Smart Grid Stability Prediction using Deep Learning

A Project Report

Submitted to

Dr. Premalatha L

Submitted by

Souvik Datta 19BEE1213
Ratnadeep Das Choudhury 19BEE1125

EEE – 4013 Smart Grid

Winter Semester 2021-22

Contents

Abstract

Consumer demand information is collected in a smart grid, centrally analyzed against current supply conditions, and the resulting proposed price information is provided back to users for them to decide on usage. Because the entire process is time-dependent, dynamically evaluating grid stability becomes not merely a problem, but a need. Decentralized Smart Grid Control (DSGC) systems monitor the frequency of the grid. As a result, it links the power price to the grid frequency, making it available to all participants, i.e., all energy consumers and producers. In the DSGC system, we investigate optimized deep learning (DL) models for solving fixed inputs (variables of equations) and equality difficulties. As a result, measuring the grid frequency at each customer's premise would be sufficient to provide the network administrator with all necessary information about the present network power balance, allowing it to price its energy offering and alert consumers accordingly.

Introduction

As we all know that the rise of renewable energy in the power generation is inevitable. In a traditional grid system the power flow is unidirectional, which cannot be implemented in Smart Grids with renewable energy sources integrated with it. Thus, the power flow is multidirectional in a Smart Grid System.

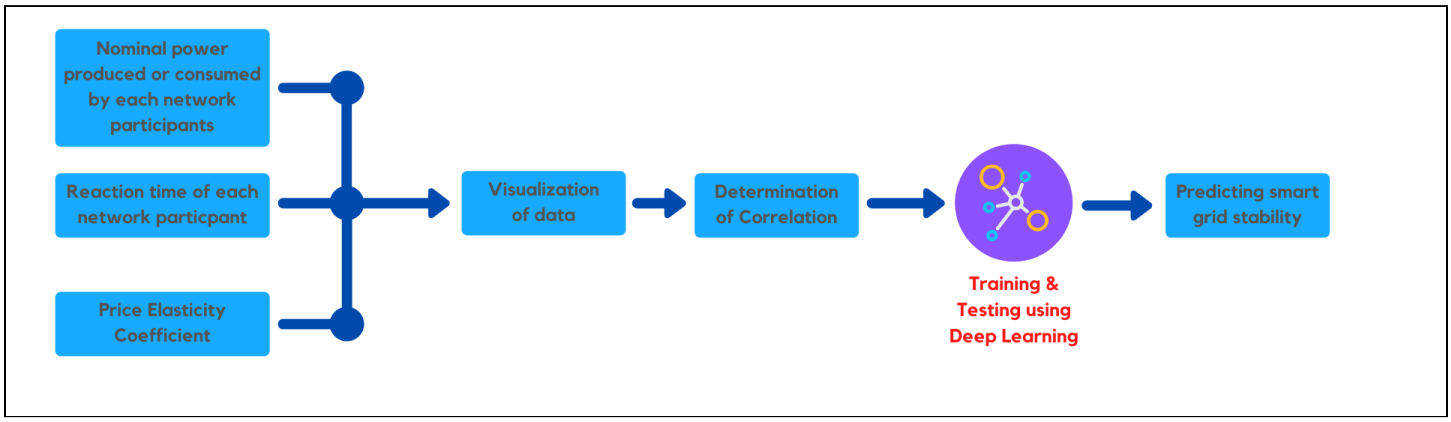
The multidirectional power flow is also due to the fact that in the Smart Grid system the consumers have also become ‘prosumers’, which is a term given to those consumers who use

electricity as well as produce it. This results in more complex generation, distribution and consumption.

In Smart Grid, the consumer demand information is collected, centrally evaluated against current supply conditions and resulting proposed price information is sent back to customers, as a result creating a process which is time dependent and dynamic. Thus, in such a dynamic process the estimation of grid stability becomes a vital requirement.

To Solve this problem, we use Decentral Smart Grid Control (DSGC). DGSC monitors one particular property of the grid - its frequency. It ties the frequency with the electricity price so that it is available to all participants. In times of excess generation frequency of the power grid increases and in times of under-production frequency of the power grid decreases. Thus, measuring the grid frequency at the premise of each customer will provide all required information about the current network power balance and will be enough for finding energy prices and informing the consumers accordingly. The DSGC differential equation-based mathematical model descriand aims at identifying grid instability for a reference four-node star architecture, comprising one power source supplying energy to three consumption nodes.

Block Diagram



In the input section, we take three inputs specifically. First, the Nominal power is produced or consumed by each network participant which is represented by ‘p1’ to ‘p4’ in the dataset. Second, the reaction time of each network participant is represented by ‘tau1’ to ‘tau4’ in the dataset and third, the price elasticity coefficient is represented by ‘g1’ to ‘g4’ in the dataset.

After collection of these inputs as data, we visualize it using histograms and scatterplots. It is followed by determining the correlation and training & testing the data set using our deep learning algorithm. So, after fitting the model to the training set, it is time to extract predictions for the testing set and segregate those above the 'threshold' of 0.5 ('unstable' cases below the

threshold, 'stable' cases above it). As a result, we can find the predicted stability and accuracy of the confusion matrix.

Description of your project

Dataset - University of California Irvine Dataset [Link]

	tau1	tau2	tau3	tau4	p1	p2	p3	p4	g1	g2	g3	g4	stab	stabf
56145	5.350829	3.627837	3.317839	5.090104	4.336520	-1.642137	-1.747661	-0.946721	0.407798	0.999336	0.915428	0.763454	0.092042	0
31472	9.398630	3.739337	4.094607	3.668571	3.103415	-0.653784	-0.668451	-1.781180	0.735296	0.928269	0.351576	0.529518	0.032826	0
12506	2.903025	8.752341	3.881151	2.762012	4.398481	-0.831853	-1.831631	-1.734996	0.845339	0.508219	0.267080	0.181972	-0.021253	1
40074	2.936039	9.265647	0.637753	3.849926	3.460467	-0.551507	-1.383955	-1.525005	0.745967	0.737205	0.081326	0.903324	0.011321	0
5828	2.057330	0.616727	1.966893	4.350094	4.246153	-1.001041	-1.861587	-1.383526	0.849111	0.075089	0.645225	0.812591	0.001459	0

The dataset consists of results taken from grid stability simulations done on a 4-node network with star topology (see Fig. 2), as described in and has a synthetic nature. It has 12 primary predictive features and two dependent variables. It has 10.000 samples. Since the grid used in simulations is symmetric we can augment the original dataset 3! (6) times. The augmented dataset consists of 60.000 samples.

Predictive features:

1. ‘tau1’ to ‘tau4’: the reaction time of each network participant, a real value within the range 0.5–10 (’tau1’ corresponds to the supplier node, ’tau2’ to ’tau4’ to the consumer nodes);

2. ‘p1’ to ‘p4’: nominal power produced (positive) or consumed (negative) by each network participant, a real value within the range − 2.0 to − 0.5 for consumers (’p2’ to ’p4’). As the total power consumed equals the total power generated, p1 (supplier node) = − (p2 + p3 + p4)

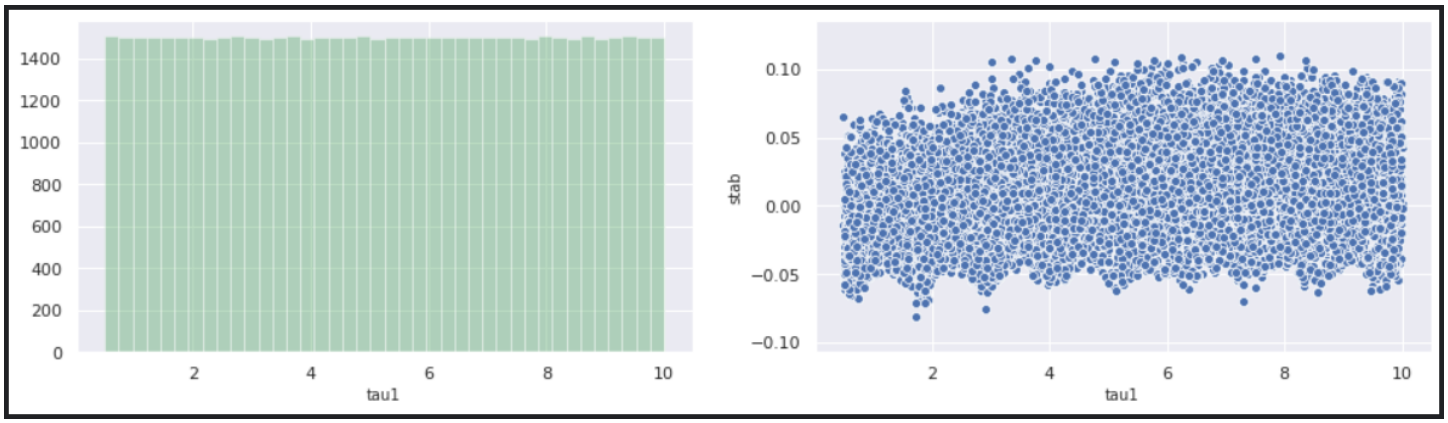
3. ‘g1’ to ‘g4’: price elasticity coefficient for each network participant, a real value within the range 0.05 to 1.00 (’g1’ corresponds to the supplier node, ’ g2’ to ’ g4’ to the consumer nodes; ’ g’ stands for ’gamma’);

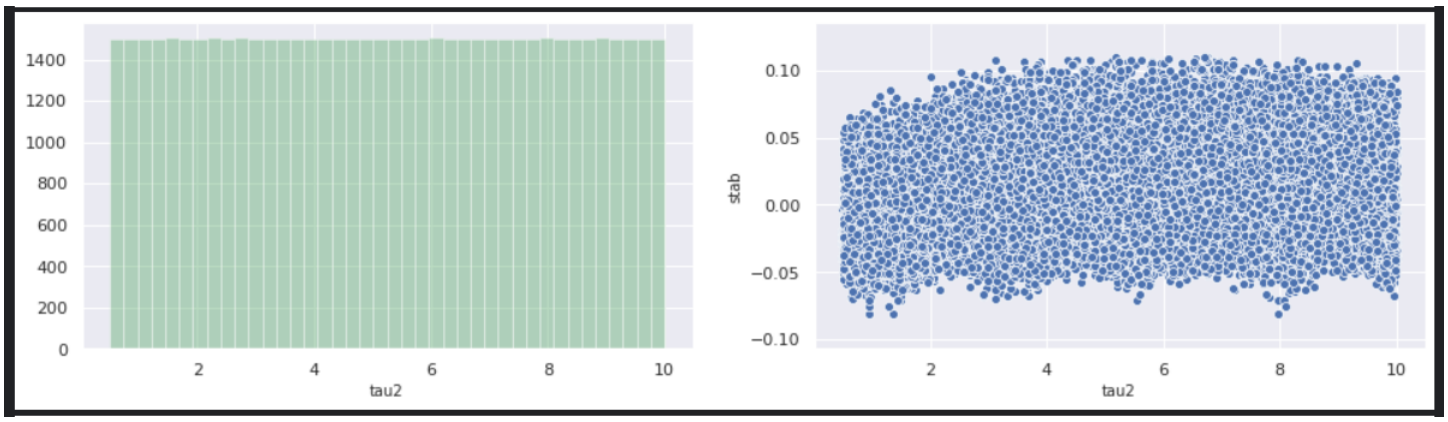
Dependent variables:

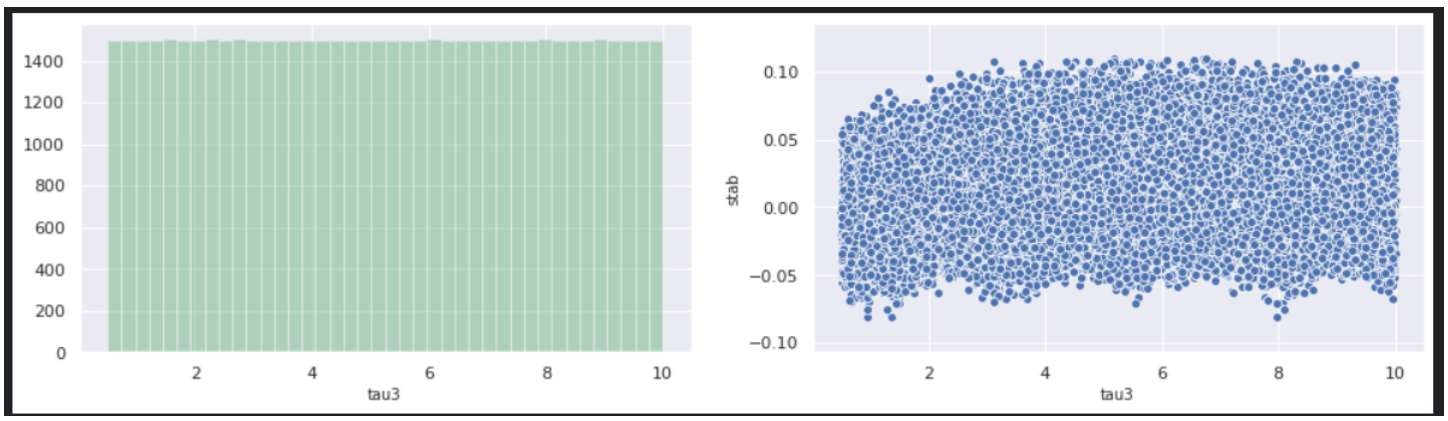
1. – ’stab’: the maximum real part of the characteristic differential equation root (if positive, the system is linearly unstable; if negative, linearly stable);

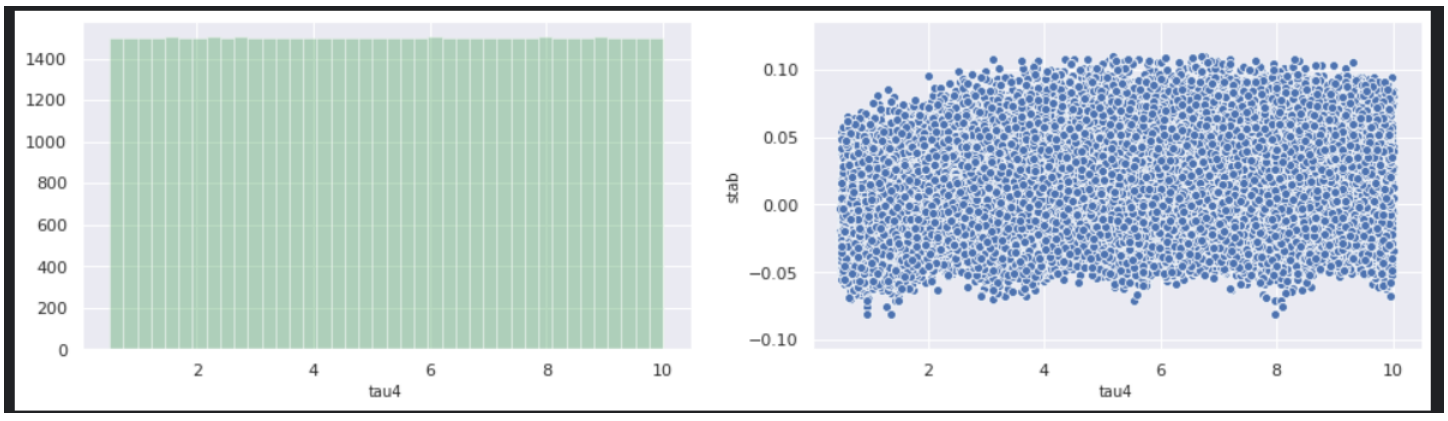
2. – ’stabf’: a categorical (binary) label (’stable’ or ’unstable’).

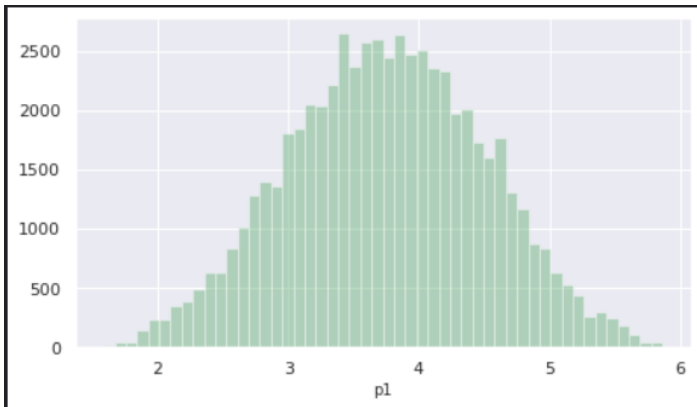
Visualization of the entire Dataset -

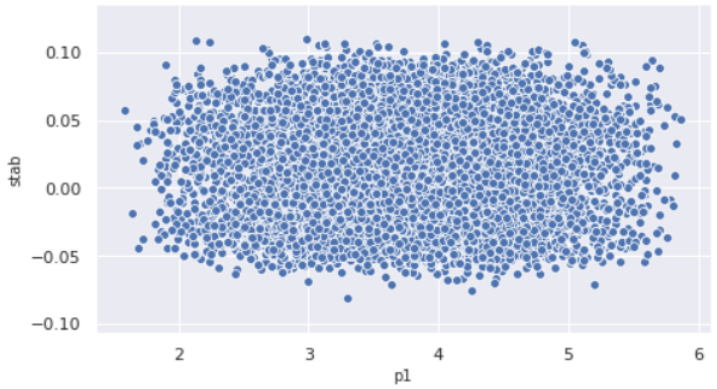

1400
1200
1000
800
600
400
200
0
2
4
6
8
10
tau1
0.10
0.05
0.00
−0.05
−0.10
stab
2
4
6
8
10
tau1

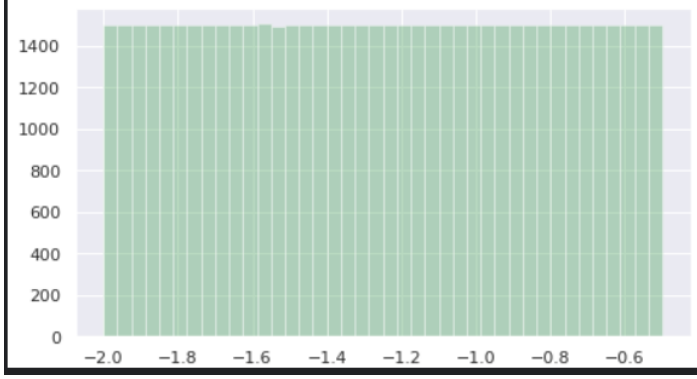

1400
1200
1000
800
600
400
200
0
2
4
6
8
10
tau2
0.10
0.05
0.00
−0.05
−0.10
stab
2
4
6
8
10
tau2

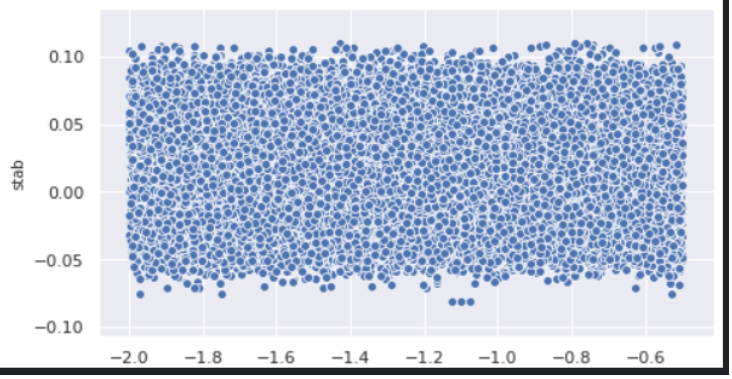

1400
1200
1000
800
600
400
200
0
2
4
6
8
10
tau3
0.10
0.05
0.00
−0.05
−0.10
stab
2
4
6
8
10
tau3

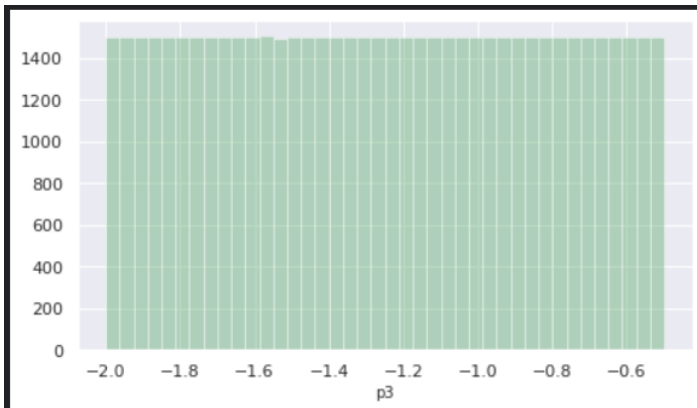

1400
1200
1000
800
600
400
200
0
2
4
6
8
10
tau4
0.10
0.05
0.00
−0.05
−0.10
stab
2
4
6
8
10
tau4

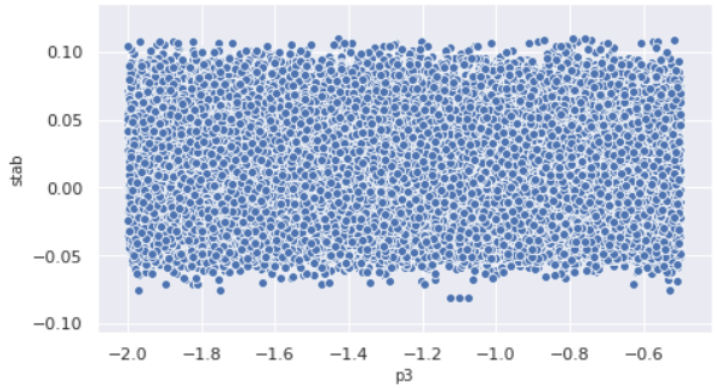

2500
2000
1500
1000
500
0
2
3
4
5
6
p1

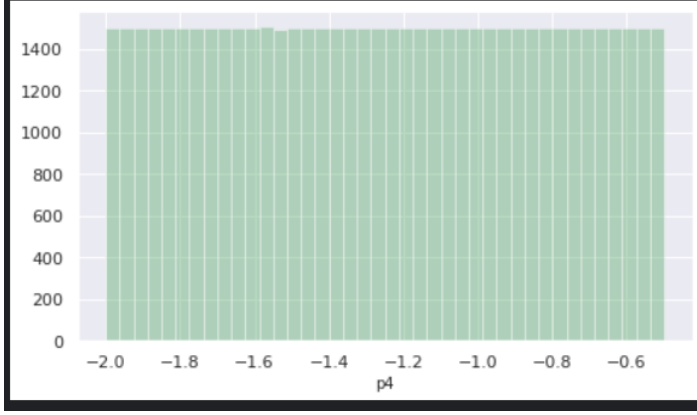

0.10
0.05
0.00
−0.05
−0.10
stab
2
3
4
5
6
p1

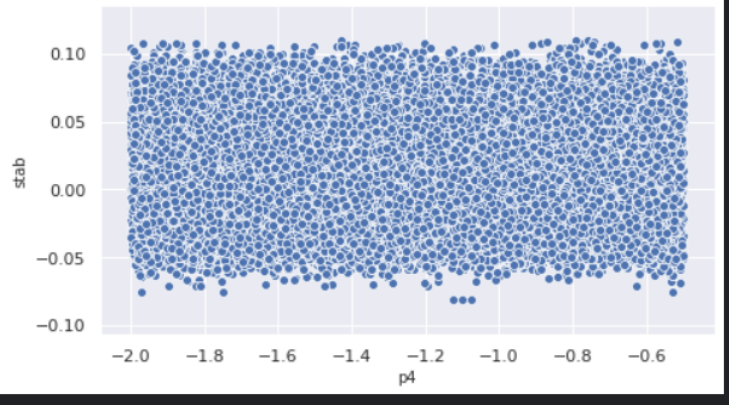

1400
1200
1000
800
600
400
200
0
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6

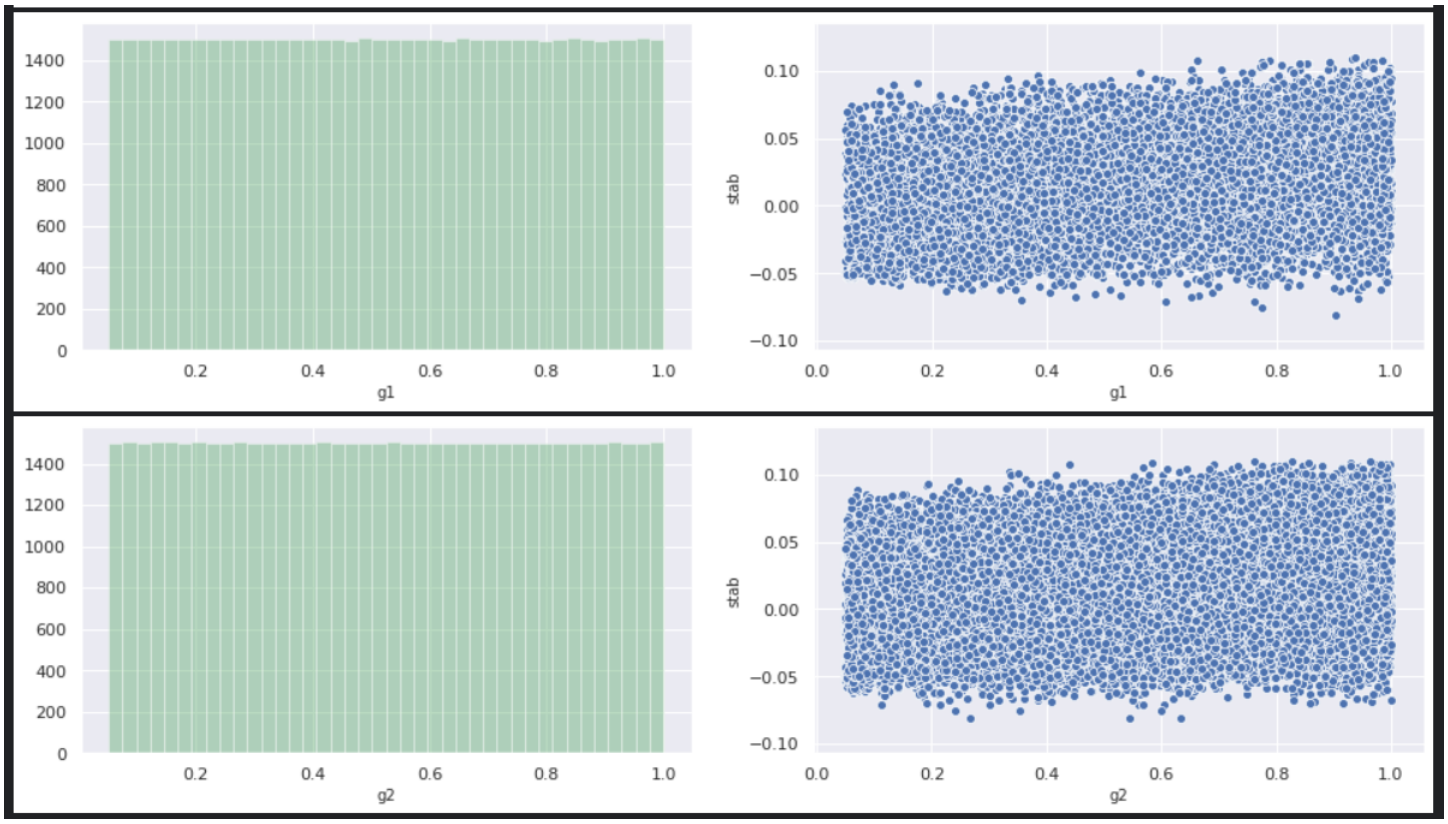

0.10
0.05
0.00
−0.05
−0.10
stab
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6

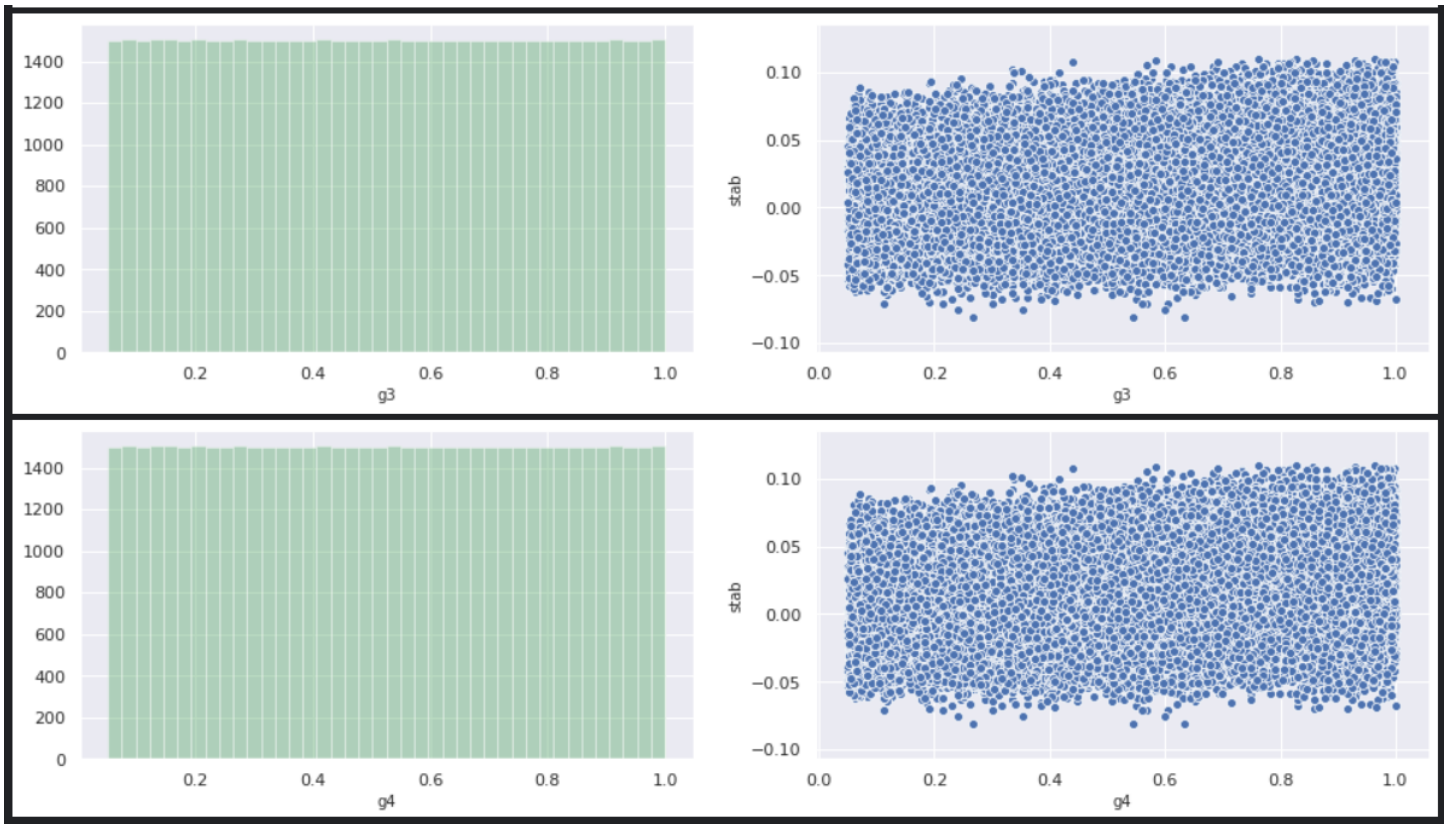

1400
1200
1000
800
600
400
200
0
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6
p3

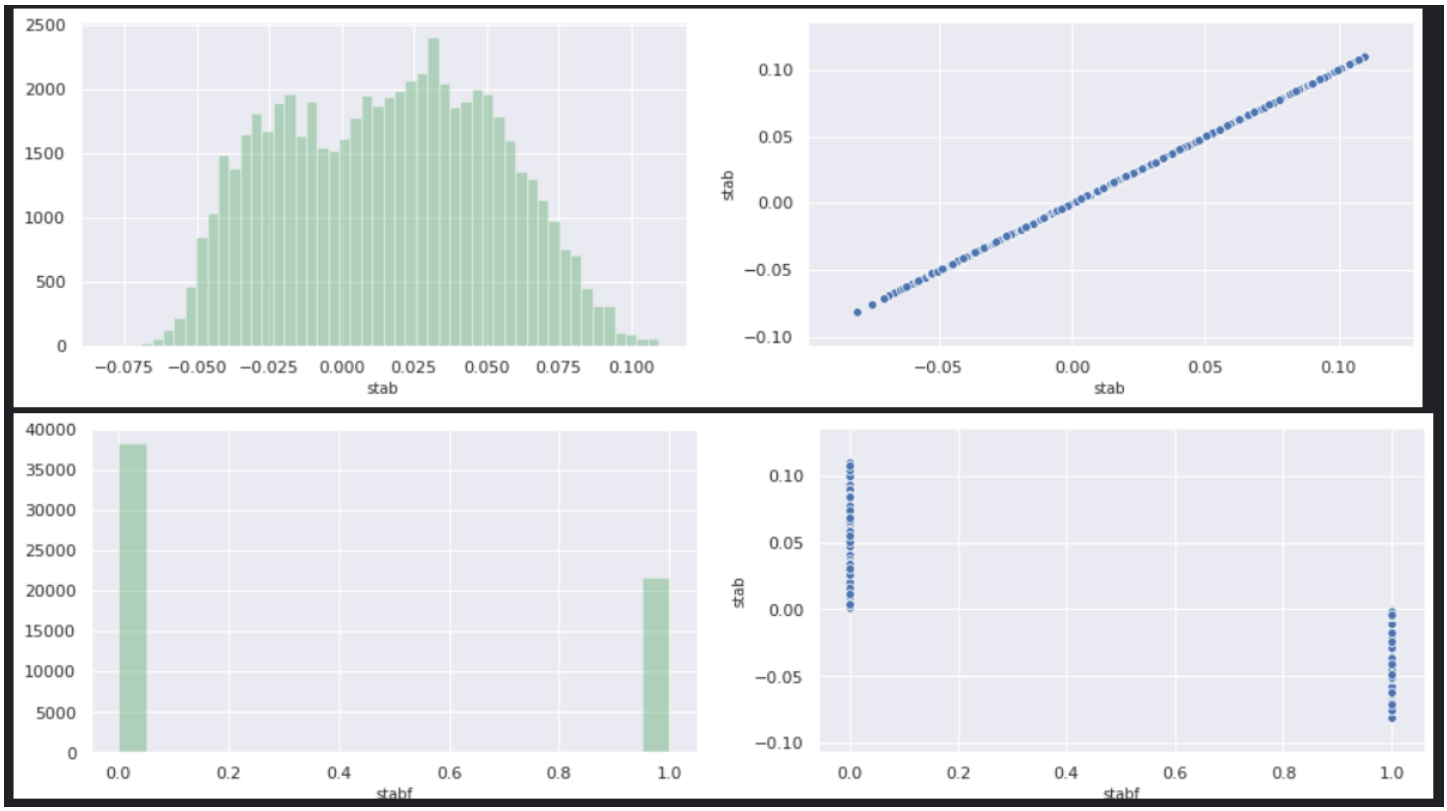

0.10
0.05
0.00
−0.05
−0.10
stab
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6
p3


1400
1200
1000
800
600
400
200
0
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6
p4

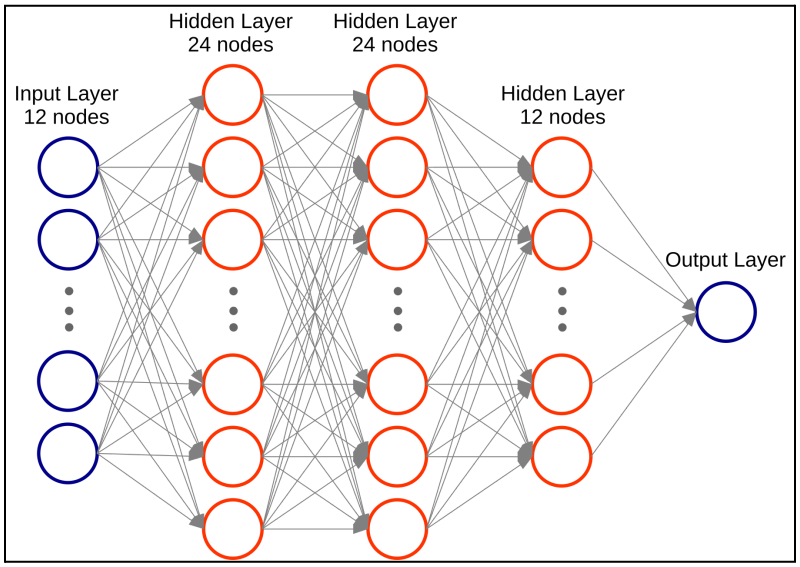

0.10
0.05
0.00
−0.05
−0.10
stab
−2.0
−1.8
−1.6
−1.4
−1.2
−1.0
−0.8
−0.6
p4


1400
1200
1000
800
600
400
200
0
0.2
0.4
0.6
0.8
1.0
g1
stab
0.10
0.05
0.00
−0.05
−0.10
0.0
g1
g2
g2


1400
1200
1000
800
600
400
200
0
0.2
0.4
0.6
0.8
1.0
g3
stab
0.10
0.05
0.00
−0.05
−0.10
0.0
g3
g4
g4



Artificial Neural Network -



- Since we have features that are numerical real numbers within a range, the choice of 'relu' as the activation function for hidden layers seems straightforward.
- Similarly, as this is a logistic classification problem, where the output is binary ('0' for 'unstable', '1' for 'stable') the choice of 'sigmoid' as activation for the output layers seemed appropriate.

- Compilation with 'adam' as optimizer and 'binary_crossentropy' as the loss function follow the same logic. The fitting performance will be assessed using 'accuracy' as the metric of choice.

```
# ANN initialization
classifier = Sequential()

# Input layer and first hidden layer
classifier.add(Dense(units = 24, kernel_initializer = 'uniform',
activation = 'relu', input_dim = 12))

# Second hidden layer
classifier.add(Dense(units = 24, kernel_initializer = 'uniform',
activation = 'relu'))

# Third hidden layer
classifier.add(Dense(units = 12, kernel_initializer = 'uniform',
activation = 'relu'))

# Single-node output layer
classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation
= 'sigmoid'))

# ANN compilation
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy',
metrics = ['accuracy'])
```

Confusion Matrix -

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm. The confusion matrix consists of four basic characteristics (numbers) that are used to define the measurement metrics of the classifier. These four numbers are

1. TP (True Positive): TP represents the number of patients who have been properly classified to have malignant nodes, meaning they have the disease.
2. TN (True Negative): TN represents the number of correctly classified patients who are healthy.
3. FP (False Positive): FP represents the number of misclassified patients with the disease but actually they are healthy. FP is also known as a Type I error.
4. FN (False Negative): FN represents the number of patients misclassified as healthy but actually they are suffering from the disease. FN is also known as a Type II error.

```
cm = pd.DataFrame(data=confusion_matrix(y_testing, y_pred,
labels=[0, 1]),index=["Actual Unstable", "Actual Stable"],
columns=["Predicted Unstable", "Predicted Stable"])
```

	Predicted Unstable	Predicted Stable
Actual Unstable	3707	116
Actual Stable	78	2099

Results

Deep learning proved to be an outstanding prediction tool for this particular application. Even considering that the dataset is well behaved and needed no significant preprocessing, the high accuracies obtained on the testing set confirm that a deep learning model may be safely considered. It would though be up to a smart grid operator to confirm if the accuracy level obtained with deep learning would suffice in practical terms (live network);

Original dataset (10,000 observations)					
Architecture	Folds	Epochs	Confusion Matrix		Accuracy
24-12-1	10	10	596	28	93.20%
			40	336	
24-12-1	10	20	605	19	95.00%
			31	345	
24-12-1	10	50	603	21	94.40%
			35	341	
24-24-12-1	10	10	604	20	95.00%
			30	346	
24-24-12-1	10	20	604	20	94.90%
			31	345	
24-24-12-1	10	50	602	22	95.80%
			20	356	

As expected, more complex ANN architectures performed better than simpler ones;

An increased number of epochs considered during fitting also plays a major role. It is evident that the more the model is exposed to the training set, the better the prediction accuracy;

From a machine learning exercise perspective, the use of an augmented dataset with 6,000 observations contributed significantly to better results;

It must be noted that input parameters utilized in the original DSGC simulations fall within predetermined ranges. As a follow-up step in the validation of this learning machine, it would be interesting to assess its performance using a new test set with observations obtained from simulations with input parameter values residing in other alternative ranges.

Augmented dataset (60,000 observations)					
Architecture	Folds	Epochs	Confusion Matrix		Accuracy
24-12-1	10	10	3795	56	96.27%
			168	1981	
24-12-1	10	20	3780	71	97.50%
			79	2070	
24-12-1	10	50	3788	63	97.93%
			61	2088	
24-24-12-1	10	10	3778	73	97.20%
			95	2054	
24-24-12-1	10	20	3763	88	97.58%
			57	2092	
24-24-12-1	10	50	3797	54	97.98%
			67	2082	

Model Training and Accuracy -

```
Model evaluation

5400/5400 [==============================] - 0s 49us/step
Round 1 - Loss: 0.1956 | Accuracy: 91.37 %
5400/5400 [==============================] - 0s 40us/step
Round 2 - Loss: 0.1411 | Accuracy: 93.69 %
5400/5400 [==============================] - 0s 54us/step
Round 3 - Loss: 0.1169 | Accuracy: 95.09 %
5400/5400 [==============================] - 0s 38us/step
Round 4 - Loss: 0.1103 | Accuracy: 95.31 %
5400/5400 [==============================] - 0s 39us/step
Round 5 - Loss: 0.0864 | Accuracy: 96.33 %
5400/5400 [==============================] - 0s 40us/step
Round 6 - Loss: 0.0868 | Accuracy: 96.39 %
5400/5400 [==============================] - 0s 40us/step
Round 7 - Loss: 0.0779 | Accuracy: 96.70 %
5400/5400 [==============================] - 0s 40us/step
Round 8 - Loss: 0.0910 | Accuracy: 96.20 %
5400/5400 [==============================] - 0s 38us/step
Round 9 - Loss: 0.0695 | Accuracy: 97.07 %
5400/5400 [==============================] - 0s 40us/step
Round 10 - Loss: 0.0762 | Accuracy: 96.70 %
```

```
print(f'Accuracy per the confusion matrix: {((cm.iloc[0, 0] + cm.iloc[1, 1]) / len(y_testing) * 100):.2f}%')
Accuracy per the confusion matrix: 96.77%
```

Conclusion

In this project, we tried to predict the stability of Smart Grid using critical parameters like Nominal power is produced or consumed by each network participant, reaction time of each network participant and price elasticity coefficient using a deep learning algorithm. We also used histograms and scatterplots for better visualization of the input data. For training the deep learning model, we used the dataset of the University of California Irvine. After training the model and testing it we successfully obtained the predicted stability and accuracy of the confusion matrix.

References

1. Breviglieri, P., Erdem, T., & Eken, S. (2021). Predicting Smart Grid Stability with Optimized Deep Models. SN Computer Science, 2(2). doi:10.1007/s42979-021-00463-5

2. Azad, S., Sabrina, F., & Wasimi, S. (2019, November). Transformation of smart grid using machine learning. In 2019 29th Australasian Universities Power Engineering Conference (AUPEC) (pp. 1-6). IEEE.

3. Omitaomu, O. A., & Niu, H. (2021). Artificial intelligence techniques in smart grid: A survey. Smart Cities, 4(2), 548-568.