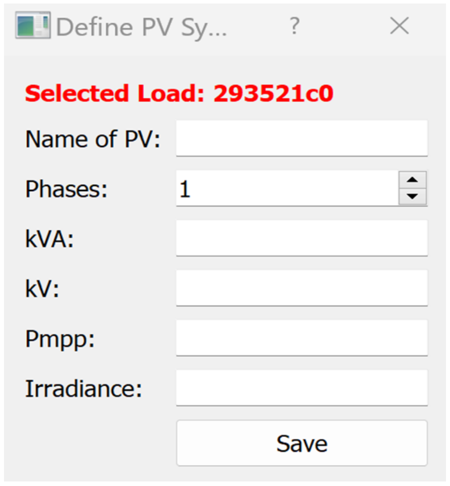
Add PV System

Run Simulation

- **Search:** search for a load.
- **Select:** or select a load from the dropdown menu.
- **Change selected load:** change the value of kW and/or kvar by a percentage, then they can choose to submit changes, and run the simulation.
- **Change all loads:** change all the loads of the system by a percentage, then submit the changes.
- **Add a PV System:** more on this in the next picture (Fig3).
- **Run the Simulation:** run the load flow model, uses OpenDSS on the backend.

Fig2. Bus Editor Panel(right) with its functionalities (left).

On a selected load, users are able to add a PV system, and then see how it impacts the grid. Fig3 shown below.



Define PV Sy...

Selected Load: 293521c0

Name of PV:	
Phases:	1
kVA:	
kV:	
Pmpp:	
Irradiance:	

Save

Fig3. PV System

Main solar panel features are:

1. **Phase:** Refers to the electrical phase of the power generated by the solar panel. It indicates whether the electricity produced is in sync with the alternating current (AC) power grid and is typically specified as single-phase or three-phase.
2. **kVa (Kilovolt-amperes):** Represents the apparent power of the solar panel system. It is the product of the voltage and current in the system, expressed in kilovolt-amperes. It measures the total power, including both real (active) power and reactive power.
3. **kV (Kilovolts):** Denotes the voltage output of the solar panel system. It represents the electrical potential difference and is typically specified in kilovolts. Voltage is a crucial parameter in determining the efficiency and performance of the electrical components in the system.
4. **pmpp (Maximum Power Point):** Refers to the maximum power output that the solar panel can deliver under standard test conditions (STC). It is the point on the current-voltage (I-V) curve where the product of current and voltage is maximized, indicating the optimal operating point for maximum power generation.
5. **Irradiance:** Represents the solar energy received per unit area on the solar panel surface. It is usually measured in kilowatt-hours per square meter (kWh/m^2) and is a critical factor influencing the efficiency of solar panels. Higher irradiance levels generally result in increased power generation.

The IEEE bus system represented on the map consists of transmission lines, loads, and generators. These elements are represented by color markers (Fig4). The used colors are:

1. Gray - transmission lines

2. Blue - loads
3. Orange - generators

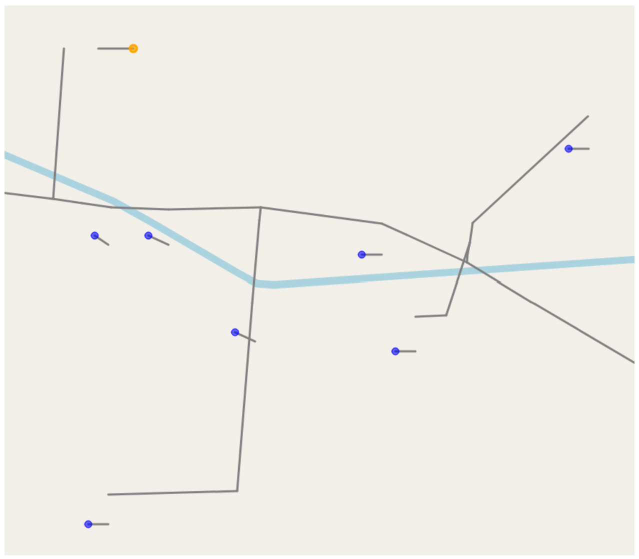


Fig4. Color markers used to present IEEE bus system elements.

After some changes to the bus system, the user has the option to run the OpenDSS load flow simulation on the backend. Then, the map will change, depending on the changes made. For example, any bus that has a PU voltage within the acceptable range will be circled in green, otherwise if not within the range, it will be circled in red. See Fig5.

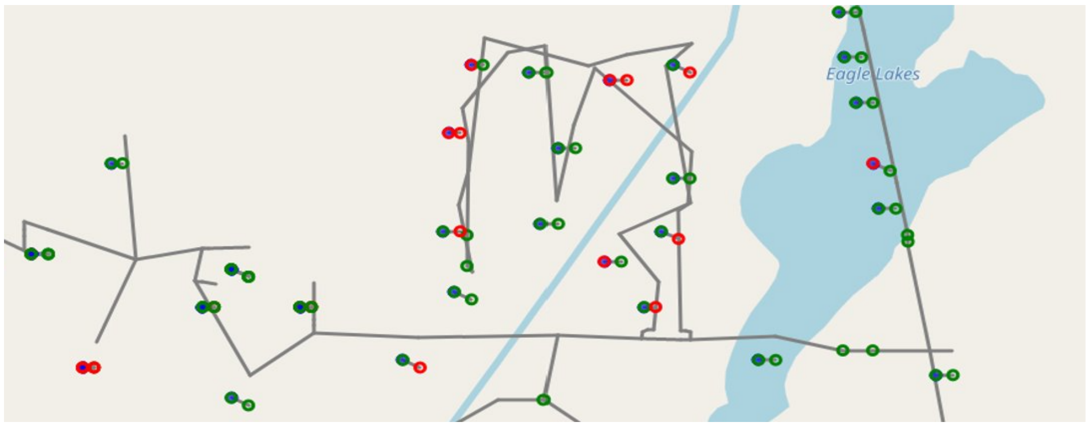


Fig5. Map changes that reflect changes on the system based on PU voltage range.

Lastly, the user has the ability to filter changes and compare how the map looked before and after changes on the bus system. The filtering button is placed on the left side of the bus panel editor. See Fig6.

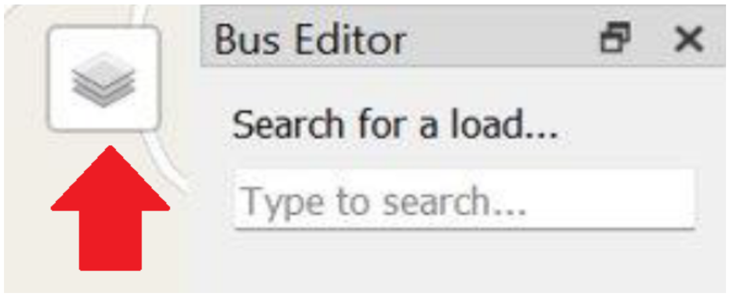


Fig6. The filtering button.

IV. Implementation of Machine Learning

A. Introduction to Machine Learning in Electrical Grid Analysis

The transformative role of machine learning in electrical grid analysis is quite impressive, especially when it comes to load flow analysis. This section of the project explores the relevance of machine learning in predicting and maintaining grid stability, a critical aspect of modern power systems. Machine learning's capability to process large datasets and identify patterns makes it an invaluable tool in predicting outcomes in electrical grids, where accurate predictions are essential for avoiding system failures.

B. Previous Research:

Innovations in machine learning for power system analysis have shown significant promise, particularly with neural network (NN) models for power flow prediction. The study "Neural Network-based Power Flow Model" by Pham and Li (2021) stands out in this topic. Their work demonstrates a neural network's ability to surpass the accuracy of traditional DC Power Flow models in predicting grid conditions, especially in systems integrating renewable energy. Employing a multi-layered neural architecture with advanced activation functions, their model effectively addresses the unpredictability of power systems with renewable sources.

The research by Pham and Li is important for our understanding of neural network applications in power systems. Their model's superior performance in testing across various system sizes, as assessed by Mean Square Error and Mean Absolute Error metrics, confirms the potential of NNs in enhancing grid stability and efficiency. Crucially, the model's effectiveness is highlighted by its low prediction error rates, with the Percentage Error in voltage magnitude across various test systems ranging narrowly from 0.54% to 0.60%, and particularly the Percentage Error for active power flow predictions remaining under 0.5% for the IEEE 39-bus system. This contribution guides our research towards leveraging machine learning for robust, real-time power system analysis and predictive operations, extending their applicability to more dynamic and uncertain grid environments.

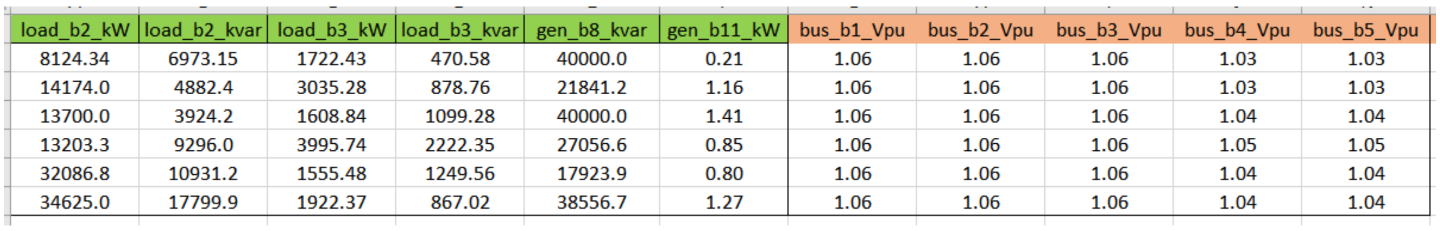
C. Data Overview: Inputs & Outputs

At the heart of our machine learning application lies a comprehensive set of input and output data. For creating the dataset, we primarily focused on the IEEE 30 bus system for our model so that we can evaluate its performance and scale it for other systems. The input data comprises bus names, which serve as unique identifiers for each bus in the electrical grid, along with detailed information on loads, generators, and PV systems. These elements are crucial for balancing the grid. We focus on voltage measurements (kW and kVAR) derived from running 2500 simulation cases. For output data, the post-run values for each bus, particularly the Per Unit

(PU) value, are of utmost importance. Maintaining these values within the optimal threshold of 0.95 to 1.05 is crucial for grid stability.

D. Generating Training Data

The generation of training data is one of the most important processes in this project. We utilized a Python script along with OpenDSS Python COM interface specifically designed for electrical power system simulation and data collection. This script is tasked with running 2500 simulations, where it modifies load and generator parameters within defined ranges, collects the resultant data, and then resets the parameters to their original state. This method enables the accumulation of a comprehensive dataset, which is vital for machine learning applications such as predictive modeling and anomaly detection in power systems.



load_b2_kW	load_b2_kvar	load_b3_kW	load_b3_kvar	gen_b8_kvar	gen_b11_kW	bus_b1_Vpu	bus_b2_Vpu	bus_b3_Vpu	bus_b4_Vpu	bus_b5_Vpu
8124.34	6973.15	1722.43	470.58	40000.0	0.21	1.06	1.06	1.06	1.03	1.03
14174.0	4882.4	3035.28	878.76	21841.2	1.16	1.06	1.06	1.06	1.03	1.03
13700.0	3924.2	1608.84	1099.28	40000.0	1.41	1.06	1.06	1.06	1.04	1.04
13203.3	9296.0	3995.74	2222.35	27056.6	0.85	1.06	1.06	1.06	1.05	1.05
32086.8	10931.2	1555.48	1249.56	17923.9	0.80	1.06	1.06	1.06	1.04	1.04
34625.0	17799.9	1922.37	867.02	38556.7	1.27	1.06	1.06	1.06	1.04	1.04

Figure: Table showing a subset of columns of training data for IEEE 30 bus system. The green columns are input features and the red columns are output labels.

E. Machine Learning Algorithms

1. Random Forest Algorithm:

Our initial focus was on the Random Forest algorithm, chosen for its robustness and capability to handle complex and nonlinear data relationships. As an ensemble learning method that constructs multiple decision trees during training, Random Forest outputs either the mode of the classes or the mean prediction of these trees, making it highly effective for predicting load flow and grid stability under various conditions. Unlike other models prone to overfitting, Random Forest is less prone to overfitting. It counteracts this by averaging across multiple deep decision trees, each trained on different parts of the same dataset. This approach not only ensures a balance between prediction accuracy and model simplicity but also provides a more generalized solution, enhancing performance on unseen data.

2. Neural Network:

We also used a Neural Network model alongside the Random Forest algorithm as it's really good at finding complex patterns in data. Neural networks are like the human brain, with layers of nodes or neurons that learn from input data. Our model is built with Keras, a tool for creating neural networks, and it's set up in a straightforward, layer-by-layer way. It starts with a layer of 64 neurons using the ReLU function, which helps in training by managing non-linear data. Then there's another layer with 32 neurons, also using ReLU, to further learn from the data's complex patterns.

The last part of our model uses a linear function, different from ReLU, to match our output with the number of things we're predicting, like bus voltages in the grid. This way, the model gives us direct numerical insights into the grid's condition. We also use the Adam optimizer, which is great for deep learning because it adjusts how it learns from each piece of data. And for measuring how well our model is doing, we use the mean squared error method. This calculates how far off our model's predictions are from the actual values, helping us make sure our model is accurate.

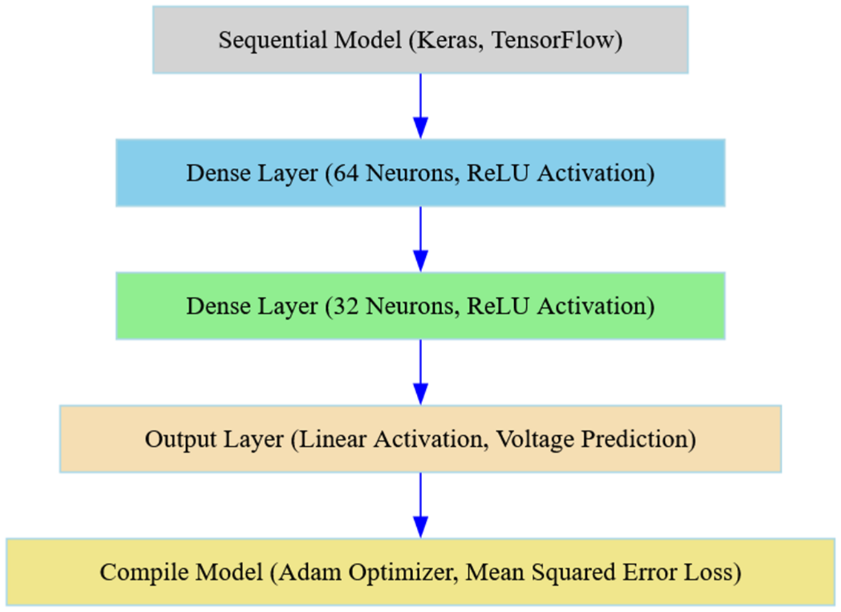

Sequential Model (Keras, TensorFlow)
Dense Layer (64 Neurons, ReLU Activation)
Dense Layer (32 Neurons, ReLU Activation)
Output Layer (Linear Activation, Voltage Prediction)
Compile Model (Adam Optimizer, Mean Squared Error Loss)

Figure: Showing Neural NEtwork Architecture

E. Optimization of Machine Learning Algorithms

The initial models underwent significant optimization. The Random Forest algorithm, in particular, was fine-tuned through a grid search process, substantially improving its performance. This optimization was critical in reducing error rates and enhancing the model's reliability.

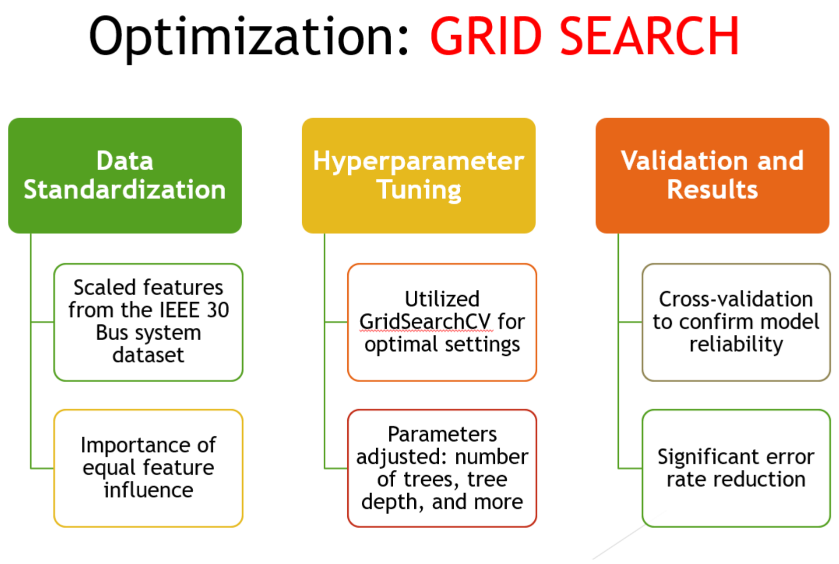

Optimization: GRID SEARCH
Data Standardization
Scaled features from the IEEE 30 Bus system dataset
Importance of equal feature influence
Hyperparameter Tuning
Utilized GridSearchCV for optimal settings
Parameters adjusted: number of trees, tree depth, and more
Validation and Results
Cross-validation to confirm model reliability
Significant error rate reduction

Figure: Grid Search optimization method

F. Architecture and Integration with GUI

Illustrating the architecture of both Random Forest and Neural Network models provides insights into their structure and data flow. A key highlight of this project is the integration of these models with the GUI, allowing real-time data manipulation and prediction updates. The GUI displays these predictions, enabling users to simulate various scenarios and see immediate visual feedback.

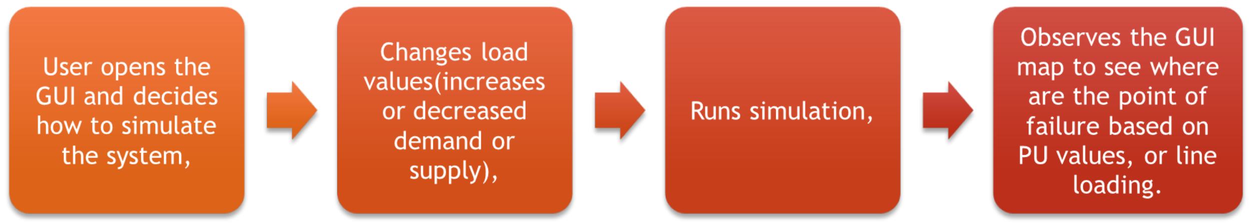

User opens the GUI and decides how to simulate the system,
Changes load values(increases or decreased demand or supply),
Runs simulation,
Observes the GUI map to see where are the point of failure based on PU values, or line loading.

Figure: Project workflow under GUI integration

V. Results and Experiences

A. Results:

Random Forest Model: The graph provided illustrates a comparative analysis between the expected Per Unit (PU) values derived using OpenDSS and the predicted values obtained from the Random Forest model. At first glance, the predictions appear to closely follow the trend of the expected values, suggesting a degree of correlation between the model's outputs and the actual data. However, it is important to acknowledge that despite the visual proximity of the two lines, the percent error in predictions is relatively high.

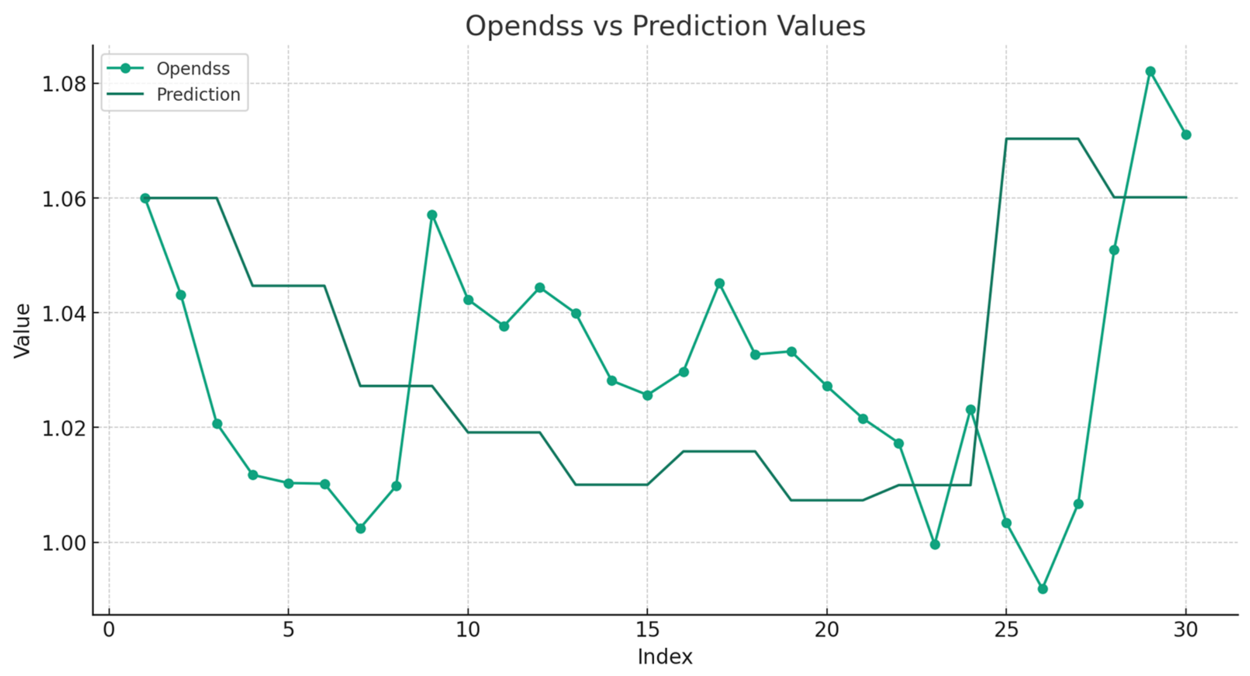


Figure: PU values using Random Forest model vs actual value.

Comparing the Error %: The error percentage of the Random Forest model for bus voltage prediction shows considerable variability, ranging from 0% for the second bus to a peak of nearly 20% for bus 27. This wide range indicates inconsistencies in the model's prediction capabilities. For critical applications like electrical grid monitoring, where precision is vital, such variability could be problematic. The high error rate, especially the 20% on bus 27, suggests that the model may not be entirely reliable for certain buses, which could lead to significant consequences if used for real-world decision-making. Further refinement and understanding of the model's limitations are necessary to improve its reliability across all buses.

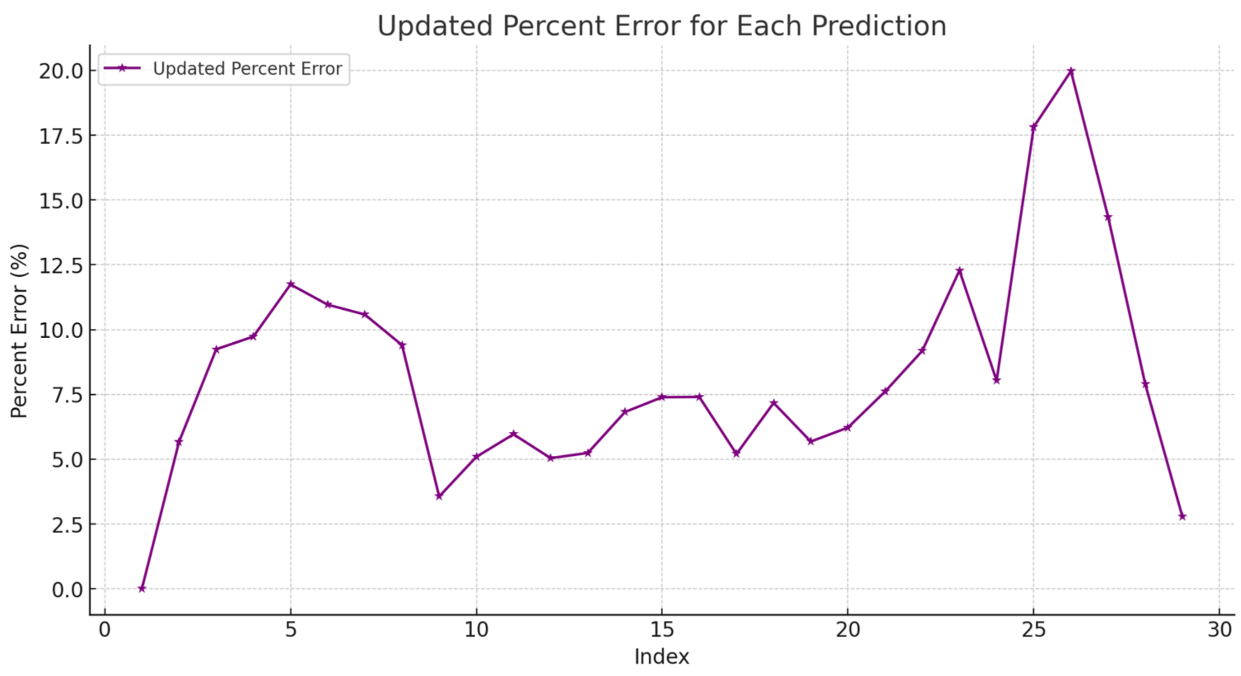


Figure: Graph showing % error of the random forest model on IEEE 30 bus system

Comparing average error for all 3 models:

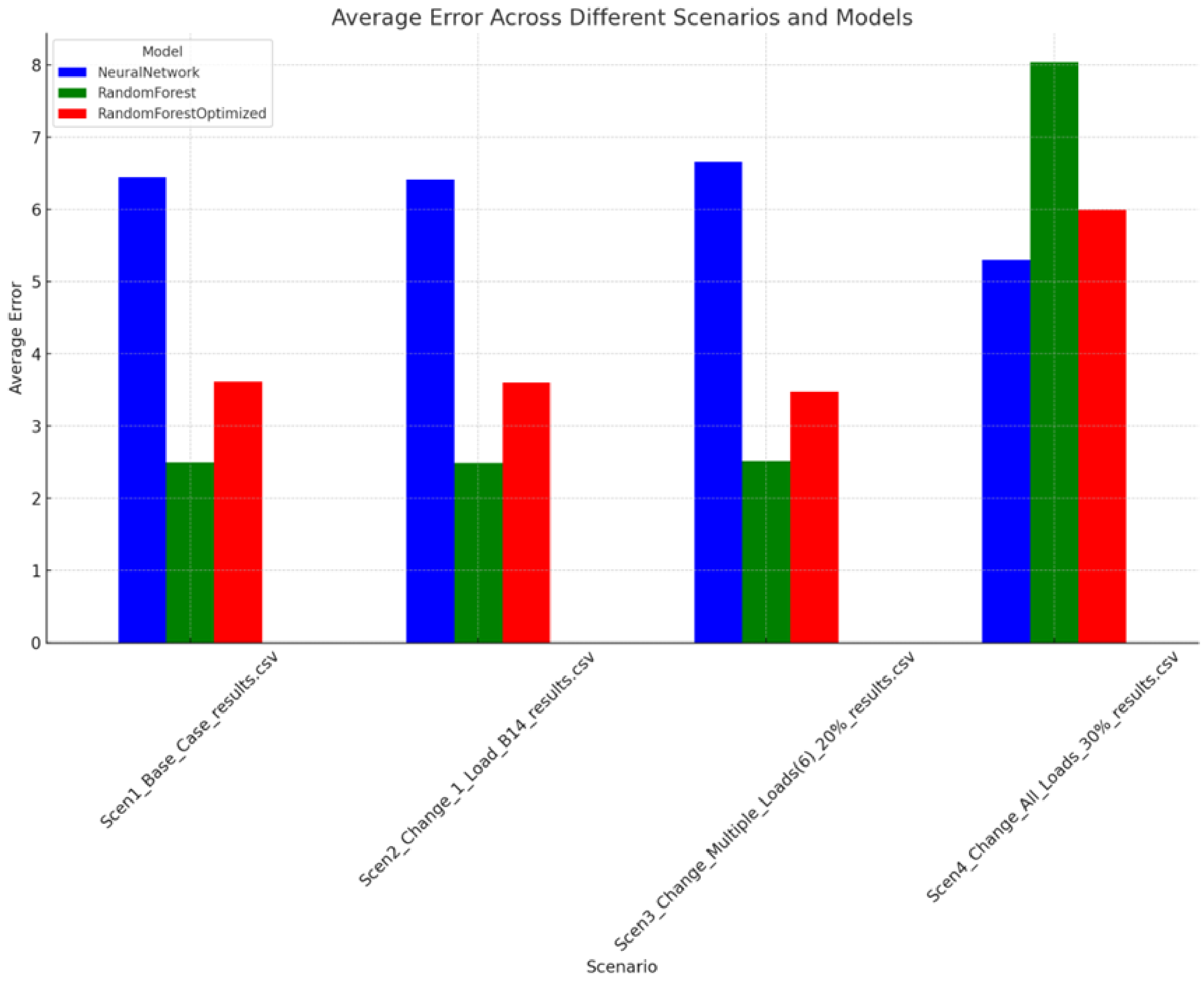


Figure: Bar chart showing average error % across 3 models under 4 scenarios.

The bar chart provided presents an analysis of average error percentages across different scenarios using three distinct models: Random Forest, Optimized Random Forest, and Neural Network. The data exhibits a lack of consistent correlation in performance among the models across varied scenarios.

In the base case scenario, the general Random Forest model surprisingly performs well, with error percentages remaining modest. However, when changes scale up to a 30% alteration in all loads, the error spikes to as high as 8%, indicating a less reliable performance under more variable conditions.

On the other hand, the Neural Network model shows a robust performance, particularly in scenarios with significant changes, such as the one with a 30% shift in all parameters. This suggests that the Neural Network model may have a higher ability to adapt to large-scale fluctuations within the system.

The Optimized Random Forest, while showing improvements over the general model, still displays variable performance, excelling in some scenarios while falling short in others. The objective was to reduce the error percentage to a target range between 1% and 2%. However, due to time constraints, there was a clear need for further research and model fine-tuning to lower the error rate. This variability underscores the challenges in creating a 'one-size-fits-all' model for electrical grid analysis, highlighting the crucial need for continued tuning of the model and its validation under a variety of conditions to increase its accuracy and reliability.

B. Project Experiences and challenges

Engaging in a collaborative project provided us with invaluable insights into the dynamics of working with project leaders and navigating the challenges of meeting deadlines.

Throughout this experience, we gleaned essential lessons about effective communication, adaptability, and managing expectations.

Working alongside project leaders allowed us to comprehend the complexities of leadership styles and the importance of aligning our efforts with their vision. Understanding their goals and expectations became paramount, forming the foundation for successful collaboration. This interaction not only developed our technical skills but also enhanced our ability to decipher and respond to the nuanced needs of project leaders.

A significant aspect of our learning involved engaging with the unpredictable nature of project development. We navigated the dynamic nature of project development, learning to adapt to changing circumstances. When encountering challenging requests, we found the value in open communication about possible limitations. This approach helped us maintain a positive working relationship and foster mutual understanding.

Moreover, we understood the art of pivoting- an essential skill when faced with roadblocks. Rather than dwelling on the impossibility of a task, we embraced the concept of presenting alternatives that closely aligned with the project’s original version. This not only showcased our problem-solving abilities but also demonstrated our commitment to finding viable solutions. It reinforced the idea that flexibility and adaptability are crucial components of successful project execution.

Our experience working with a team of different backgrounds provided us with a comprehensive understanding of the intricacies involved in collaborative projects. We learned that challenges are inevitable, but effective communication, honesty about limitations, ability to pivot toward feasible solutions, regular updates, progress reports, discussions about challenges ensure that everyone involved is on the same page are key to overcoming obstacles. These learnings go beyond this project, offering valuable skills for a wide range of professional contexts.

VI. Conclusion

Summary of Key Points

- OpenDDS, a detailed and thorough flow analysis was conducted on the provided bus system. This utilization of OpenDDS showcased its advanced capabilities in scrutinizing system dynamics.
- The project involved the development of a specialized model designed specifically for replicating load flow analysis, showcasing precision and accuracy.
- The primary objective was to anticipate potential disastrous outcomes within a power grid, signifying a forward-thinking strategy for addressing challenges proactively.
- Our focus was placed on identifying vulnerabilities related to transmission line failures. This includes a detailed exploration of factors such as load distribution, network topology, and potential cascading effects, all with the overarching goal of preventing blackout scenarios.

Future Direction

Generating a comprehensive and detailed documentation is crucial for the ongoing project, given the possibility of future expansions, the addition features, and the potential transformation into a full-stack application to enhance interactivity. This documentation aims to serve as a comprehensive guide for developers, stakeholders, and any individuals involved in the project's evolution.

In terms of future editions, potential enhancements are proposed to augment the project's capabilities. This involves a detailed outline of the steps required for seamless integration into the existing codebase, emphasizing considerations for scalability and maintainability to ensure the project's long-term viability. Secondly, envisioning a full-stack transformation, a comprehensive roadmap is provided, outlining the necessary architectural changes, database integration strategies, and server-side considerations. Notably, security measures to safeguard the integrity of the transformation process are also articulated. Lastly, focusing on the machine learning aspect, the existing model is thoroughly documented, clarifying its structure and algorithm.

Proposed enhancements are offered with the aim of enhancing model accuracy. Furthermore, specific data requirements for model training are highlighted, along with suggestions for potential datasets to optimize overall performance. This detailed roadmap for future development emphasizes a comprehensive and strategic approach to evolving the project in a sustainable and effective manner.

Project Link(GitHub):

https://github.com/deepankarck2/Below-Ground-Code-FIles

References:

1) Pham, T., &amp; Li, X. (2022). Neural network-based Power Flow Model. 2022 IEEE Green Technologies Conference (GreenTech). https://doi.org/10.1109/greentech52845.2022.9772026