A tree based classifier for transient stability prediction following island splitting

Erlend Sandø Kiel
Sigurd Hofsmo Jakobsen
Eirik Haugen
Sondre Duna Lundemo
Department of Energy Systems
SINTEF Energy Research
Trondheim, Norway

Signe Riemer-Sørensen
Filippo Remonato
Department of Mathematics and Cybernetics
SINTEF Digital
Trondheim, Norway

***Abstract*—An unexpected failure or outage of one or multiple system components can cause a new operational situation that requires remedial actions. An important remedial action to model correctly is islanding. Finding the transient stability of an island is computationally heavy, and it may be necessary with a trade-off between speed and accuracy in the classification of island stability. This is especially the case if one has to perform a large number of simulations.**

In this paper, a decision tree based ensemble method is used to predict the stability of islands in the power system during a contingency event. A comparison study shows that the trained model can contribute with a large reduction in time spent on the transient stability assessment, while being substantially more accurate than a static power flow simulation.

***Index Terms*—machine learning, power system stability, power system dynamics, reliability**

I. Introduction

The power system has traditionally been planned and operated according to the N-1 criterion. This principle states that the system should be able to withstand, at all times, a credible contingency, i.e. an unexpected failure or outage of a system component, in such a way that the system is capable of accommodating the new operational situation without violating operational security limits [1]. In the recent European FP7 project GARPUR it was argued that a move towards probabilistic reliability criteria aiming at minimizing the socio-economic cost of power system operation and planning is desirable [2]. According to transmission system operators (TSOs) one of the barriers preventing usage of probabilistic criteria is that the fact that remedial actions may fail is insufficiently accounted for [3]. However, considering such failures will substantially increase the computational time needed to perform the analyses necessary for planning and operating the system.

For some networks, the most important remedial action to model correctly is islanding [4]. However, analysing whether or not an island is stable is more computationally expensive than performing a power flow analysis. Different stability problems can occur during an islanding event. For instance if there is a significant imbalance between produced and consumed power, there may be under or over frequency resulting in tripping of generation or shedding of load. Another problem that may occur is that of transient stability, which is the problem we will treat in this paper. Transient stability may be analysed using time domain simulation or the extended equal area criterion [5], [6]. The challenge of long computational times when analysing large-scale power systems is discussed in [7] and it is suggested to make use of the potential that machine learning offers to increase the computational speed in an security constrained optimal power flow (SCOPF). Decision trees have also been proposed for determining transient stability in power systems [8], [9]. A good review of papers including transient stability as a constraint in SCOPF is included in [10], which uses a statistical method for including transient stability.

The aim of this work is not to include transient stability as a constraint in a SCOPF. It is rather to predict if an island is stable or not for the purpose of a contingency analysis. Preferably, the prediction should be faster than running a computationally expensive simulation, and more accurate than running a standard power flow. The findings will also be useful for developing prediction models for other purposes as well, although our focus is on contingency analyses.

This paper investigates if a decision tree ensemble based classifier, `XGBoost` [11], can be used instead of a time domain simulation to increase the accuracy of the analysis, while keeping the simulation time low. Although the topic is similar to that of [8], significant improvements have occurred within the field of machine learning since then, and our approach is focused towards the special case of island splitting.

To demonstrate our findings we present a comparison study which investigates the difference and similarities of using time-domain simulation, power flow analysis and machine learning assisted power flow for contingency analysis in the case of island splitting. The comparison study is based on the test grid described in [12]. Some preliminary theory is presented

The research leading to these results has received funding from the Research Council of Norway through the project "Resilient and Probabilistic reliability management of the transmission grid" (RaPid) (Grant No. 294754), The Norwegian Water Resources and Energy Directorate, and Statnett.

in Section II. Our method is presented in Section III, the results in Section IV, and the conclusions in Section V.

II. Theory

A. Machine learning

Classification is a key problem in machine learning. The goal is essentially to predict a discrete random variable Y from another random variable X. More specifically, consider the data set

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) | \mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^m, y_i \in \mathscr{L}\}.$$

where $i \in \{0, \dots, n\}$, when the data set contains n samples. The goal is to learn functions $\phi : \mathcal{X} \to \mathscr{L}$, known as *classification rules* or *classifiers*, that map their inputs onto a discrete set of labels, each associated to a certain class. When observing a new x, y is predicted to be $\phi(x)$. We restrict our attention in the following to *binary* classification; where the set of labels is $\mathscr{L} = \{0, 1\}$, corresponding to an unstable and stable power system respectively. A regression tree is a type of decision tree where the function with output associated to an instance $\mathbf{x}$ is given by travelling from the root node of the regression tree to a leaf, choosing (i.e. deciding) the successor child at each node of the tree on the basis of a test done on one of the features of $\mathbf{x}$. Regression trees can be used for classifying by predicting the probability that $\mathbf{x}$ is in a certain class. For the regression trees we use `XGBoost` [11] that implements a decision tree ensemble model consisting of many regression trees for the probability of each class that are then combined to one single prediction. In a classification problem, `XGBoost` first generates a *base classifier* which is subsequently expanded to reduce the variance of the overall model. This expansion is called the boosting.

B. Transient stability

In this paper we will use machine learning to predict whether or not synchronous machines remain stable after a fault has split the system into two areas. This problem is related to transient stability, and we will therefore give some preliminaries in transient stability analysis. For more details the interested reader may for instance refer to [13]. For the analysis we will consider a single machine infinite bus (SMIB), which is a generator connected to an infinite bus through a line with a reactance X. The infinite bus can be viewed as an aggregated generator that is so large that its voltage, frequency, and angle will remain constant. For the generator we assume that it can be modelled as a constant electric field E'_q behind its synchronous reactance X_d. If we assume the mechanical power to be constant $P_m = P_{m0}$ we can write the swing equation for the SMIB as

$$\frac{2H}{\omega_s}\ddot{\theta} + K_d\dot{\theta} = P_{m0} - \frac{E'_q U_N}{X'_d + X}\sin\theta, \tag{1}$$

where H is the inertia constant of the generator, K_d is the damping factor, θ is the rotor angle relative to the infinite bus, and U_N is the voltage amplitude of the infinite bus.

A popular criterion for determining transient stability is the equal area criterion:

$$\int_{\theta_0}^{\theta_m} \frac{\omega_0}{H}(P_m - P_e)d\theta = 0, \tag{2}$$

where θ_0 is the relative rotor angle between the generator and the infinite bus at the start of the disturbance, and θ_m is the maximum angle. Let us now consider a sudden increase in the mechanical power from P_{m0} to P_{m1}. From (1) we see that this will accelerate the rotor leading to an increase in the rotor angle until it reaches θ_m. If the system is stable we see, from (2), that at some point the electric power will have to be larger than the mechanical power. When the electrical power is larger than the mechanical power we see from (1) that the rotor will be decelerated. In Fig 1, we have plotted (2) for this case. In the figure, we see two areas, one representing the acceleration phase, and one representing the deceleration phase. The equal area criterion states that these areas should be equal. The electrical power will typically drop during a disturbance such as a short circuit and the rotor will be accelerated. How fast the fault is cleared, so that the rotor can be decelerated, is therefore important.

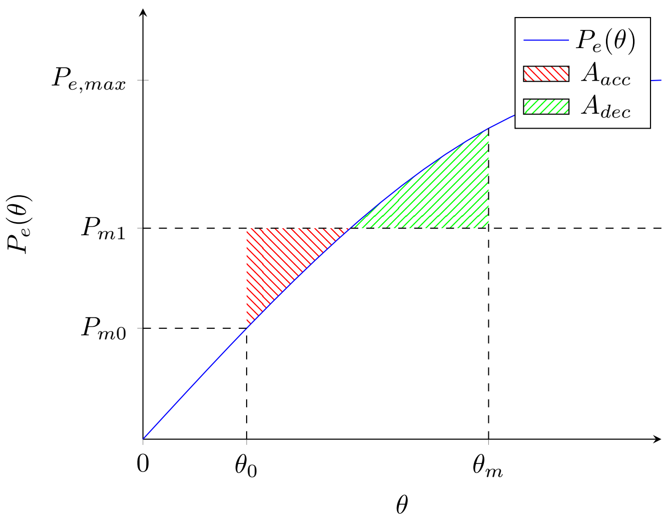

Fig. 1. Figure illustrating the equal area criterion. A_{acc} and A_{dec} refers to the area under the curve in the accelerating and decelerating phase respectively.

C. Feature selection

To train the machine learning model we need to select a set of features $\mathcal{X}$ that are given as input to the machine learning model. From (1) and (2), we see that the following power system variables are important for the transient stability: generator loading, fault clearing time, post-fault reactance, generator inertia, and generator and power system voltages. Since we are interested in predicting if individual power system islands are stable or not we will use the variables from the equal area criterion aggregated per area. For the area inertia we will use:

$$M_i = \sum_{g\in i} M_g = \frac{2}{\omega_0}\sum_{g\in i} H_g S_g \tag{3}$$

where g denotes a generator in area i, ω_0 is the synchronous frequency in rad/s, H_g is the inertia constant, and S_g is the rating of machine g. For the angle of an area we use the centre of inertia angle.

$$\theta_i = \frac{\sum_{g\in i} M_g\theta_g}{\sum_{g\in i} M_g} \tag{4}$$

where θ_i is the angle of the voltage at the generator bus bar. Preferably, we should have used the difference between the rotor angle and the voltage angle at the generator bus bar. However, we want to use variables readily available from a power flow solution. For the voltages and reactances we use the following variable

$$ux_i = \min_{g\in i} \frac{U_{N,g}}{X'_{d,g}} \tag{5}$$

For ux_i we also use the variables readily available for a power flow, except $X'_{d,g}$, however, which is not a variable that has to be calculated, unlike the grid reactance that is dependent on contingencies and grid configuration.

We also calculate the following powers:

$$P_{i,L} = \frac{1}{S_i}\sum_{l\in i} P_{l,L},\ P_{i,G} = \frac{1}{S_i}\sum_{g\in i} P_{g,G} \tag{6}$$

$$F_{i,sum} = \frac{1}{S_i}\sum_{g\in i} F_l,\ F_{i,max} = \frac{1}{S_i}\max_{g\in i} F_g \tag{7}$$

where $P_{g,G}$ is the power produced by machine g, $P_{l,L}$ is the power consumed by load l, F_l is a flow between areas, and S_i is defined as

$$S_i = \sum_{g\in i} S_g \tag{8}$$

The clearing time for faults t_c is also included as a feature.

III. Method

The aim of our method is to be able to compare the results from a time-domain simulation, a power flow, and a machine learning assisted power flow for the purpose of contingency analysis, specifically when a system is split into islands. To do this we developed a framework capable of consistently generating training data and comparing methods, which consists of a power system simulator, a data generation scheme and the training of a machine learning model.

A. Power system simulator

The developed tool was implemented in Python using the commercial software PowerFactory for the power flow and time-domain simulations. It was important to ensure that each contingency resulted in the same islands for all methods. To do this, each simulation started with a contingency analysis using a power flow. This implemented contingency analysis includes a very simple implementation of remedial actions. It works by disconnecting overloaded lines and buses that have a voltage under $0.9p.u.$. This is done in an iterative manner until there are no more overloads in the system. In case some loads are disconnected in this sequence, they are reported as lost. This sequence of events are stored, so that the same sequence can be run in the time-domain simulation. It should be noted that PowerFactory assumes all islands with sufficient generation to survive in power flow simulations.

The power system simulator can post process the results from the power flow contingency analysis using a time-domain simulation or a machine learning model. The software can also store the state of the system prior to the contingency for the sake of generating training data, or collecting features for the machine learning model.

The time-domain simulation works by running the same sequence of events as the power flow contingency analysis after the power flow contingency has finished, with the addition of including a clearing time, randomly sampled from $\mathcal{U}_{[0,1]}$ (seconds). If PowerFactory reports that one of the generators in an island has lost synchronism during the simulation, the island is considered unstable, and all load in that island is considered lost. The time-domain simulation is also responsible for marking islands in the training data as stable or not.

The machine learning model takes the state of each island from before the contingency happens and predicts whether or not the island will be stable or not. To generate the features for the prediction, a power flow is run on the operating state without any contingencies. After a contingency resulting in islands, the simulator finds the initial state of the components prior to the system separation and calculates the features. The machine learning model predicts a number $\mathrm{R} \in [0,1]$ for the island, and the island is classified as either stable or unstable based on a threshold value. All load is considered lost if the island is predicted to be unstable.

B. Generating training data

The training data was generated by running the time-domain simulation on contingencies resulting in island separation on several operating states. This procedure will with m operating states and n contingencies resulting in z islands result in $m \times n \times z$ data points. The operating states were generated by tweaking the base cases presented in [12]. This was done according to the following procedure, inspired by the method presented in [8]:

Set base case:
- Select random base case from list of base cases.

Distribute generation:
- Retrieve the total active generation for each machine $\mathbf{P}^N_{\text{gen}}$ in the base case.
- A random number of generators, between 1 and 5, is set as not in service and with zero generation.
- Choose $\mathbf{P}_{\text{gen}}$, such that the generation of each machine $P_{\text{gen},i}$ is sampled from $\mathcal{N}(P^N_{\text{gen},i}, \sigma^2)$, truncated between 0 and S_i where S_i is the machine rating. The variance σ^2 is chosen such that $\mathbb{P}(P_{\text{gen},i} < 0) = 0.01$.

Distribute load to areas:
- Let $P_{\text{tot}} = 0.99\sum_i P_{\text{gen},i}$ denote the total load level. The factor of 0.99 is to account for line losses.
- Retrieve the total active load for each area $\mathbf{P}^N_{\text{load}}$ in the base case.

- Randomly set weights $\boldsymbol{\zeta}$ s.t. all $\zeta_i = 1$ except one $\zeta_j = 5$.
- Distribute the load, $\mathbf{P}_{\text{load}}$ on the areas, based on $\boldsymbol{\zeta}$ and $\mathbf{P}_{\text{load}}^N$ according to the following formula

$$\mathbf{P}_{\text{load}} = \frac{\boldsymbol{\zeta} \odot \mathbf{P}_{\text{load}}^N}{\boldsymbol{\zeta} \cdot \mathbf{P}_{\text{load}}^N} P_{\text{tot}} \tag{9}$$

where the operator $\odot$ represents the Hadamard (element-wise) product.

Distribute load within areas:

- To further distribute the load within each area, choose new weights $\boldsymbol{\lambda}$ sampled from $\mathcal{U}_{[1,10]}$ associated to each load in the area. Update the loads of the buses in a similar manner as was done on the areas

$$\mathbf{P} = \frac{\boldsymbol{\lambda} \odot \mathbf{P}^N}{\boldsymbol{\lambda} \cdot \mathbf{P}^N} P_{\text{tot}}^{\mathcal{A}},$$

where $P_{\text{tot}}^{\mathcal{A}}$ is the power distributed on the area, $\mathcal{A}$, to which the load belongs, in the preceding step. That is, any one of the components of $\mathbf{P}_{\text{load}}$.

By distributing load and generation in this way, one ends up with not *too* extreme deviations from the base case, since loads and generations that in the base case are large, will have a tendency to be large also when tweaked. This is useful in the sense that the analysis is less prone to convergence issues and numerical instability. Furthermore, choosing the area-specific weights for loads in this way ensures that one of the areas will be importing more power than the others, increasing the chances of large flows between areas and instability. In addition to this, a random number of generators (at most 5) is chosen to be taken out of service before the time-domain simulation. This is done in order to vary the inertia of each area.

C. Machine learning

We consider binary classification and hence the model is trained with a binary cross entropy loss function, and the output is a probability of the island being stable.

There are a number of different hyper-parameters which can be tuned in an `XGBoost` model; examples are `max_depth`, which regulates the depth of each regression tree, and `learning_rate` which scales each regression tree output by a constant [14]. We optimise the different parameters associated to our `XGBoost` model by performing a Bayesian hyper-parameter tuning combined with K-fold cross-validation on a subset of the total data set. The Bayesian hyper-parameter optimisation has recently become a standard method for hyper-parameter optimization thanks to its efficiency [15], [16]. The complete hyper-parameter space is shown in Table I. To prevent overfitting we apply early stopping, which stops the training before if the previously added `early_stopping_rounds` epochs have not lead to any improvement. We use 5 rounds while doing hyper-parameter optimization and 20 rounds in the final training.

The training data exhibits a notable degree of imbalance between the number of stable and unstable cases, despite efforts in the data generation process to produce sufficient unstable cases. This has an effect on the choice of evaluation metric, and the setting of the probability threshold separating the classes. A model trained on unbalanced data could perform well according to many metrics, simply by always predicting the majority class. This leads to poor prediction of the minority class. One suggestion is to use the area under the precision and recall curve (AUC PR) as an evaluation metric, which has been shown to have good properties when evaluating rare events, rather than the area under the receiver operating characteristic curve (ROC AUC). In this paper the model is trained using the AUC PR. Furthermore, in a balanced data set the probability threshold separating a binary class is usually set to 0.5, but this threshold is not necessarily ideal for imbalanced data. AUC PR is a threshold-independent metric, and it has been suggested to tune the classification threshold value to obtain the best F1 metric (the harmonic mean of precision and recall) based on the training set [17]–[19], an approach which is adopted in this paper.

TABLE I
DESCRIPTION AND DOMAIN OF HYPER-PARAMETERS IN THE SPACE USED FOR TUNING THE MODEL[a].

Hyper-parameter	Description	Domain
`early_stopping_rounds`	Number of training epochs without further model improvement after which to stop training.	$5, 20$ (fixed)
`learning_rate`	Rate with which to reduce the weight of each new tree added to the model in order to prevent overfitting.	$(0, 1]$
`subsample`	The propotion of all samples randomly selected for construction of each regression tree, for preventing overfitting.	$[0.7, 1.0]$
`max_depth`	Maximum depth of tree.	$[2, 10]$
`gamma`	The minimal loss reduction required before making a further partition of each leaf node.	$[3, 10]$
`reg_alpha`	L_1 regularisation term on weights.	$[0, 1]$
`reg_lambda`	L_2 regularisation term on weights.	$[0, 1]$
`colsample_bytree`	Subsample ratio of columns when constructing each tree.	$[0.5, 1]$
`min_child_weight`	Minimum sum of instance weight needed in a child.	$[1, 10]$
`num_parallel_tree`	Number of trees contributing to the boosted model.	$[10, 200]$

[a)] We refer to the `XGBoost` documentation for extensive explanations of the parameters. All the hyper-parameters are initially drawn from uniform distributions.

IV. RESULTS

The model was trained, validated and tested with a 50-30-20 percent split, on 63,756 data points generated by simulating 4,000 operating states for the system depicted in Fig. 2 subjected to three contingencies separating the rest of the grid from area 1, area 2 or area 3 and 4. The simulated operating states resulted in 7,361 unstable islands, and 56,396

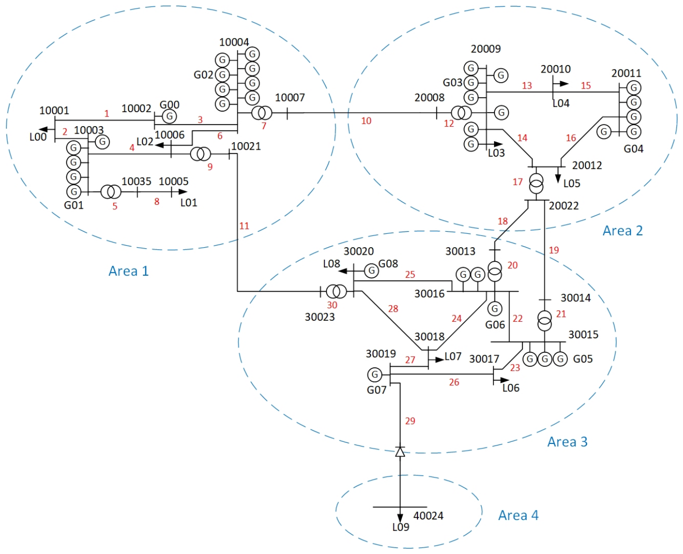

Fig. 2. Four area grid [12].

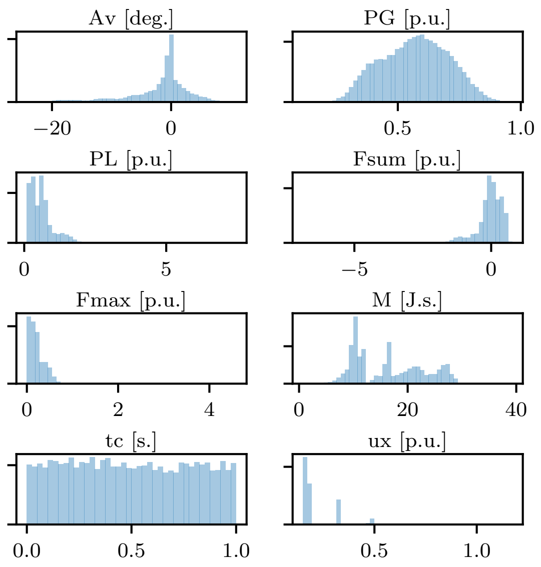

Fig. 4. Density histogram of features in the training data.

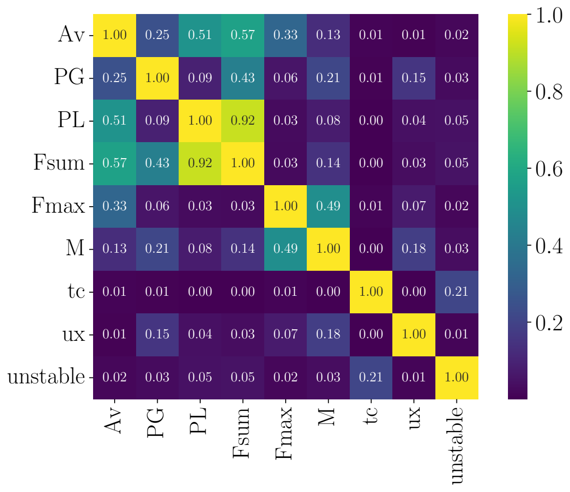

Fig. 3. Spearman correlation matrix of the generated features based on the entire dataset. We see that the stability measure is most correlated with the fault clearing time.

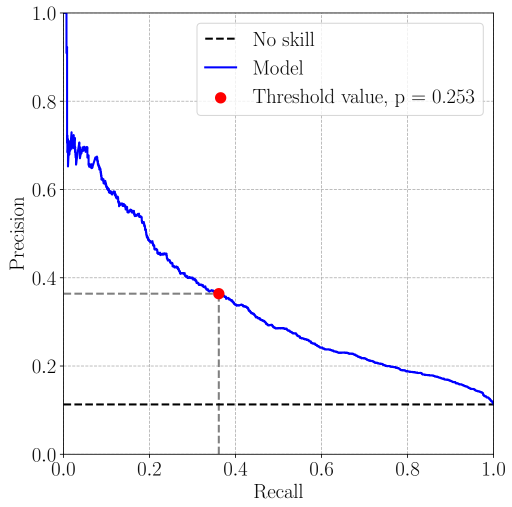

Fig. 5. Precision-recall curve of the XGBoost model. Red scatter indicates precision and recall at the selected probability threshold value.

stable islands, a data set with an imbalance ratio of 0.13. The histogram of the generated features are shown in Fig. 4. It shows that we have a reasonable spread of the features. However, as seen from the correlation plot Fig. 3, some of the features may later be discarded for a simpler model.

The ability of the model to predict is illustrated on the test-set in Fig. 5, showing that the model is significantly better than pure guessing, as indicated by the dashed black line. The performance of the machine learning model can add value to alternatives only relying on a static method if a large set of operating states is considered as part of a reliability analysis, even if it has shortcomings when analysing isolated cases.

The developed simulator was applied to the same test system and contingencies, on an additional 1000 operating states which were not used as part of the model training and testing procedure. Results were calculated using a power flow, a time-domain simulation and a power flow together with the trained machine learning model. The aggregated results from the comparison are shown in Table II. It shows that the machine learning model provides accurate results for lost load for a large sample, with a near identical predicted lost load compared to the more detailed time-domain simulation. The time-domain simulation suggest a lost load almost 11 times that found in a simple static simulation. This means that the machine learning model is a drastic improvement compared to the static model. Moreover, the machine learning model uses less than twice as much time as the static, whereas the

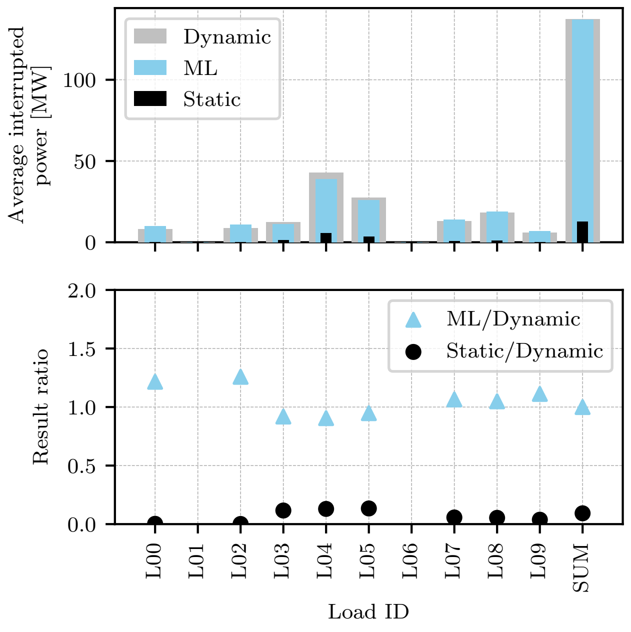

Fig. 6. Overview of lost load by load point per case (top panel), and ratio of lost load found by the static and AI method, compared to the dynamic analysis.

time-domain simulation uses over 13 times more.

TABLE II
RESULTS

Simulation	Average interrupted power (MW)	Time (minutes)
Static	12.67	3.10
Time-domain	137.26	41.97
ML	137.24	5.93

Results per load point is depicted in Fig. 6. The figure shows that the machine learning predictions differ somewhat between the areas, and that the model may perform differently for the various islands the test system typically is split into. Loads L01 and L06 never draw any load in the base operating state, thus the interrupted power at these load points are always zero, and the result ratio is not reported. Loads 1 to 2 in area 1 has a ratio of lost load slightly higher than what is found in the time-domain simulations. Load 3 to 5 in area 2 is slightly lower, while load 5 to 6 in area 3 and 4 is near the same. This could indicate that there are some features missing which differentiate stability based on important topological and/or electrical differences between the islands.

V. Concluding Remarks

In this paper we performed a preliminary analysis and comparison of a purely static, a time-domain and a machine learning approach for calculating lost load for a system subjected to islanding. The aim was to achieve better results than with a purely static approach and faster results than with a time-domain simulation. Our study demonstrated that this is indeed possible. Although generating the training data is time consuming, the speed up may be worth it if one is performing many simulations or if the simulations have to be executed relatively fast. Potential further work is suggested to be on feature selection to more accurately capture important topological and electrical differences in the islanded systems, and to verify the results on other power systems.

References

[1] GARPUR Consortium, “D1.1: State of the art on reliability assessment in power systems,” Tech. Rep. https://www.sintef.no/projectweb/garpur/, 2014.

[2] GARPUR Consortium, “D9.2: A transition roadmap towards probabilistic reliability management.”

[3] GARPUR Consortium, “D1.2: Current practices, drivers and barriers for new reliability standards,” Tech. Rep. https://www.sintef.no/projectweb/garpur/, 2014.

[4] I. B. Sperstad, E. H. Solvang, and S. H. Jakobsen, “A graph-based modelling framework for vulnerability analysis of critical sequences of events in power systems,” *International Journal of Electrical Power and Energy Systems*, vol. 125, 2021. Publisher: Elsevier Ltd.

[5] Y. Xue, T. Van Custem, and M. Ribbens-Pavella, “Extended equal area criterion justifications, generalizations, applications,” vol. 4, no. 1, pp. 44–52.

[6] Y. Xue, L. Wehenkel, R. Belhomme, P. Rousseaux, M. Pavella, E. Euxibie, B. Heilbronn, and J.-F. Lesigne, “Extended equal area criterion revisited (EHV power systems),” *IEEE Transactions on Power Systems*, vol. 7, pp. 1012–1022, Aug. 1992. Conference Name: IEEE Transactions on Power Systems.

[7] E. Karangelos and L. Wehenkel, “An iterative AC-SCOPF approach managing the contingency and corrective control failure uncertainties with a probabilistic guarantee,” *IEEE Transactions on Power Systems*, vol. 34, no. 5, pp. 3780–3790, 2019.

[8] L. Wehenkel and M. Pavella, “Decision trees and transient stability of electric power systems,” *Automatica*, vol. 27, no. 1, pp. 115–134, 1991.

[9] O. A. Alimi, K. Ouahada, and A. M. Abu-Mahfouz, “A review of machine learning approaches to power system security and stability,” *IEEE Access*, vol. 8, pp. 113512–113531, 2020.

[10] A. Zerigui, L.-A. Dessaint, R. Hannat, R. T. Ah King, and I. Kamwa, “Statistical approach for transient stability constrained optimal power flow,” vol. 9, no. 14, pp. 1856–1864.

[11] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’16, (New York, NY, USA), p. 785–794, Association for Computing Machinery, 2016.

[12] I. B. Sperstad, E. H. Solvang, S. H. Jakobsen, and O. Gjerde, “Data set for power system reliability analysis using a four-area test network,” *Data in Brief*, vol. 33, 2020. Publisher: Elsevier Inc.

[13] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*, vol. 7. McGraw-hill New York.

[14] “Elements of Statistical Learning: data mining, inference, and prediction. 2nd Edition..”

[15] A. Klein, S. Bartels, S. Falkner, P. Hennig, and F. Hutter, “Towards efficient bayesian optimization for big data,” 2015.

[16] L. Wang, F. Dernoncourt, and T. Bui, “Bayesian optimization for selecting efficient machine learning models,” 2020.

[17] H. He and E. A. Garcia, “Learning from imbalanced data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[18] Q. Zou, S. Xie, Z. Lin, M. Wu, and Y. Ju, “Finding the best classification threshold in imbalanced classification,” *Big Data Research*, vol. 5, pp. 2–8, 2016. Big data analytics and applications.

[19] H. R. Sofaer, J. A. Hoeting, and C. S. Jarnevich, “The area under the precision-recall curve as a performance metric for rare binary events,” *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 565–577, 2019.