Hoeffding-Tree-Based Learning from Data Streams and Its Application in Online Voltage Security Assessment

Zhijie Nie

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Virgilio Centeno, Chair
Jaime De La Ree Lopez
Vassilis Kekatos

August 8, 2017
Blacksburg, Virginia

Keywords: Power Systems Stability, Voltage Security Assessment, Machine Learning, Decision Trees, Hoeffding Trees

Hoeffding-Tree-Based Learning from Data Streams and Its Application in Online Voltage Security Assessment

Zhijie Nie

ABSTRACT

According to the proposed definition and classification of power system stability addressed by IEEE and CIGRE Task Force, voltage stability refers to the stability of maintaining the steady voltage magnitudes at all buses in a power system when the system is subjected to a disturbance from a given operating condition (OC). Cascading outage due to voltage collapse is a probable consequence during insecure voltage situations. In this regard, fast responding and reliable voltage security assessment (VSA) is effective and indispensable for system to survive in conceivable contingencies. This paper aims at establishing an online systematic framework for voltage security assessment with high-speed data streams from synchrophasors and phasor data concentrators (PDCs). Periodically updated decision trees (DTs) have been applied in different subjects of security assessments in power systems. However, with a training data set of operating conditions that grows rapidly, re-training and restructuring a decision tree becomes a time-consuming process. Hoeffding-tree-based method constructs a learner that is capable of memory management to process streaming data without retaining the complete data set for training purposes in real-time and guarantees the accuracy of learner. The proposed approach of voltage security assessment based on Very Fast Decision Tree (VFDT) system is tested and evaluated by the IEEE 118-bus standard system.

Hoeffding-Tree-Based Learning from Data Streams and Its Application in Online Voltage Security Assessment

Zhijie Nie

GENERAL AUDIENCE ABSTRACT

Voltage security is one of the most critical issues in the power systems operation. Given an operating condition (OC), Voltage Security Assessment (VSA) provides a tool to access whether the system is capable to withstand disturbances if there is one or more than one elements is not functioning appropriately on the power grid. Traditional methods of VSA require the knowledge of network topologies and the computational contingency analysis of various circumstances. With trained models, decision-tree-based VSA is able to assess the voltage security status by collectible measurements among the system in a real-time manner. The system topology may alter over and over by system operators in order to meet the needs of heavy load demand and power quaility requirements. The proposed approach based on Very Fast Decision Tree (VFDT) system is capable of updating trained decision-tree models regarding to changes of system topology. Therefore, the updated decision-tree models is able to handle different system topology and to provide accurate security assessment of current OC again.

Acknowledgments

I would like to express my gratitude to Dr. Virgilio Centeno, my committee chair and advisor, who has been provide guidance, support, and assistance behind this thesis. And I am also grateful to my fellow Dr. Duotong Yang, who has incented me, and broaden my academic scope from various perspectives.

I am also grateful to Dr. Kevin D. Jones from Dominion Energy for his encouragement and help.

I would also like to thank Dr. Jaime De La Ree Lopez and Dr. Vassilis Kekatos for their patience and support in overcoming problems that I have been confronted with during my research and coursework.

Last but not least, I would also like to acknowledge my fellow doctoral students at Center for Power and Energy for their friendship: Chen Wang, Tapas Kumar Barik, Andreas Schmitt, Elliott Mitchell-Colgan, Jacques Delport, and Xiawen Li.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern power systems are experiencing an unprecedented occasion with high-speed data streaming due to the increasing amount of installations of Phasor Measurement Unit (PMU) and Advanced Metering Infrastructure (AMI). One of the great challenges for power electric utilities and regional transmission organizations (RTOs) is to meet the system-wide voltage security requirement. Therefore, it is critically important for operators and coordinators to build an intelligent tool for wide-area situational awareness and decision-making in response to the continuously streaming data. With machine learning techniques implemented, controllers driven by PMU measurements and synchrophasor-based linear state estimator (LSE) would be the keystones to achieve this goal [1–4]. The study of this paper aims to design a framework for online Voltage Security Assessment (VSA) for region voltage controller that provides voltage security status analysis within a control region where PMUs are installed. It is expected that this proposed assessment framework is capable to handle real-time high-speed data streams.

From the time being, there are various voltage control methodologies developed for transmission grids in the past century. Instead of solving the single-constrained OPF problem [5–7], researchers from Denmark and United States proposed a two-phase control scheme based on decision trees (DTs) [8]. The proposed DT model in the first phase is developed for identifying potential security issues in dynamic security assessment. Then, the second phase DT model provides online decision support on preventive control strategies. The insecure operating conditions (OCs) are detected by the first DT, then the final control trajectory

is searched out of numerous new OCs based on optimal generators outputs subject to minimum economic cost. While this approach provides faster online situational awareness and controllability, its accuracy and reliability are highly dependent on how the learning data set of OCs is generated. Topology changes due to forced outages of network elements could result in differences between the offline learning database and the real-time operating conditions, and ultimately cause inaccurate classification predictions. A more robust data mining framework introduced in [9] shows its effectiveness in dealing with variations of OCs and system topology changes by using online ensemble learning method.

The study of this paper aims at designing a framework of online voltage security assessment that timely provides decisions on security status for real-time system operating conditions. It is also expected that the proposed decision-tree-based voltage security assessment is capable to deal with high-speed PMU data streams. To provide a cost-effective and accurate VSA scheme, there are two primary issues inherent in decision-tree-based methodologies with respect to high-speed data streams. First, machine learning techniques provide power systems with a probabilistic perspective, which is highly dependent on the pre-analyzed data set of OCs and their possible variances. However, system configurations like transformer load tap changers (LTCs) switching or topology changes, might be conducted frequently during a day at the system operation center. Hence, an offline-trained decision-tree model may return inaccurate predictions due to the differences between the prepared OCs and the unexpected actual OCs. Second, regarding to the high-speed data streams collected from PMUs, it is necessary to establish a timely update scheme for the trained model in order to adapt to the modified system configurations and new operating situations.

To maintain the robustness of such a decision-tree-based assessment, Diao *et al.* [3] introduced multiple optimal DTs and corrective DTs to further improve the performance as an addition to the trained model, but it also increases the burdens of computation complexity and memory management with an increasing number of DTs. He *et al.* [9] employed an adaptive ensemble decision-tree learning method, which adjusts the voting weights of weak learners based on inaccurate predictions, and flips the predictions whose errors are more than 0.5 during each periodic update. In the paper, Incremental Tree Induction (ITI) [10] is used as a lossless online updating algorithm; with a new incoming data set, ITI provides the same DT ultimately as the DT built from scratch using Quinlan's ID3 [11]. However, this technique is memory-intensive and time-consuming [12] because it involves recursive transpositions that require storing all the training and updating data within the decision nodes.

Alternatively, in regards to mining high-speed data streams, Hoeffding trees (HTs) [13] as an incremental decision-tree learning method is proposed and capable to be trained in constant time per sample. Empirical studies have revealed its advantages when continuing to update the DT model with massive numbers of data samples [14].

1.1 Risks of Power Systems Voltage Security

In recent years, with increasing demand on electricity, the voltage security problem still raises some concerns regarding to current complex transmission networks. Voltage security has been addressed as a critical problem for system operation since the 1920s [15]. As listed in [16], voltage collapses and voltage disturbance incidents happened in many mature systems worldwide in the past century. From previous occurrences of voltage issues, discussions have been elevated to IEEE/CIGRE Joint Task Force in order to tackle voltage issues with a systematic definition. Here, the definitions of voltage security established and developed in [17] are adopted, which are:

- The ability of a power system to maintain steady voltages at all buses in the system after being subjected to a disturbance from a given initial operating condition, which depends on the ability to maintain/restore equilibrium between load demand and load supply from the power system.

And four categories have been classified regarding to voltage security issues: *large-disturbance voltage stability*, *small-disturbance voltage stability*, *short-term voltage stability*, and *long-term voltage stability*. This paper focuses on small-disturbance short-term voltage stability that involves load-tap-changers (LTCs), shunt compensators, and any other slower acting equipment when subjected to increasing load in the system. The time-responses for short-term voltage stability may vary from several to many minutes, and the static system analysis method is applied to study the factors influencing the stability margins and the effects of these acting equipment.

1.2 Overview of Machine Learning Methods

Under the background of Big-Data, data analysts and engineers are capable to collect large amounts of data from images, videos, sensors, Internet-of-Things (IoT) devices, etc.. *Machine Learning*, explores the study and construction of pure data-driven algorithms that finds patterns in data and make predictions with newly arrived data based on established models.

In machine learning studies [18, 19], two types of categorization of machine learning algorithms are commonly discussed.

1. Depending on whether a learning technique requires "labels" or not, the algorithms in machine learning can be classified into two categories, which are supervised learning and unsupervised learning. Supervised learning focuses on learning a rule to map inputs and outputs with presented desired "labels" for outputs. And unsupervised learning develops hidden patterns in data without given "labels" to find the nature in the inputs.

2. Depending on the output from different studied tasks, the algorithms in machine learning can be classified into three categories, which are classification, regression, and clustering.

    - *Classification* algorithms establish models to classify data into specific categories, which are regarded discrete "labels". Popular algorithms include k-Nearest Neighbor (kNN), Logistic Regression, Naive Bayes methods, Support Vector Machine (SVM), Decision Trees, etc..
    - *Regression* algorithms build models to predict continuous datapoints as "labels". Popular algorithms include Linear Regression, Neural Networks (NN), Decision Trees, etc..
    - *Clustering*, as an unsupervised learning task, finds the nature groupings or patterns on the data without any "labels" provided in the original dataset. Popular algorithms include k-Means method, Gaussian Mixture Models (GMM), hidden Markov models, etc..

Examples of machine learning problems are listed in Table 1.1.

Table 1.1: Overviews of Machine learning algorithms

Task	Category	Common Problems for Studies
Classification	Supervised learning	Spam filtering, cancer diagnosis, bank marketing
Regression	Supervised learning	Electricity load forecasting, pre-programmed trading, communities and crime study
Clustering	Unsupervised learning	Character trajectories, facial recognition

Decision Trees approach in the classification usage is a broadly discussed machine learning methodology applied in voltage security assessment and dynamic security assessment of power systems [3, 8, 20, 21], which proposed a novel way to tackle system-wide security problem instead of using conventional methods like P-V and V-Q curves analysis. With regard to security assessment problems, decision trees approach offers simple and direct decision rules to classify operating conditions, which benefits the real-time operation of power systems analytics. On the use of decision trees, Cutsem *et al.* [20, 21] proposed machine learning frameworks for voltage security assessment and transient security assessment of power systems. In these application, a decision tree (DT) model is established to classify specific operating conditions as "Secure" and "Insecure".

An example of DT model is shown in Figure 1.1. A trained DT model is structured as a flowchart-like tree drawn upside down, consisting of two groups of internal nodes: attribute nodes, including the "root", represent testing conditions for particular nodes regarding to the input parameters; and terminal nodes, indicate the results for a set of input values as class "labels".

In this paper, an adaptive ensemble decision-tree-based voltage security assessment using Very Fast Decision Tree (VFDT) system for updates is proposed in order to guarantee the robustness of the proposed technique after possible topology changes or any unpredictable system conditions captured by PMU data streams. Initially, the ensemble DT model consists of multiple DTs with voting weights generated by the Adaptive Boosting (AdaBoost) algorithm. Instead of re-training the whole set of DTs from scratch, an adaptive DT updating scheme of VFDT system is introduced to update the ensemble trees. The proposed methodology provides a lightweight method that benefits memory management for high-speed data streams, which requires only the latest updating data and basic statistical analysis instead

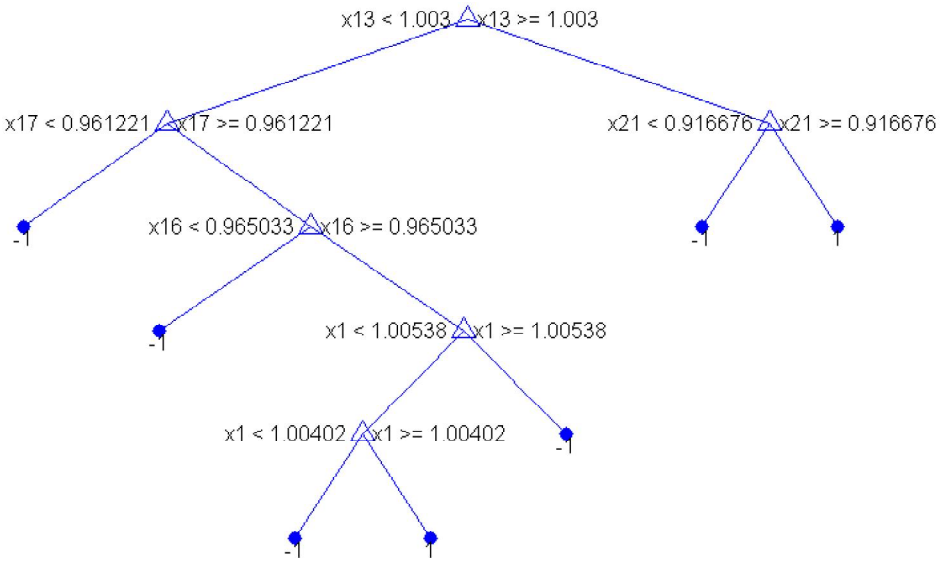

Figure 1.1: Example: A trained decision tree model structure

of the whole data set for updating purposes.

The remaining parts of this paper is organized as follows: Chapter 2 details the induction process of a decision tree, discuss the ensemble learning of DTs, and illustrates the basic concept of voltage security assessment. Chapter 3 demonstrates the framework VFDT system scheme applied in online voltage security assessment analytics. Simulation results of proposed methodology using IEEE 118-bus system model are presented in Chapter 4. Finally, conclusions are drawn in Chapter 5.

Chapter 2

Background on Decision Trees and Online Voltage Security Assessment Framework

2.1 Overview of Decision Trees and Ensemble Learning Method

The decision tree methodology, is a supervised learning technique that offers a statistical perspective identifying the inherent boarders between multiple classes by learning the thresholds of parameters in the candidate attributes that each distinct class possesses.

First of all, clarify some terminology used throughout the explanation of inductive process of a decision tree model.

- *Attributes*: parameters from the input data
- *Attribute node*: a node that splits data into successor nodes based on certain criterion against a threshold value on an attribute[1]

[1]The "root" is also an attribute node with certain criterion against a threshold value on an attribute.

Table 2.1: A training dataset example for supervised learning task

Instance	Desired Label	Attrib.1	Attrib.2	Attrib.3	...	Attrib.M
$\boldsymbol{x}_1$	ClassA	1.004	0.976	0.988	...	1.062
$\boldsymbol{x}_2$	ClassB	0.999	1.026	0.967	...	1.035
$\boldsymbol{x}_3$	ClassB	1.011	1.023	0.937	...	1.075
...	...	...	...	...	...	...
$\boldsymbol{x}_N$	ClassA	1.021	0.983	0.961	...	1.015

- *Best split*: the one obtaining the highest information is selected as the basis at each attribute node by applying the heuristic measurement on all possible attributes
- *Data instance*: a series of attribute values related to a certain desired label, examples for data instances are shown in 2.1 representing a training dataset
- *Desired label*: the given class label of a data instance
- *Heuristic measurement*: the indicator that measures the information by the parameters in all data instances of the dataset
- *Split*: a process of changing a node to an attribute node and splitting into multiple terminal nodes
- *Predicted label*: the class label of a data instance predicted by the trained model
- *Terminal node*: a node indicates the class to which the data instance belongs, used interchangeably with "leaf node"

2.1.1 Induction of a Decision Tree

The growing of a decision tree from scratch is an inductive process starting from the "root". A pseudo-code for an incremental tree induction process can be summarized as follows:

1. Initialize the tree with a "root" with the training dataset installed;

2. At a node, find the best attribute according to the *best splitting rule* for each attribute;
3. Assign the attribute node with best splitting criterion achieved above, and split the installed data instances into two successor nodes;
4. Determine whether to stop the induction process at successor nodes according to the *termination rule*;
5. Recursively call the induction process for all non-terminal nodes.

It is required to answer two major questions throughout the induction, namely:

- To find the best split points, what kind of heuristic measurements for a splitting criterion is efficient in the classification task?
- Should the node-splitting process be terminated or continued? If terminated, which class "label" should be assigned for a terminal node?

Best Splitting Rule

On the use of a DT model, an object could be grouped into a specific class using an inductive approach, which searches exhaustively for the best split points at all attribute nodes. The *Gini index*, an alternative approach of Information Gain, favors larger partitions in the two-class problems; it measures the input node to efficiently choose the best split position that has the minimum value to generate the DT inductively. A general expression of Gini index is given by

$$Gini = 1 - \sum_{c=1}^{n_C} (p_c|k)^2 \tag{2.1}$$

where n_C is defined as the number of discrete classes, and p_c denotes the probability of a specific class at the fallen node k. This approach is also used as the heuristic measurement $G(\cdot)$ to find the best splits when deriving the Hoeffding tree in Chapter 3.2.

With Gini index implemented as the heuristic measurement, notice that the higher the score, the better the criterion that distinguishes class labels with one attribute. Therefore, the best splitting rule for an attribute node is to choose the highest score in heuristic measurement as the splitting criterion to ensure the efficiency of the decision tree.

Termination Rule

By applying the best splitting rule above, a decision tree could be fully established to tell all training instances apart with accurate class label predictions just as CART algorithm provides[2]; however, this procedure is unnecessary and poses a risk of *over-fitting* for a developed classification model. As a result, a well-developed decision tree model does not only distinguishes different classes efficiently, but also maintains a high training accuracy with a rational number of tree levels. The termination rule could be applied if decrease in impurity is smaller than a user-defined bound in heuristic measurement.

2.1.2 Ensemble Learning of Decision Trees

Ensemble learning methodology is a process that utilizes multiple models of expert to improve the performance of single weak learning algorithm. In this paper, the Adaptive Boosting (AdaBoost) algorithm [22] is adopted in the offline VSA training to improve the performance and accuracy of the security assessment of OC variations and unexpected topology changes. AdaBoost is an ensemble learning algorithm which linearly combines the outputs from L weak hypotheses of decision tree $h_l(\boldsymbol{x})$ with a weighted sum to represent the final result $H(\boldsymbol{x})$ of the combined model.

$$H(\boldsymbol{x}) = sgn\left(\sum_{l=1}^{L} \alpha_l h_l(\boldsymbol{x})\right) \tag{2.2}$$

where α_l denotes the voting weight for the weak classifier h_l, and $sgn(\cdot)$ is the sign function (or signum function) that extracts the sign of a real number.

[2]CART (*Classification And Regression Trees*) algorithm fully grows a decision tree model to realize zero impurity, and prunes the tree by assigning terminal nodes regarding to the largest probability of $(p_c|k)$.

Given a training dataset $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, in which for any i, $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$. Initially, each the training instance is uniformly distributed with a weight $w_i^{(1)} = \frac{1}{n}$, then call the weak learner to train with this distribution $\boldsymbol{d}^{(1)}$. After that, calculate the training error e_1 under the distribution, and set the voting weight α_l for each classifier as follows:

$$\alpha_l = \frac{1}{2} \ln \left(\frac{1 - e_l}{e_l} \right) \tag{2.3}$$

This iterative process is sequentially continued for L times by re-distributing the weights of data instances with

$$d_i^{(l+1)} = d_i^{(l)} \frac{\exp\left(-y_i \alpha_l h_l\left(\boldsymbol{x_i}\right)\right)}{\Delta_l} \tag{2.4}$$

where Δ_l is the normalization factor for the distribution for the next weak learner $\boldsymbol{d}^{(l+1)}$.

To demonstrate the ensemble learning method, Table 2.2 details a randomly generated 2-D training dataset with 10 instances labeled as "+1" and the other 10 labeled as "−1". Figure 2.1 presents a scatter graph of these two-class data instances, in which "+1" class instances are represented by blue plus sign markers, and "−1" class instances are represented by red circle markers.

By applying AdaBoost algorithm to fit the training dataset to a classification model, here $L = 6$ weak learners are adopted in this example. To simplify the demonstration, weak learners adopted are simple one-level decision trees, each of which only consists of one "root" and two "leafs" corresponding to the classification results at the "root". Table 2.3 illustrates all classification criteria and the voting weights of each weak classifiers. The trained ensemble model of all weak classifiers is shown in Figure 2.2. The voting weights are revealed in the form of color saturation in the figure.

Table 2.2: A training dataset for ensemble learning example

Label	x_1	x_2
$+1$	-0.7867	0.7061
$+1$	0.9238	0.2441
$+1$	-0.9907	-0.2981
$+1$	0.5498	0.0265
$+1$	0.6346	-0.1964
$+1$	0.7374	-0.8481
$+1$	-0.8311	-0.5202
$+1$	-0.2004	-0.7534
$+1$	-0.4803	-0.6322
$+1$	0.6001	-0.5201
-1	-0.1371	-0.1655
-1	0.8213	-0.9007
-1	-0.6363	0.8054
-1	-0.4723	0.8896
-1	-0.7089	-0.0183
-1	-0.7278	-0.0215
-1	0.7386	-0.3246
-1	0.1594	0.8001
-1	0.0997	-0.2615
-1	-0.7101	-0.7776

Table 2.3: An ensemble decision trees trained by AdaBoost

l	Voting weight	Criterion	if **True**	if **False**
1	0.1413	$x_2 < 0.7531$	$+1$	-1
2	0.2280	$x_1 < -0.7573$	$+1$	-1
3	0.1726	$x_1 < 0.3546$	-1	$+1$
4	0.2044	$x_1 < 0.7380$	$+1$	-1
5	0.0999	$x_1 < 0.8725$	-1	$+1$
6	0.1536	$x_2 < -0.4223$	$+1$	-1

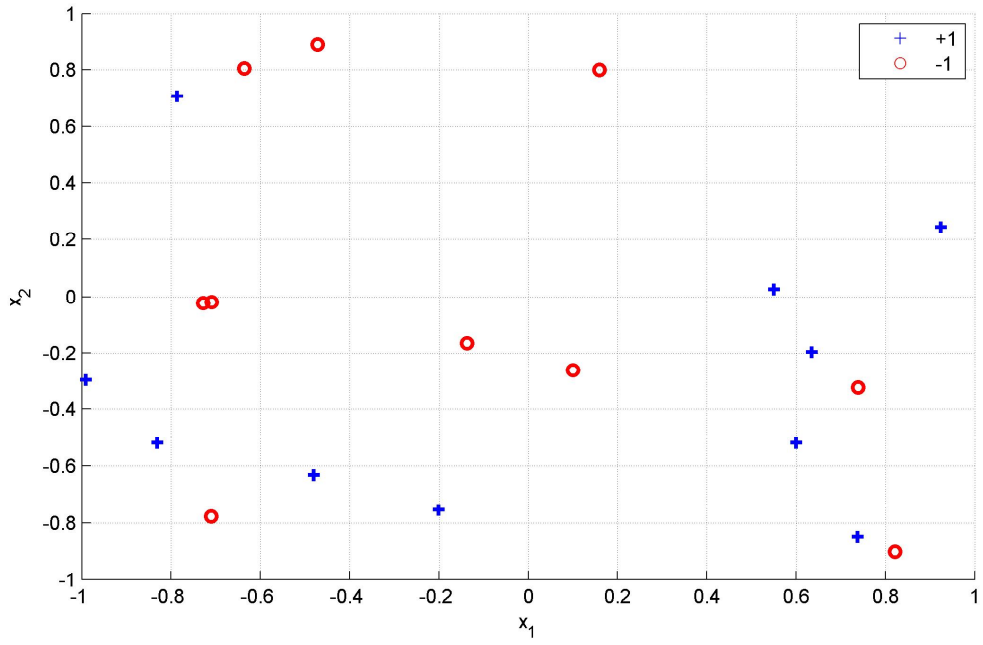

Figure 2.1: A training dataset for ensemble learning example

2.2 Voltage Security Assessment

From a general perspective [23], power system stability analysis concentrates on the capability of withstanding any conceivable disturbance under a certain operating condition (OC). Voltage Security Assessment is an evaluation tool that conducts voltage and steady-state stability computations to determine whether the system is operating within the maximum loadability, and it is also capable to be execute to determine the post-control stability status of a particular OC [2,8]. In practice, the P-V curves and V-Q curves are two commonly used techniques for voltage collapse analysis of power systems. The bifurcation points (or knee points), of P-V curves and V-Q curves indicate the maximum loadability limits that the power could be delivered without incurring any voltage instability issues. Voltage collapses will occur on the heavily loaded system with a load demand higher than the maximum loadability. These knee points are the bifurcation points of the nonlinear power system model. Represented in mathematical terms, the maximum loadability corresponds to a scalar function ζ of $\boldsymbol{p}$, in which $\boldsymbol{p}$ represents the load demand vector $\boldsymbol{p} = [P_1, \ldots, P_m, Q_1, \ldots, Q_m]$, where m denotes the number of buses in the system. Then, consider the following optimization problem as follows:

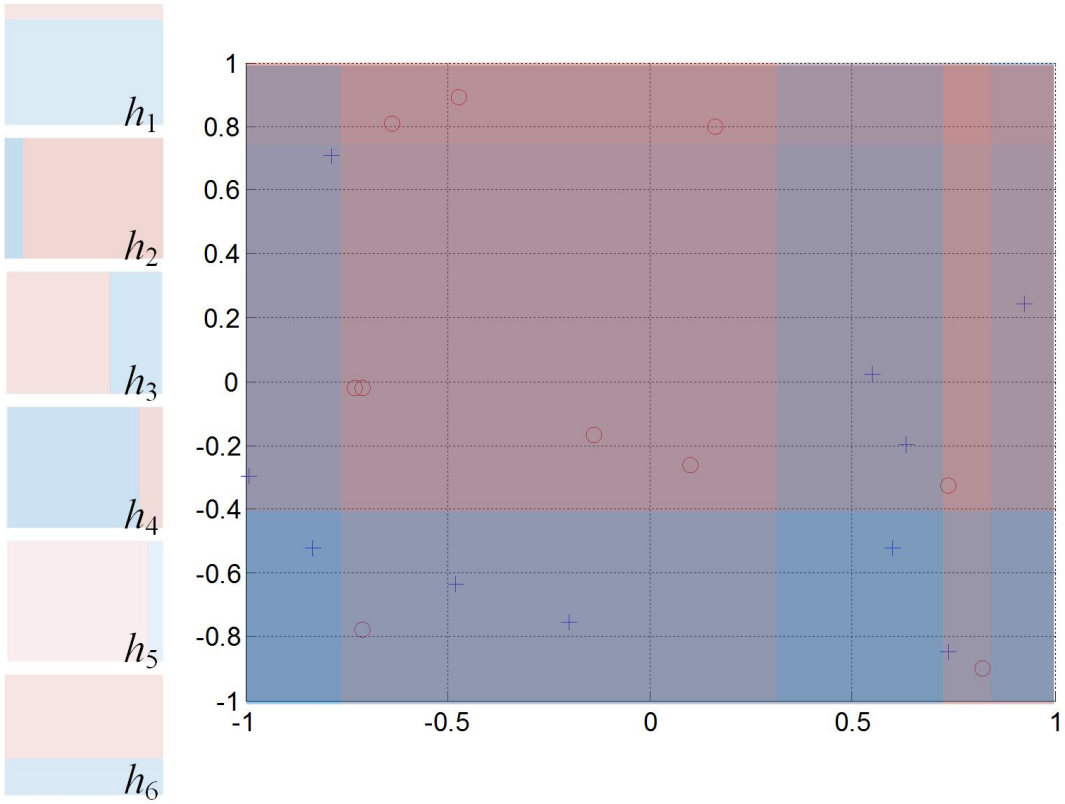

Figure 2.2: An ensemble decision trees trained by AdaBoost

$$\begin{aligned} \max_{\boldsymbol{p},\boldsymbol{u}} \quad & \zeta(\boldsymbol{p}) \\ \text{subject to} \quad & \boldsymbol{\varphi}(\boldsymbol{p},\boldsymbol{u}) = \boldsymbol{0} \end{aligned} \tag{2.5}$$

where $\boldsymbol{\varphi}(\boldsymbol{p},\boldsymbol{u}) = 0$ represents the set of equality constraints, which generally corresponds to the OC of steady-state power systems, and $\boldsymbol{u}$ denotes a set of steady-state variables, for instances, $\boldsymbol{V}$ and $\boldsymbol{\theta}$. To solve the problem above, define a Lagrangian function below:

$$\mathcal{L}(\boldsymbol{p},\boldsymbol{u},\boldsymbol{w}) = \zeta(\boldsymbol{p}) + \boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{p},\boldsymbol{u}) \tag{2.6}$$

where $\boldsymbol{w}$ denotes the vector of Lagrangian multipliers. According to the Karush-Kuhn-Tucker's first-order optimality conditions [24], the Jacobian matrix $\boldsymbol{\nabla}_{\boldsymbol{u}}\boldsymbol{\varphi}$ of the Lagrangian function is singular at a loadability limit.

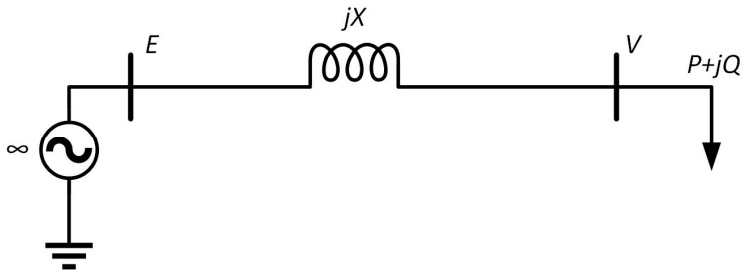

Figure 2.3: Single-Load Infinity-Bus System

Consider a single-load infinity-bus system of Figure 2.3, with a load supplied by an infinite bus through one transmission line. The steady-state equations, which is the equality constraints in the optimization problem of Equation ((2.5)) could be listed as follows:

$$P = \frac{EV}{X} \sin\theta \tag{2.7}$$

$$Q = \frac{EV}{X} \cos\theta - \frac{V^2}{X} \tag{2.8}$$

By normalizing the variables based on the short circuit power, which is E^2/X , it yields

$p = \frac{PX}{E^2}$, $q = \frac{QX}{E^2}$, $v = \frac{V}{E}$,

Rewrite the steady-state equations as follows:

$$p = v \sin\theta \tag{2.9}$$

$$q = v \cos\theta - v^2 \tag{2.10}$$

Given the trigonometric identity equation $v^2\sin^2\theta + v^2\cos^2\theta = v^2$, it yields:

$$p = \sqrt{v^2 - (q + v^2)^2} \tag{2.11}$$

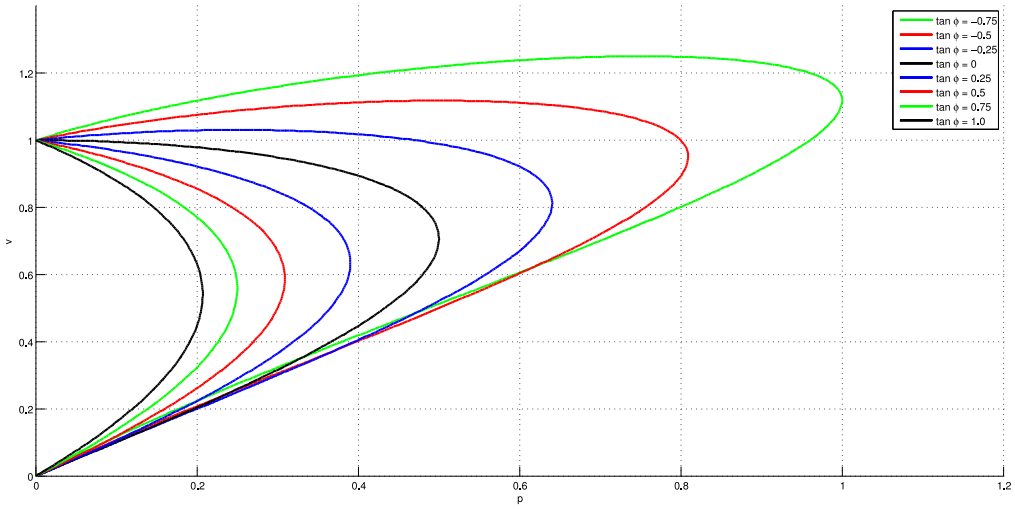

Figure 2.4: p-v Curves

For different settings of the constant power factors on load, $\tan\phi = \frac{P}{Q}$, Figure 2.4 reveals the group of normalized p-v curves.

As Figure 2.4 shows, at a unity power factor load, $\tan\phi = 0$ (given $q = 0$), the (normalized) maximum power is limited at $p_{\max} = 0.5$ when the voltage at load bus drops to a critical value as $v_{critical} = 0.707$[3]. For an increasing lagging power factor $\tan\phi > 0$ (given $q > 0$), which represent more and more inductive loads are connected to the load bus, we can see that the maximum power limit decreases significantly. On the other hand, for leading power factors $\tan\phi < 0$ (given $q < 0$), the loadability limits are higher, and the voltages at these limits are also higher. To achieve a leading power factor, connecting shunt compensation at load bus is a feasible and practical way in order to raise up the loadability limit and to prevent the system from voltage collapse. Therefore, localized shunt compensation and LTC transformers are commonly applied solutions in local and regional voltage control schemes.

For different settings of the constant active power values on load, p, Figure 2.5 reveals the group of normalized v-q curves[4].

From Figure 2.5, for a high loading setting, $p = 1$, the critical voltage is even higher than

[3]The result can be deducted by taking the derivative of Equation (2.11) and setting it to zero

[4]The result can also be achieved by mapping p-v curves sampling points in Figure 2.4 onto the v-q plane.

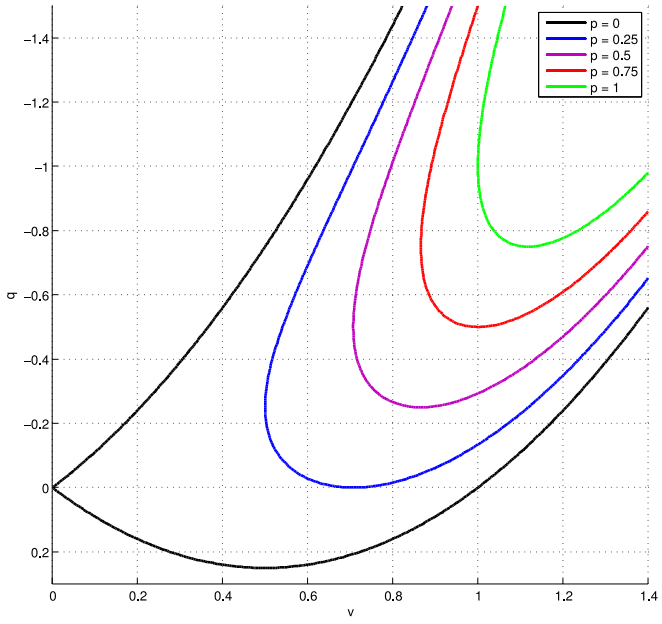

Figure 2.5: v-q Curves

1 p.u., but it also requires reactive power compensation to be sufficient. Similar to the results shown in the p-v curves, it is notable that, localized shunt compensation is critically necessary for heavy loads condition for the sake of secure voltage profiles. The advantage for the v-q curves is revealing the operating conditions as the intersection of the reactive power compensator characteristics and the v-q curves as system characteristics [23]. This advantage helps engineers testing the system robustness regarding to the voltage stability analysis.

Finally, conclude the sources of voltage collapse in the elementary power system model as follows:

- Excessive active power connected to the load buses
- Insufficient reactive compensation at load buses
- Long transmission lines between the generations and the loads
- Low source voltage at the generators buses

Consider a system of n buses, it is assumed that the current operating point $\boldsymbol{p}_0$ is stable in the equilibrium $\boldsymbol{u}_0$. As the load demands vary, the equilibrium point $\boldsymbol{u}$ varies in the state-space system. In search of the steady-state stability limits (SSSL), when the increasing load demand reaches a critical load limit $\boldsymbol{p}_*$, the system can easily lose its stability when encountering any disturbance. Such a set of critical load demands $\boldsymbol{p}_*$ denote the voltage stability margin Σ. In this study, a series of indicators are computed to measure the Euclidean distances between the current stable OC $\boldsymbol{p}_0$ to the set of voltage stability margin Σ to find the shortest distance of $\|\boldsymbol{p}_0 - \boldsymbol{p}_*\|$. In addition, for the cases with minimum distances $d_{\min} < \delta$, the OCs are also regarded as insecure OCs.

$$d_{\min} = \min_{\boldsymbol{p}_i \in \Sigma} \|\boldsymbol{p}_0 - \boldsymbol{p}_i\| \tag{2.12}$$

As discussed in [2, 3], in an unstable OC, no information is left regarding to the nature and locations of the problem. Therefore, it is necessary to create a security boundary as shown in Figure 2.6, which indicates how far away the system is approaching to voltage collapse. Kessel and Glavitsch in [25] presented a methodology using L-indicator to assess the stationary voltage stability. Shukla and Mili in [26] proposed a hierarchical coordinated voltage instability detection scheme by computing the parameter vector at the point of voltage collapse. In this work, the voltage security boundary is approximated by finding the insecure OCs whose Euclidean distances to the secure operation limits are less than a user-defined threshold value δ.

2.3 Online Voltage Security Assessment Framework

To perform an accurate and timely evaluation of VSA for varying OCs, a machine-learning-based online framework is introduced to identify the voltage security status with time-series streaming data in this chapter.

Grid Protection Alliance (GPA) has been providing open-source software platforms for analytics development in power systems since 2010. Open-source Phasor Data Concentrator (*openPDC*) and open-source and Extensible Control and Analytics (*openECA*) are two software development systems designed for streaming real-time synchrophasor data under IEEE

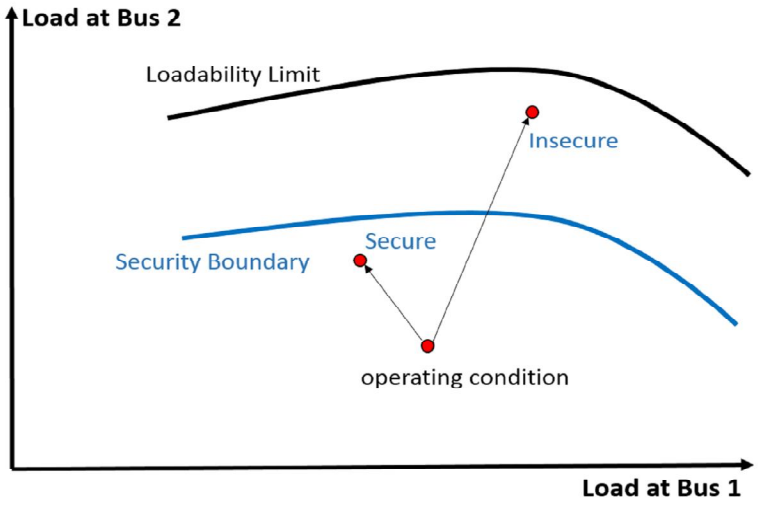

Figure 2.6: Loadability limit and security boundary

C37.118 protocols. OpenECA provides a platform accessing time-series synchrophasor data to practical analytics, such as online voltage security assessment, localized VAR controllers, and regional voltage controllers.

In the online voltage security assessment application using openECA platform, time-series voltage measurements are continuously collected from distributive PMUs and centralized PDCs in a real-time manner. Figure 2.7 reveals our proposed online VSA applications with periodic update of the classification models in five major stages as follows:

1. *Offline dataset preparation*: Based on the knowledge of system model, use the power system simulator software to run power flow calculation and to generate a dataset of voltage measurement frames[5] regarding to different settings on load patterns. By applying network contingencies on each voltage measurement frames, identify the voltage security status and label the voltage measurement frames with "Secure" or "Insecure".

2. *Online supplement dataset preparation*: Voltage measurement frames in a period of time are collected from openECA platform. After applying network contingencies on each frames, these frames are labeled with security status according to contingencies analysis results. These data frames are grouped as updating data instances for periodic update of classifiers.

[5]Each voltage measurement frame represents the voltage profiles at all selected buses after the power flow calculation, and is treated as a data instance for the classifier training.

3. *Offline training of classifiers*: The ensemble of classifiers are initialized by AdaBoost algorithm using offline dataset. Each classifier model in the ensemble model is trained by the weak learner to prevent over-fitting problems. The algorithm also calculates the voting weights for each classifier model.

4. *Periodic update of classifiers*: The proposed scheme for classifier update should be capable to update the ensemble model with the online supplement dataset. This process should be capable to handle large among of input data that openECA platform provides.

5. *VSA applications*: It consists of two main applications: Preventive VSA application and Remedial VSA application. *Preventive VSA* reads each data frames published by openECA and returns a security assessment result of "Secure" or "Insecure" based on the trained/updated classifier model. *Remedial VSA* responses when the preventive VSA returns a "Insecure" label for current openECA input frame, and starts searching for feasible voltage control strategies that bring the system back to a "Secure" operating condition. Remedial VSA incorporates a series of classifiers trained/updated based on the prepared dataset, and the result of each classifier is representing the predicted security status if the related control strategy is executed according to current voltage profiles.

With ensemble decision trees trained offline using AdaBoost, it is ready to apply the initial classifiers to the online VSA applications in Figure 2.7, whose workflow is shown as the solid line. However, a robust VSA application should be able to adapt to different operating conditions by updating the classifiers (shown as the dash line in the figure). The following chapter proposes an incremental tree induction method to update the trained ensemble decision trees by testing the Hoeffding bound to decide whether it is necessary or not to update the models when collecting data from openECA platform.

2.4 Summary

This chapter reveals the use of decision-tree models in the online voltage security assessment. At the beginning, a common induction process of decision-tree model is illustrated. To develop a robust classification model, one of the ensemble learning methods is presented.

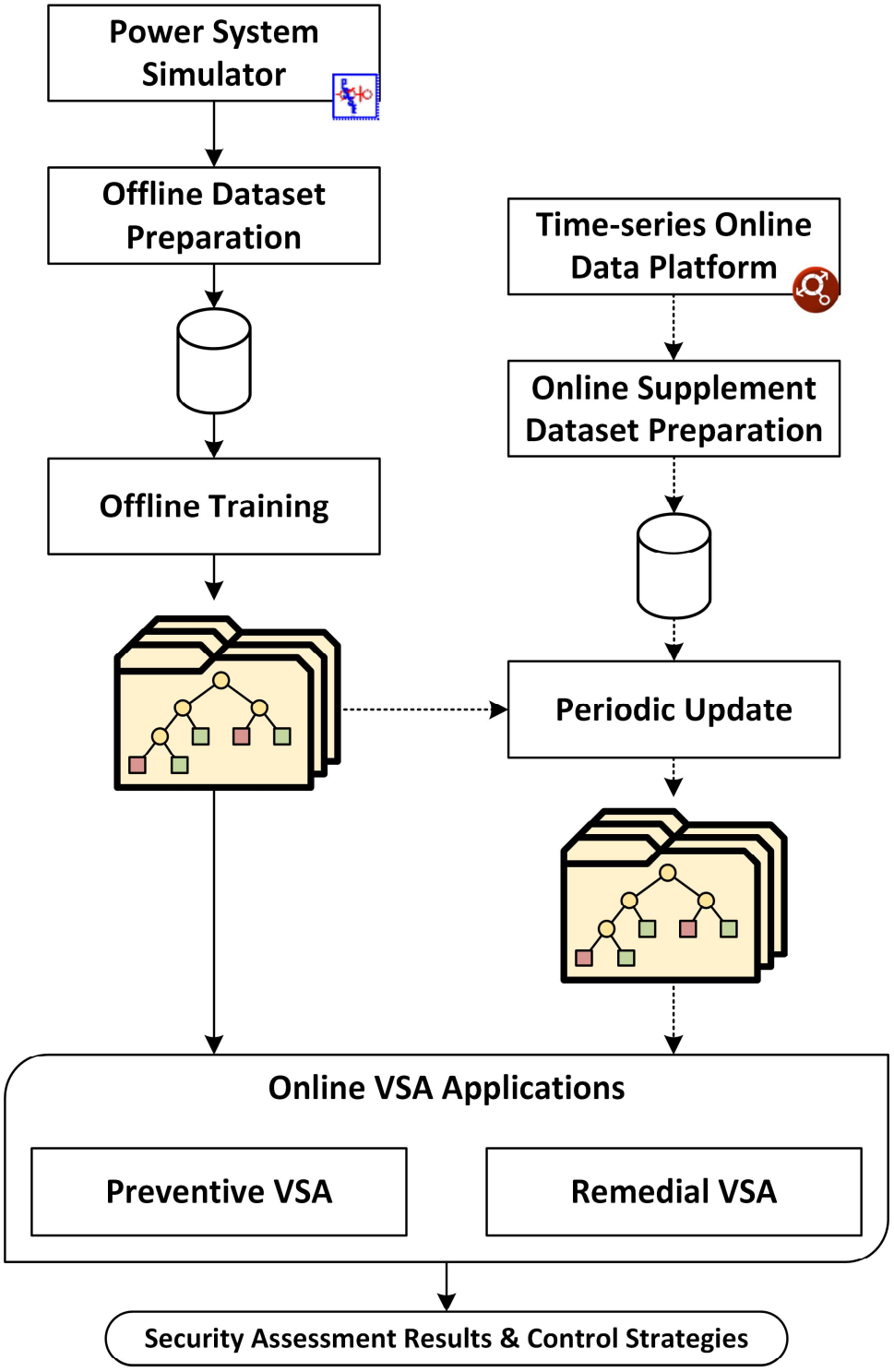

Figure 2.7: The Workflow of Online Voltage Security Assessment Framework

Subsequently, the voltage security problem and related analysis methodology are discussed. In order to perform online VSA in a real-time manner, the assessment framework incorporated with offline training and online periodic update is elaborated. The AdaBoost algorithm is applied in the offline training stage to initialize the ensemble model of decision trees. To adapt the ensemble model according to different configurations on power systems, a new method for periodic update will be presented in the following chapter.

Chapter 3

Very Fast Decision Trees System and Its Application in Online Voltage Security Assessment

For a common decision tree, each terminal node contains a number of training instances passed down from the root. During the tree induction process, to decide whether a terminal node should be split to a new generated attribute node or not depends on the criterion of heuristic measurement selecting the best attributes in the decision tree as illustrated in Chapter 2.1.1. Alternatively in [27], Domingos and Hulten applied *Hoeffding bound*, also known as additive *Chernoff bound*, to serve as a role of the statistical condition about how many samples should be seen before executing a split under a high-speed data streams environment.

3.1 Hoeffding Bound

Let Z_1, Z_2, ..., Z_n be independent bounded random variables within the range of $[a, b]$. According to the *Hoeffding Inequality*, for $\forall t > 0$,

$$\Pr\left\{\left|\frac{1}{n}\sum_{i=1}^{n}\left(Z_i-\bar{Z}\right)\right|\geq t\right\}\leq 2\exp\left(-\frac{2nt^2}{B^2}\right) \tag{3.1}$$

where $\bar{Z}$ is the estimated mean of the random variable Z, $\bar{Z}=(Z_1+Z_2+\cdots+Z_n)/n$. And B denotes the range of existing random variables, which equals to $|b-a|$. Say that with a confidence of $1-\delta$, the true mean μ is at least $\bar{Z}-\epsilon$, i.e.

$$\Pr\left(\mu\leq\bar{Z}-\epsilon_H\right)\geq 1-\delta \tag{3.2}$$

By the Equations (3.1) and (3.2), the Hoeffding bound is defined as follows:

$$\epsilon_H=\sqrt{\frac{B^2\ln\left(1/\delta\right)}{2n}} \tag{3.3}$$

3.2 Hoeffding Trees Algorithm and VFDT System

3.2.1 Induction of A Hoeffding Tree

From Equation (3.2), by replacing the random variable Z with the heuristic measurement value, notice that with confidence $1-\delta$, we have received sufficient information to split a terminal node into branches if the difference of the highest two heuristic measurements $\Delta\bar{G}=\bar{G}\left(X_a\right)-\bar{G}\left(X_b\right)$ is larger than the Hoeffding bound ϵ_H. This is the primary principle for the incremental tree induction process of Hoeffding trees (HTs), whose pseudo-code is shown in Algorithm 1.

Algorithm 1 demonstrates the Hoeffding tree induction process. The inputs are the decision-tree model DT, training dataset $\mathcal{D}$, attribute set $\boldsymbol{X}$, heuristic measurement method G, split confidence δ, and grace period $n_{\min}$ respectively. Line 2 initializes the model with a root. However, this initialization is optional; in Chapter 2.3, the decision-tree models are generated offline using AdaBoost algorithm. Line 3 20 is the iterative process of inducing a HT for every single training instance. In line 4, the model DT sorts it to a terminal node according

to its attribute values and the attribute conditions in DT; then, the training instance is stored at the terminal node holding sufficient statistics. The code from line 7 18 is executed periodically according to the grace period $n_{\min}$. This periodic process speeds up the training by postponing the data since recalculating the heuristic measurement for all data is time-consuming and re-evaluating is unnecessary for each instance. Line 8 17 restates the core principle of inducing HT. It is a key feature in the condition of line 13 that involves the "null" attribute $X_\emptyset$ considering the benefit of not splitting. The "null" attribute could be regarded as a pre-pruning process that prevents splitting unless an attribute is sufficiently better than $X_\emptyset$ according to Hoeffding bound. In line 13, τ is a user-defined threshold called "tie-breaking" considering the situation of two or more attributes causing a low Hoeffding bound, and the setting of τ allows continuing the induction in this case.

Algorithm 1 Hoeffding Tree Induction Algorithm

1: **procedure** HoeffdingTree($DT, \mathcal{D}, \boldsymbol{X}, G, \delta, n_{\min}$)
2: **initialize** DT with a root **if** the decision tree is empty
3: **for all** training instances $(\boldsymbol{x}_k, y_k) \in \mathcal{D}$ **do**
4: Sort each instance into the terminal node l using DT
5: Update sufficient statistics in l
6: Increment n_l, the number of instances seen by l
7: **if** n_l mod $n_{\min} = 0$ **and** instances seen by l are not the same class **then**
8: Compute $\bar{G}_l(X_i)$ for each attribute $X_i \in \boldsymbol{X}$
9: Let X_a be the attribute with highest $\bar{G}_l$
10: Let X_b be the attribute with second-highest $\bar{G}_l$
11: Compute the Hoeffding bound in equation (3.3)
12: **end if**
13: **if** $X_a \neq X_\emptyset$ **and** $(\bar{G}_l(X_a) - \bar{G}_l(X_b) > \epsilon_H$ **or** $\epsilon_H < \tau)$ **then**
14: Replace the terminal node l with an attribute node splitting on X_a
15: **for all** branches under the new attribute node **do**
16: Add a new terminal node with initialized sufficient statistics
17: **end for**
18: **end if**
19: **return** DT
20: **end for**
21: **end procedure**

The induction process of a Hoeffding three shows its merit of independence from the distri-

bution from training data. This property of HTs benefits the application of online learning algorithm with infinite data instances. The trade-off resides in the rule of splitting becomes more conservative [27] compared with traditional distribution-dependent algorithms, which will have to capture more data instances to reach the confidence $1 - \delta$. The parameters in the algorithm above are detailed in the next section.

3.2.2 The VFDT System

In [27], Domingos and Hulten refer the implementation of online decision-tree learning system based on Hoeffding tree algorithm as Very Fast Decision Tree (VFDT) system. VFDT system is capable to apply different kinds of heuristic measurement such as Information Gain and Gini index to find the best attributes when building decision-tree models. Its efficiency has been studied in a distributed wireless sensor network application [14]. In this chapter, five essential factors are studied in the Algorithm 1 that influences the split of a HT during the tree induction process.

Split Confidence

As introduced in Chapter 3.1, the definition of "Hoeffding bound" applies a parameter of confidence $1 - \delta$ that tells the true mean μ is at least $\bar{Z} - \epsilon_H$. Figure 3.1 illustrates a series of different settings on δ with the same range value $B = 1$. The blue solid line shows the Hoeffding bound with a setting of $\delta = 10^{-3}$, the red dot line with $\delta = 10^{-5}$, and the green dash line with $\delta = 10^{-7}$. From the graph, we can notice that with lower confidence, the Hoeffding bound drops rapidly with the first thousand data instances. We also notice that for increasing confidence, it requires a larger number of data instances n to achieve the same Hoeffding bound level as desired. As a result, on configuring the value of δ, the higher the confidence at each attribute nodes results in more data instances should be collected to examine the boundary, which subsequently affects whether this node should be split or not.

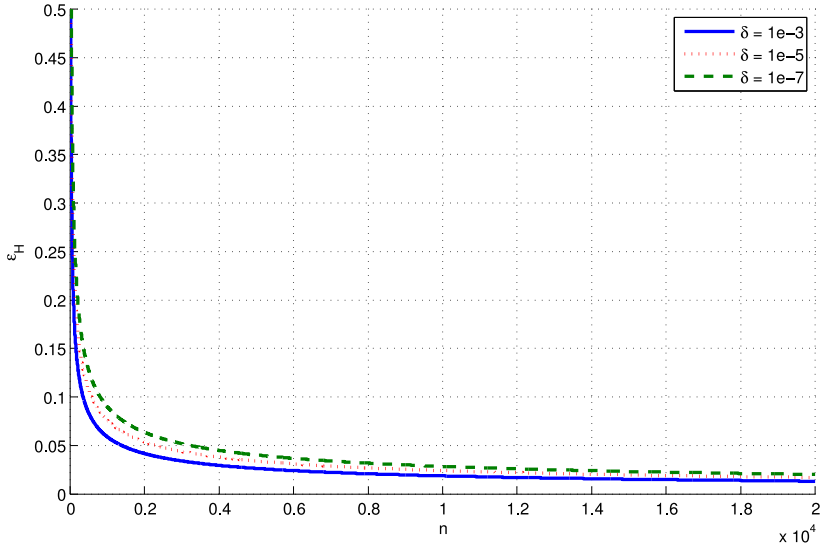

Figure 3.1: Hoeffding bounds on different settings of δ

Grace Period

Except for the confidence value $1 - \delta$, there is another parameter plays a critical role in the induction of a Hoeffding tree, which is called the *Grace Period*, as denoted as $n_{\min}$.

On the use of high-speed streaming data, it is costly to examine every data instance to tell whether the classification model should be updated or not. Usually, one single data instance has trivial influence on the accuracy of the classifier. To achieve the efficiency of the update process, the VFDT system provides an execute query of delay on streaming dataflow by setting the grace period. In Algorithm 1, the condition $n_l \bmod n_{\min} = 0$ is examined every time a new data instance is collected and before testing the Hoeffding bound. This mod condition indicates that we can execute the process at least every $n_{\min}$ instances dropped on this terminal node. In other words, we don't need to examine the Hoeffding bound of a terminal node until the number of instances reaches a multiple of grace period, while the terminal node maybe impure. The effect of grace period has been discussed in [13, 28] with a dataset whose number of data instances is in a million-class. The grace period decides the time to split an impure node, and alleviates the computation burden when new data keeps flowing in frequently.

Heuristic Measurement

Heuristic Measurement, guides which splitting rule to be used when choosing the best split. The higher scores in heuristic measurement indicates the better the splitting rule is capable to tell classes apart. For example, both Quinlan's ID3 algorithm [11] and it successor C4.5 algorithm utilize the normalized information gain as the heuristic measurement during training. However, the VFDT system is more flexible on choosing methods of heuristic measurement and allows the use of different kinds of heuristic measurement depending on the type of classification tasks.

In this paper, Gini index discussed in Chapter 2.1.1 is used as the heuristic measurement due to its advantages in two-class classification tasks. The usage of Gini index is also tested in some other machine learning algorithms, for instance, *CART* [29]. In the applications, pick the attribute of the highest Gini index value at each attribute node to achieve the best split. Note that recalculating the heuristic measurement happens frequently when developing a Hoeffding tree. With an adequate setting on split confidence $1-\delta$ and grace period $n_{\min}$, the recalculating frequency is reduced, in which the tree induction process spends most of time. If a larger value is set to split confidence, the computation of heuristic measurement spends more time to collect data instances and find the suitable attribute due to the strict confidence condition.

Hoeffding Tie Threshold

To further enhance the robustness of VFDT system, Domingos and Hulten have also introduced the *Hoeffding tie threshold* τ considering the circumstance that two or more attributes may have close values in heuristic measurement. For power systems, the voltage magnitude measurements are quite similar on two close buses when the electrical distance is short between buses. Hence, such circumstances can easily arise for the VSA applications. The condition $\bar{G}_l(X_a) - \bar{G}_l(X_b) > \epsilon_H$ is going to be hard to achieve. The tree induction process will be halted consequently, because the two attributes yield small difference in heuristic measurement. To eliminate the effects of these potential situations, an alternative condition is introduced to resume the induction process, which is $\epsilon_H < \tau$ shown in Algorithm 1. When the Hoeffding bound of two or more attributes drops to a sufficiently small value, the tie threshold τ allows the algorithm to continue on incremental induction by splitting

on the best one, although the second-best option is approximate. This action is referred as *Tie-breaking* in [28].

Null Attribute

Notice that, in the Algorithm 1, a *Null Attribute* $X_\emptyset$ is referred as a "pre-pruning" process when inducing the Hoeffding tree. *Pruning*, represents a process of trimming the terminal nodes in a decision tree model into a smaller scale. This process discards some of terminal nodes, and replaces the attribute nodes with terminal nodes assigned with the class of higher probabilities. Generally speaking, it is necessary to prune a decision tree when the models, for example, a fully-grown CART, is so complex that it poses risks of over-fitting problems. And such process is commonly applied after a decision tree is fully trained, as known as *post-pruning*.

Accordingly, VFDT system offers a way to determine whether a node should be pruned or not during the induction instead of pruning after the whole tree is structured. It introduces a null attribute to consider not to split a node. For a non-splitting node, it predicts the most frequent class dropped to this node. Just like a normal attribute, the heuristic measurement of a null attribute accounts for the impurity of a node. An arbitrary node is prevented from splitting until the impurity of this node increases to a certain level, then a useful split is executed. Therefore, this introduced feature benefits the efficiency during inducing a Hoeffding tree.

3.3 Reweighting of Ensemble Hoeffding Trees

As an incremental induction process of decision trees, the VFDT system provides flexibility in the initialization of Hoeffding tree. It requires the tree structures to be updated with new data instances regardless of the method used to generate the initial models. Therefore, the VFDT is capable to handle the ensemble learners with multiple trained decision trees. In Chapter 2.1.2, the ensemble models learnt by AdaBoost algorithm is discussed. Given trained ensemble decision trees models, each decision tree can be further induced incrementally using the same induction process of Hoeffding trees.

In addition to the updated tree structures, the voting weights for all decision trees are also required to be recalculated due to new updating data instances incoming. In [30], Bauer and Kohavi discuss the empirical usages of different ensemble learning methods regarding to the voting techniques, including Bagging, Boosting, and their variants. In this study, the reweighting method for modified classifiers based on the exponential loss function is adopted. In this case, the voting weight of each Hoeffding tree in the ensemble model is re-calculated as follows:

$$\alpha_l^* = \frac{1}{2} \ln \left(\frac{1 - {e_l}^*}{{e_l}^*} \right) \tag{3.4}$$

In Equation (3.4), the * sign represents the updated parameters after new data instances are dropped throughout the tree model. With re-weighted ensemble model, we are allowed to continue to use the ensemble model for a specific classification task.

3.4 VFDT System applied in Online Voltage Security Assessment

A time-series data publication platform is capable to publish voltage measurements with time-stamps through capturing measurement from distributive PMUs installed in power systems in a real-time manner. As already shown in Figure 2.7, the measurement data will continue to be labeled[1] as "Secure" or "Insecure", before updating the classification model in the online VSA applications.

Figure 3.2 reveals the proposed update stage in the online VSA applications in order to update the classification models prepared in advance. It consists of three major steps as follows:

1. The existing ensemble model of DTs are initialized by AdaBoost algorithm using offline training dataset. Each of DT in the ensemble model is a short tree stump learned by the weak learner to prevent the model from over-fitting problems. The algorithm also

[1] In this study, this process as is referred as *Online Supplement Dataset Preparation*.

generates the voting weights for each DT model to achieve a weighted sum of the ensemble model.

2. The proposed scheme is capable to update the ensemble model with one new OC at one time instead of re-training the ensemble model with the online supplement dataset. During the periodic update stage, each DT model's structure is further induced incrementally based on the Hoeffding bound, which guarantees that the inductions are necessary with sufficient statistics to improve the performance. By using VFDT system, it relieves the computation burden in the tree induction process when adapting the ensemble model to changed OCs.

3. Then, the voting weights α_l of all updated DTs in the ensemble model are re-calculated according to the training errors of each modified DT.

4. Online VSA applications utilize the modified ensemble model to assess the security status of current OC again according to the voltage measurements collected in real-time.

3.5 Summary

This chapter illustrates how the Hoeffding bound controls the tree induction process from scratch or from a moderately-developed classification model. A set of configured parameters for the VFDT system is explained. To realize the periodic update stage in online VSA applications, the VFDT system is adopted to update the ensemble model developed as explained in the previous chapter, followed by a reweighting process. The online voltage security assessment framework has been completed heretofore with offline training stage and online periodic update stage.

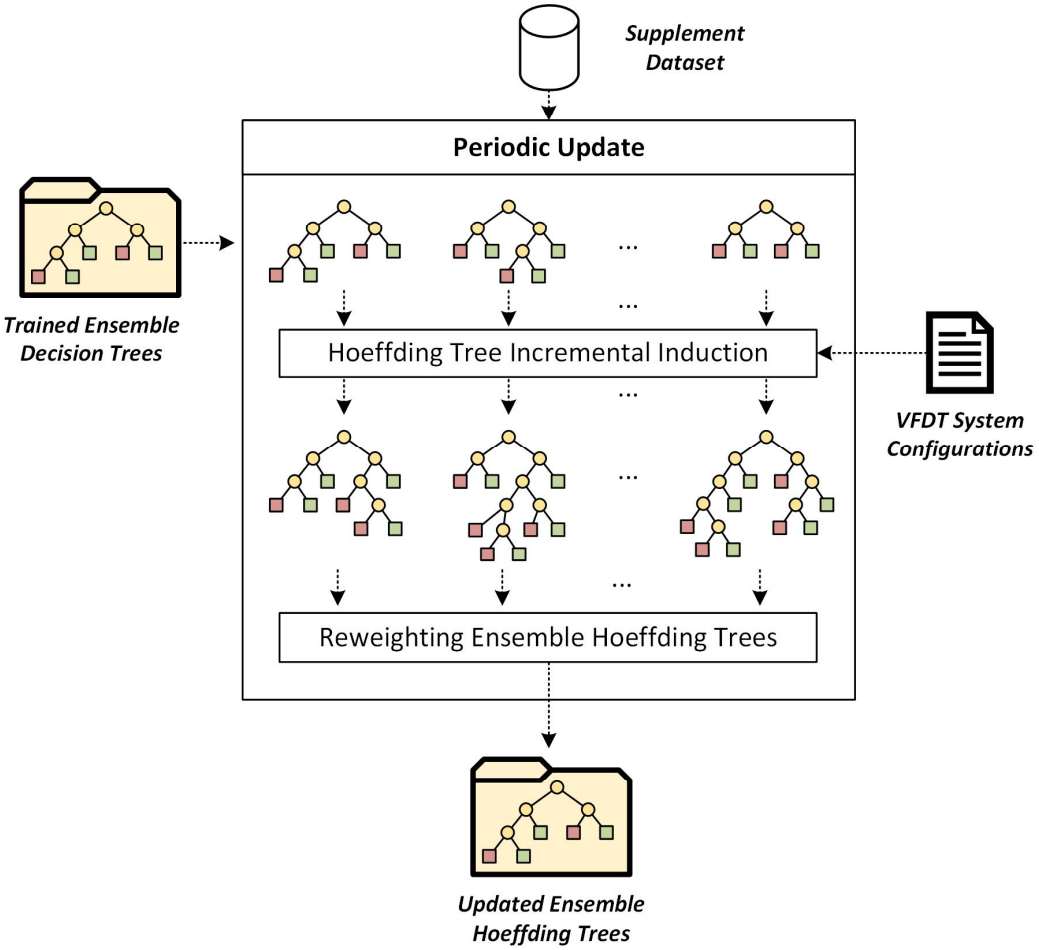

Figure 3.2: Periodic Update of Online VSA Applications using VFDT System

Chapter 4

Case Study and Implementation on openECA Platform

4.1 Preparation of Training and Updating Database of OCs

The proposed scheme of VSA is evaluated using IEEE 118-bus standard system shown in Figure 4.1. The system is divided into 3 areas as suggested in [31]. The system division is conducted based on two levels of decomposition. The first level provides a cluster analysis that determines the appropriate number of divided areas. The second level decomposition assigns the remaining buses to the areas identified in the first level according to proximity analysis. In each area, load buses are assumed to have the same loading variation pattern, and the load is randomly scaled up and down in the same percentage to generate different OCs. Thus, the productions of generators among the system are dispatched within their areas by following the zone-based loading pattern. Voltage magnitude measurements at all buses are selected for the generation of learning database. Overall, 20,245 OCs are generated by scaling up the load demands from 100% to 150% of their base case value for each area. VSA is implemented to determine whether the OC is "Secure" or "Insecure". In this paper, it is also assumed that the maximum loadability and the security boundary mentioned in Chapter 2.2 are overlapping. The minimum distance threshold to determine the insecure OCs is highly depending on the security requirement for the system. A smaller the threshold indicates the

Table 4.1: Number of Secure/Insecure OC Samples for Case Study

Label	Training	Initial Test	Update	Post-test
Secure	9948	6615	1752	1164
Insecure	2199	1483	1450	967

system can operate closer to the security boundary. In this study, the threshold value is selected as 15 which provides a clear boundary between the normal OCs and the unstable OCs. For all the secure and insecure OCs, 60% of them are used for training, and the rest are reserved for testing the initial ensemble model as shown in Table 4.1. The initial ensemble of decision-tree is trained offline using AdaBoost algorithm mentioned in Chapter 2.1.2 with $L = 30$ weak learners.

4.2 Updating Performance Test Using VFDT System

To test the robustness of proposed methodology using VFDT system for updating, the transmission line (15, 33) is tripped on the IEEE 118-bus system. As shown in Table 4.1, new updating data and test data are created using the same OCs generation method in previous section with a different system topology. Within these new OCs, 3,202 of them are ready to be deployed to update the initial ensemble model, and 2,131 are reserved as the post-test database for the modified ensemble model. The computation is run in Python 3.5.2 using PyCharm Community Edition on an Intel® Core™ i5 @2.20GHz computer.

Figure 4.2 and Figure 4.3 show the computation time and testing error during the updating process for a DT in the ensemble model. It is observed that, with more and more instances incorporated, the default method takes more time to rebuild the DT model because of the increasingly growing database. By contrast, the Hoeffding tree algorithm saves considerable computation time by only inducing the leaf nodes with sufficient statistics, instead of growing the DT from scratch. For the proposed method, the computation time for each update is only related to the size of update dataset at one time. The step differences of red line in Figure 4.3 reveals the changes in the DT model. Despite that the Hoeffding tree algorithm is relatively conservative to apply the splits in leaf nodes, the testing errors is still able to

decrease to the same level as the default DT.

4.3 Online VSA Applications implemented with openECA

4.3.1 Introduction of openECA Platform

OpenECA is an open-source Extensible Control and Analytics platform for synchrophasor data collected from phasor measurement units (PMUs) and phasor data concentrators (PDCs). It contains a robust database for different types of synchrophasor-based measurements captured from different devices. Figure 4.4 reveals the relationship of time-series data structure established for openECA platform. For a generated openECA project of analytic, it virtually establishes an internal phasor data concentrator processing interested measurements, realizes the functionality according to the developed algorithm with the use of measurement values, and publishes related output signals to the openECA server in the end. For example, for an openECA-developed linear state estimator [4], it utilizes available measurements provided by synchrophasors to analyze and estimate the operating state of power systems in real-time use[1].

The deployment of an analytic project using openECA contains the following five major steps:

1. Establish the data structure for expected measurements
2. Establish the data mappings of measurements with IDs assigned by the internal historians
3. Generate a code template and develop the analytic within the generated "Algorithm" file
4. Test and debug the completeness and functionality of developed analytic
5. Create an installable file for analytic to be released and deployed at substations or control centers depending on the analytic's functionality

[1] *https://github.com/kdjones/openLSE*

4.3.2 Measurements Data Structure and Data Mappings Setup

Manage Data Structure

There are six fields in the data structure[2] required to be defined for each measurement by using SQL query scripts:

- *SignalID*: a globally unique identifier (GUID) generated by openECA when a measurement is created
- *PointTag*: a string of primary key relating to the particular measurement in the data table, which is the unique record. This field is commonly designated in a form of "DEVICEPREFIX!DEVICESUFFIX:MEASUREMENTNAME"
- *SignalTypeID*: an integer used to distinguish the measurement types, including magnitude and angle of voltage and current, frequency, rate of change of frequency (df/dt), calculated value, digital value, etc.
- *SignalRefernce*: a string that facilitates other internal or external connections to search and access such measurement
- *Description*: a string of text that describes the measurement
- *Enabled*: a binary value that determines whether the measurement is enabled in openECA platform

For online VSA applications, measurements are created for all bus voltage magnitudes, the capacitor banks' circuit breakers states, and their closing and tripping signals as listed in Table 4.2. Additionally, in the last column, *AssignedID* is an identifier automatically assigned by Primary Phasor Archive (PPA) as the default historian. *ResetSignal* is created for initializing and abort the testing environment for online VSA applications. And *LoadIncrementPercentage* is simulating as the load patterns on the system, which is by a CSV file using the "CsvInputAdapter" developed within openECA platform.

[2]A complete data structure of each measurement includes SignalID, HistorianID, PointID, DeviceID, PointTag, AlternateTag, SignalTypeID, PhasorSourceIndex, SignalReference, Adder, Multiplier, Description, Enabled. Detailed settings of all fields can refer to *https://github.com/GridProtectionAlliance/openPDC*.

Table 4.2: Measurements Data Structure for Online VSA Applications

Name	DataType	PointTag	SignalType	SignalReference	AssignedID
ResetSignal	short	SS_118:RESET	DIGI	SS118-RESET	PPA:62
LoadIncrementPercentage	double	SS_118:LOADINCRE	DIGI	SS118-LOADINCRE	PPA:41
ActSnB34Close	short	SS_118:ACTSNB34CLOSE	DIGI	SS118-ACTSNB34CLOSE	PPA:63
ActSnB44Close	short	SS_118:ACTSNB44CLOSE	DIGI	SS118-ACTSNB44CLOSE	PPA:64
ActSnB45Close	short	SS_118:ACTSNB45CLOSE	DIGI	SS118-ACTSNB45CLOSE	PPA:65
ActSnB48Close	short	SS_118:ACTSNB48CLOSE	DIGI	SS118-ACTSNB48CLOSE	PPA:66
ActSnB74Close	short	SS_118:ACTSNB74CLOSE	DIGI	SS118-ACTSNB74CLOSE	PPA:67
ActSnB105Close	short	SS_118:ACTSNB105CLOSE	DIGI	SS118-ACTSNB105CLOSE	PPA:68
ActSnB34Trip	short	SS_118:ACTSNB34TRIP	DIGI	SS118-ACTSNB34TRIP	PPA:69
ActSnB44Trip	short	SS_118:ACTSNB44TRIP	DIGI	SS118-ACTSNB44TRIP	PPA:70
ActSnB45Trip	short	SS_118:ACTSNB45TRIP	DIGI	SS118-ACTSNB45TRIP	PPA:71
ActSnB48Trip	short	SS_118:ACTSNB48TRIP	DIGI	SS118-ACTSNB48TRIP	PPA:72
ActSnB74Trip	short	SS_118:ACTSNB74TRIP	DIGI	SS118-ACTSNB74TRIP	PPA:73
ActSnB105Trip	short	SS_118:ACTSNB105TRIP	DIGI	SS118-ACTSNB105TRIP	PPA:74
StateSnB34CapBkrV	short	SS_118:STATESNB34CAPBKRV	DIGI	SS118-STATESNB34CAPBKRV	PPA:75
StateSnB44CapBkrV	short	SS_118:STATESNB44CAPBKRV	DIGI	SS118-STATESNB44CAPBKRV	PPA:76
StateSnB45CapBkrV	short	SS_118:STATESNB45CAPBKRV	DIGI	SS118-STATESNB45CAPBKRV	PPA:77
StateSnB48CapBkrV	short	SS_118:STATESNB48CAPBKRV	DIGI	SS118-STATESNB48CAPBKRV	PPA:78
StateSnB74CapBkrV	short	SS_118:STATESNB74CAPBKRV	DIGI	SS118-STATESNB74CAPBKRV	PPA:79
StateSnB105CapBkrV	short	SS_118:STATESNB105CAPBKRV	DIGI	SS118-STATESNB105CAPBKRV	PPA:80
MeasB1VoltV	double	SS_118:MEASB1VOLTV	VPHM	SS118-MEASB1VOLTV	PPA:81
MeasB2VoltV	double	SS_118:MEASB2VOLTV	VPHM	SS118-MEASB2VOLTV	PPA:82
MeasB3VoltV	double	SS_118:MEASB3VOLTV	VPHM	SS118-MEASB3VOLTV	PPA:83
...	...	...	...	...	...
MeasB118VoltV	double	SS_118:MEASB118VOLTV	VPHM	SS118-MEASB118VOLTV	PPA:198

The complete SQL scripts used to construct the database's structure is detailed in Appendix B.1.

Manage Data Mappings

In Table 4.2, a data structure for necessary measurements is established. Now these measurements are ready to connect to analytic's variables by assigning PPA IDs shown in the last column. According to the functionality of analytics, some measurements act as input data, and other measurements act as output data. Given assigned with the same PPA IDs for all measurement, openECA provides data sharing between different analytics. As mentioned later in this paper, a testbed analytic called *Shadow System Simulator* is developed to simulate the control results of certain control actions provided by online VSA. Shadow System and online VSA applications respectively uses these measurements in a different way, as shown in Table 4.3. OpenECA configures data mappings for analytics in basic ASCII files detailed in Appendix B.2.

4.3.3 Configurations of Analytics generated by openECA Client

- *AllowPreemptivePublishing*: a boolean configuration of openECA analytic. If "True", the internal concentrator is allowed to publish for analytic's use when all measurement data arrives within a user-defined "LagTime". The setting of this parameter should be set as "True" unless the measurement data is not able to arrive routinely as expected.
- *FramesPerSecond*: defines the time resolution for measurement data[3] published by internal concentrator. Then, the concentrator is capable of assigning timestamps with specific time intervals to measurement data. For example, given a synchrophasor sending data at a rate of 30 frames per second, the timestamps of data are assigned with 33.333 ms interval.
- *LagTime*: defines the maximum time for an openECA analytic to wait for the new measurement data from concentrator before algorithm execution.

[3]The measurement data includes input data from openECA, and output data, which will be received by openECA after algorithm execution through the internal concentrator.

Table 4.3: Manage Input/Output Data Mappings in openECA

Name	AssignedID	For Shadow System	For Online VSA
ResetSignal	PPA:62	Input	N/A
LoadIncrementPercentage	PPA:41	Input	N/A
ActSnB34Close	PPA:63	Input	Output
ActSnB44Close	PPA:64	Input	Output
ActSnB45Close	PPA:65	Input	Output
ActSnB48Close	PPA:66	Input	Output
ActSnB74Close	PPA:67	Input	Output
ActSnB105Close	PPA:68	Input	Output
ActSnB34Trip	PPA:69	Input	Output
ActSnB44Trip	PPA:70	Input	Output
ActSnB45Trip	PPA:71	Input	Output
ActSnB48Trip	PPA:72	Input	Output
ActSnB74Trip	PPA:73	Input	Output
ActSnB105Trip	PPA:74	Input	Output
StateSnB34CapBkrV	PPA:75	Input	Output
StateSnB44CapBkrV	PPA:76	Input	Output
StateSnB45CapBkrV	PPA:77	Input	Output
StateSnB48CapBkrV	PPA:78	Input	Output
StateSnB74CapBkrV	PPA:79	Input	Output
StateSnB105CapBkrV	PPA:80	Input	Output
MeasB1VoltV	PPA:81	Input	Output
MeasB2VoltV	PPA:82	Input	Output
MeasB3VoltV	PPA:83	Input	Output
...	...	...	...
MeasB118VoltV	PPA:198	Input	Output

- *LeadTime*: defines the maximum time that the concentrator allows when the incoming measurement data is assigned with future timestamps. This configuration is defined due to the difference between the clock of local machine and the GPS synchronous time. It prevents the concentrator from discarding "invalid" measurement data with future timestamps within a specified tolerance.

4.3.4 Shadow System Simulator Testbed

Shadow System Simulator is an openECA analytic deployed as a Windows service written in C# scripts. It interacts with the power flow engine powered by Siemens PSS®E to provide a set of time-series measurement signals of all of the values collected from the static power flow calculation. Shadow System also can receive control signals from other openECA analytics like Localized Volt-VAR Control and Regional Voltage Control for raising or lowering LTCs, opening or closing circuit breakers, changing load patterns.

In order to test the functionality of online VSA applications, the Shadow System is deployed as a testbed simulating the actual real-time operation of IEEE 118-bus system. Shadow System is capable of streaming system measurement data to openECA Server node at a rate of 1 frame per second. The frame rate is so limited primarily due to the execution time that PSS®E needs to finish the power flow calculation. A better processor on the server where Shadow System is located could perform in a higher frame rate with a lower average publication time. On the other hand, the online VSA applications return a set of control signals according to the decision-tree-based classifier to openECA Server node. Using the identical channels for these signals, openECA platform will provide Shadow System with control signals to construct a complete control loop. Figure 4.5 demonstrates the dataflow between different openECA analytics.

Figure 4.6 reveals a list of critical metrics during the execution of Shadow System Simulator on openECA platform. In the figure, we need to be aware of the some of the statuses when running Shadow System as a background Windows service[4]. In the last few rows in the list, we can notice that the analytic establishes a phasor data concentrator to provide measure-

[4]In this work, the openECA-based analytic Shadow System is running as a service when testing the functionality of other control analytics deployed by NSSM. NSSM is a service manager program that facilitates the background service deployment of applications. Please refer to *https://nssm.cc/* for further information.

ment value at 1 frame per second as designed (shown as "Timer reference count"). Besides, the "Defined frame rate" also verifies the setting of this openECA analytic. The status of "Average publication time" shows that the algorithm spends around 600 ms during each publication of the concentrator. This is totally acceptable because this value is sufficiently smaller than 1000 ms, the time interval which the concentrator should publish measurements with. However, we notice that the "Estimated mean frame rate" is shown as 0.93 frame per second. A rational explanation for this would be, when starting the Shadow System service, the compilation process prevent the concentrator from immediate publication of data. This phenomenon is acceptable since none of measurement expected to be published is discarded, as shown as "Last discarded measurement" in the status metrics.

In addition, *Grafana Dashboard*[5] is a back-office visualization system designed to view time-series data used to support processing analytics. By creating an internal subscription to openECA, it monitors the real-time values of voltage magnitudes and the corresponding states of circuit breakers of capacitor banks as shown in Figure 4.7.

4.4 Deployment of Online Voltage Security Assessment Analytic

As shown in Figure 4.5, in order to establish the closed-loop architecture to test the functionality of proposed online voltage security assessment, the Shadow System Simulator plays a role as a testbed simulating the operation of the IEEE 118-bus system. Several configurations are required to develop the cooperation between two openECA-based analytics. The meaning for these settings has been illustrated in Chapter 4.3.3, and these settings are specified in Table 4.4.

An explanation for these settings is demonstrated as follows:

1. For both analytics, it is assumed that the power flow calculation results and the outputs from online VSA including the assessment result and the feasible control strategy if needed, are able to arrive routinely at 1 frame per second as designed. Therefore, we allow preemptive publishing for both analytics as default.

[5] *https://grafana.com/*

Table 4.4: Configurations for openECA-Based Closed-Loop Control Architecture

Setting	Shadow System Simulator	Online VSA
AllowPreemptive Publishing	True	True
FramesPerSecond	1	1
LagTime (in seconds)	1	2
LeadTime (in seconds)	4	1

2. The "LagTime" for Shadow System depends on the publication time needed in each execution of the internal concentrator. To prevent the incompleteness of measurement data for online VSA usage, the lag time waiting for measurement is set to be larger than the publication time of Shadow System analytic.

3. Despites that the online VSA analytic takes much less time than the Shadow System does, the "LagTime" for online VSA is set to be larger than that fo Shadow System. The reason is to ensure execution sequence for both analytics. For each frame being processed, we firstly run Shadow System to collect the power flow results as measurements to be published to openECA. Then in the online VSA analytic, we exploit these measurement to assess the voltage security status of this frame. This setting synchronizes two analytics which may start at different time so that the online VSA can provide assessment according to the measurement values at this exact time frame.

4. If necessary, the online VSA analytic provides control signals to the Shadow System when an insecure voltage profile is detected. To complete the control architecture, a modification on control signals is set to be 3 seconds (larger than the "LagTime" of online VSA) more than the key frame's timestamp, so that the Shadow System is capable of controlling system devices like capacitor banks in the following frames.

5. According to the setting of updated timestamp, the Shadow System need to set "LeadTime" as 4 seconds in order to receive control signals at a future time.

So far, we have completed the control loop as power flow calculation and measurement publication (Shadow System), voltage security assessment (online VSA), decisions for control strategy (online VSA), and execution of control signals (Shadow System) eventually. Figure 4.8 visualizes the simulation result of the closed-loop control performance of Shadow

System and online VSA applications. A heavily increasing load condition is simulated to approach the loadability margin on the system. In this figure, we notice with load increasing, a control action of closing capacitor banks at bus 44, 45 and 105 is conducted to the system due to detected insecure voltage security status. By closing capacitor banks, voltages at certain buses are escalated, and the system returns to a secure operating condition accordingly. This simulation is running in loops. By the end of each loop, the load demand decreases to the initial value, and all capacitor banks are tripped to return to the initial circumstance.

4.5 Summary

At the beginning, this chapter demonstrates the preparation process of the training and updating dataset of system operating conditions. As illustrated in the previous chapter, the VFDT system is employed to the periodic update in online VSA. The performance of updating classification model using VFDT system is evaluated, and the result shows that it is capable of training classifiers in a batch mode instead of developing the whole classifiers from scratch.

In order to apply the online VSA analytic, the use of a time-series synchrophasor-data-driven platform — openECA is detailed subsequently. The configuration parameters and necessary data structure preparation for an openECA-based analytic is discussed. A openECA-based testbed analytic — Shadow System Simulator has also been developed in order to complete the closed-loop control architecture for the online VSA applications. Finally, the online voltage security assessment analytic is deployed and tested with Shadow System implemented with the IEEE 118-Bus System model.

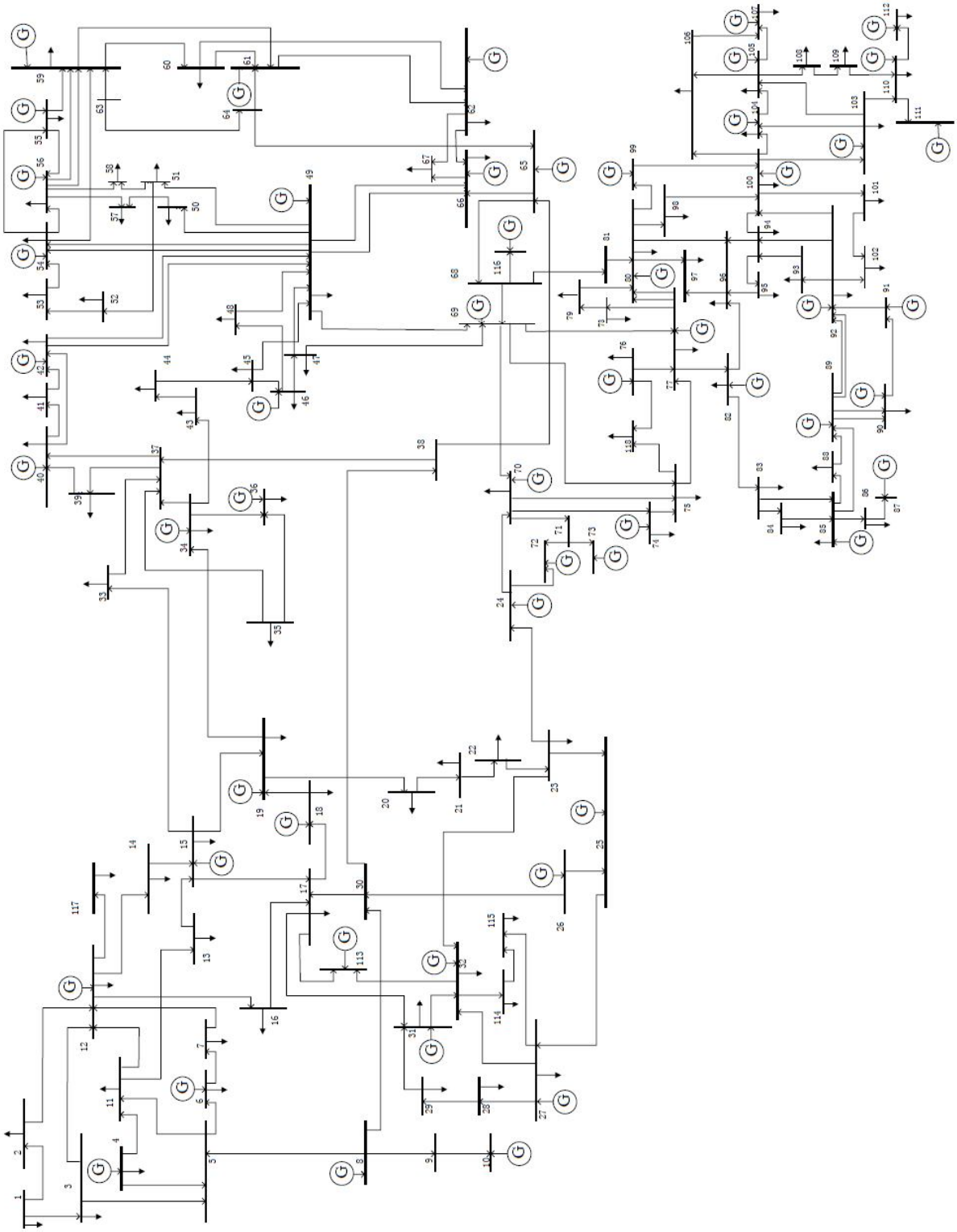

Figure 4.1: IEEE 118-bus standard system

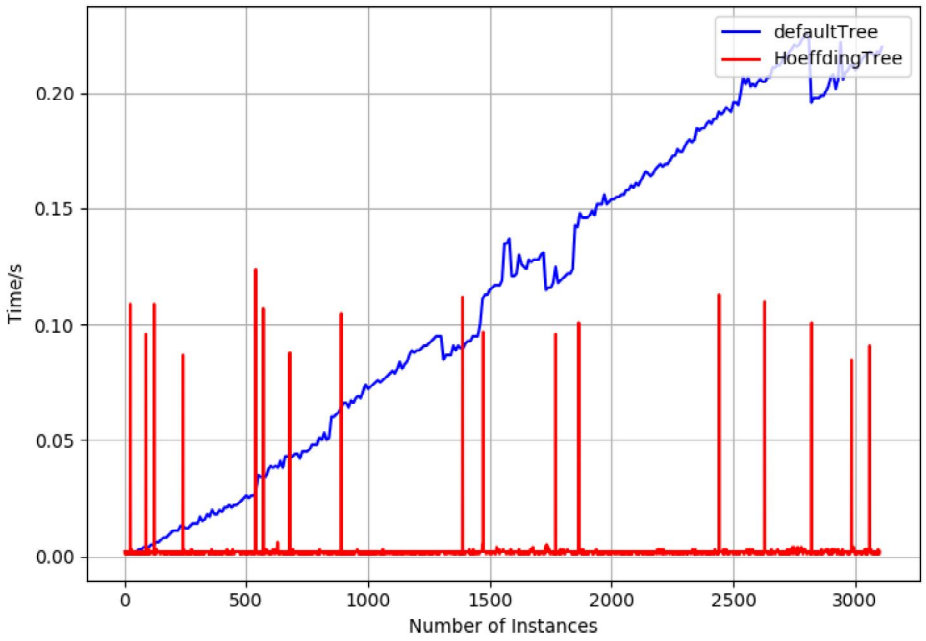

Figure 4.2: Computation time for modifying DT model

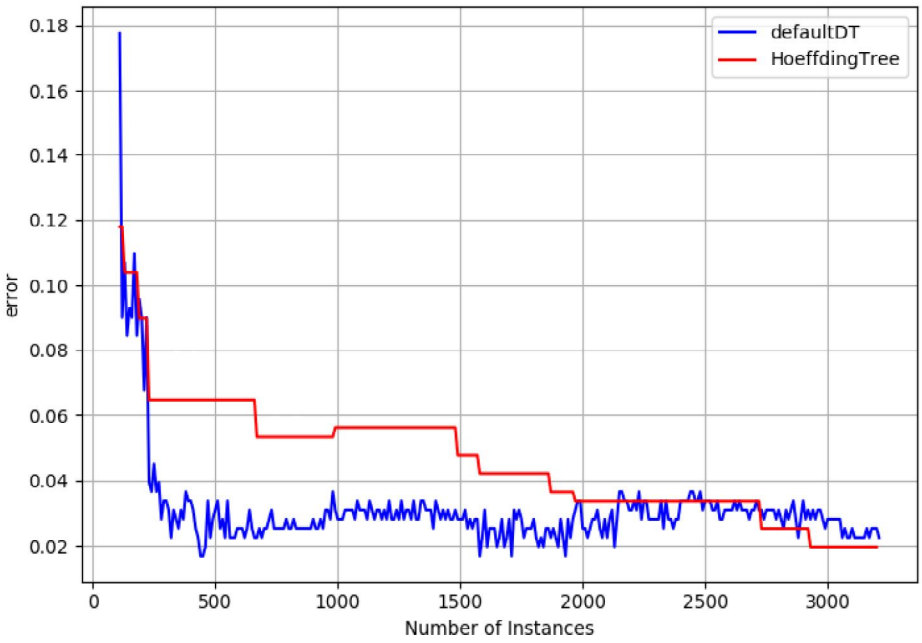

Figure 4.3: Testing error for modified DT model

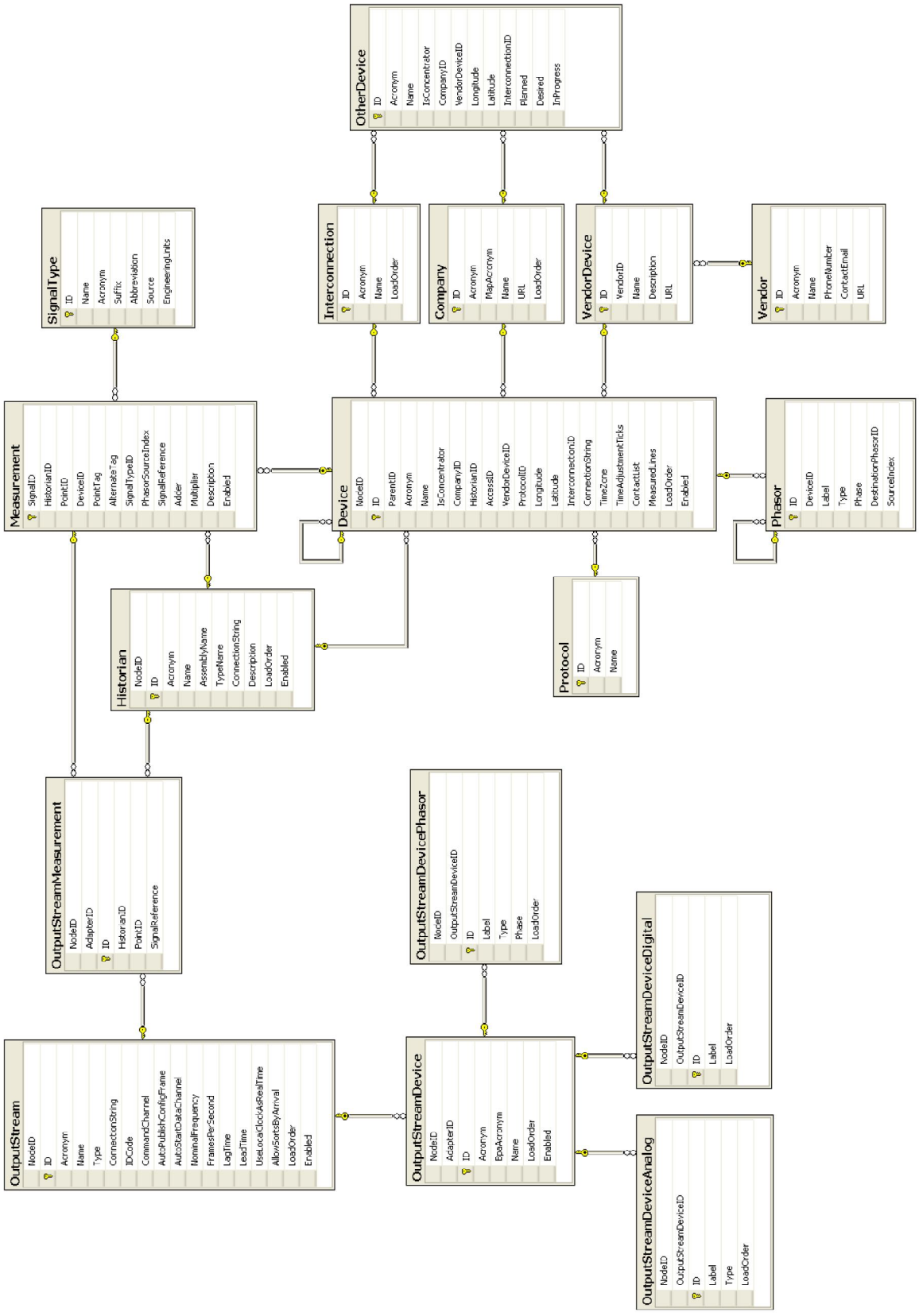

Figure 4.4: Complete Data Structure of openECA Platform

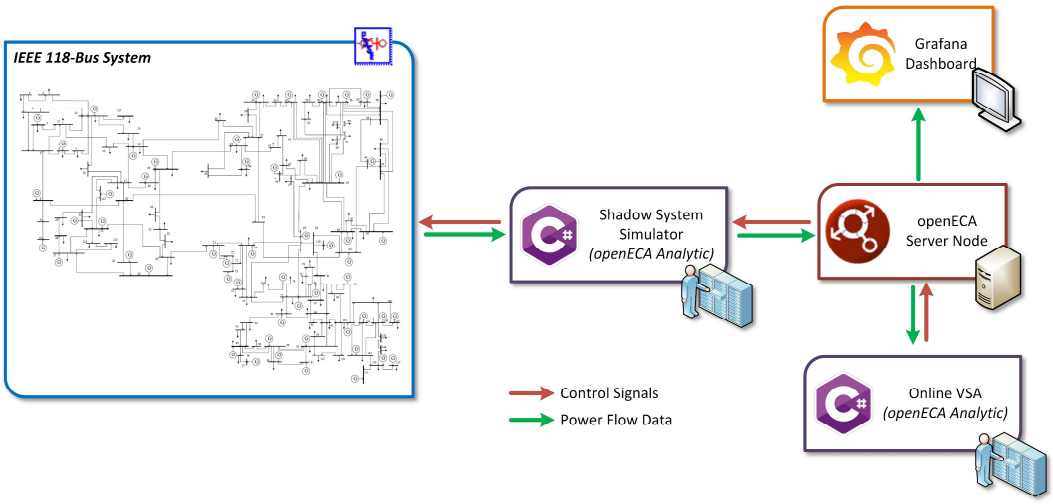

Figure 4.5: Dataflow Overview of Online VSA Applications

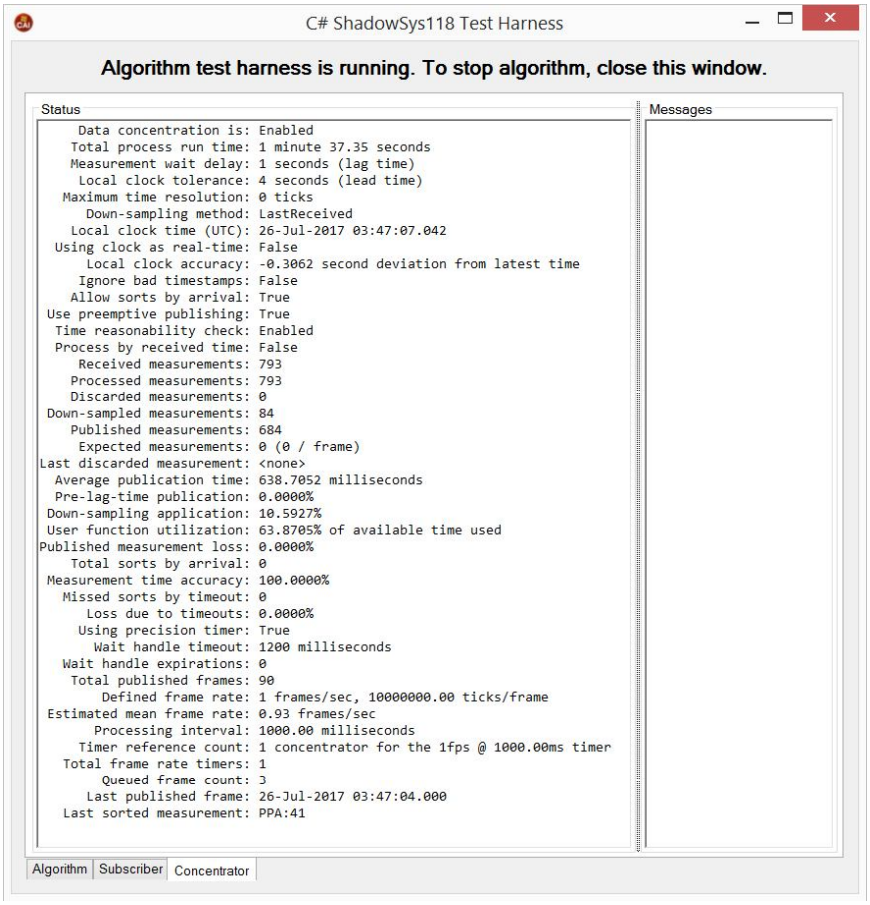

Figure 4.6: Test Harness Metrics of Shadow System Deployed as openECA Analytic

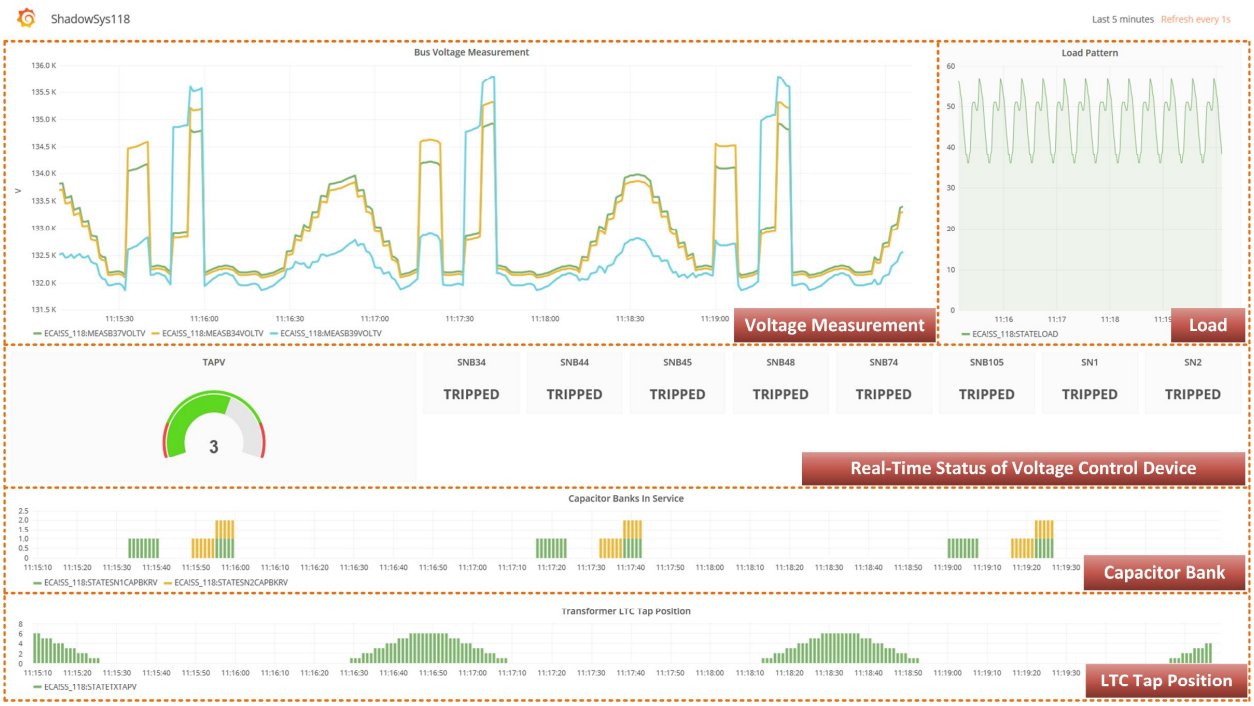

Figure 4.7: Grafana Dashboard for Time-series Data Visualization

Figure 4.8: Simulation Result of Closed-Loop Control Architecture

Chapter 5

Conclusions and Future Works

5.1 Conclusions

In this study, a Hoeffding-tree-based methodology for online voltage security assessment is proposed with respect to real-time high-speed data streams collected from PMUs and PDCs. The proposed assessment framework is initially employed with the generation of DT training sample database considering different OCs with randomly scaled bus load conditions. AdaBoost algorithm, a popular ensemble learning method is implemented in order to develop initial ensemble model of decision trees according to the offline dataset. With the Very Fast Decision Tree (VFDT) system, the developed ensemble models of decision trees are capable of being adaptively updated by an incremental tree induction process based on Hoeffding bound with voting weights recalculated depending on the accuracy of each classifier.

To simulate the real-time use of the online VSA applications, an open-source control and analytics platform — openECA is engaged to manage the time-series measurement data. In order to simulate the static behavior of power systems, a system simulator called "Shadow System Simulator" is developed using power flow calculation results from PSS®E to publish measurement data to openECA in a fixed-interval manner. With Shadow System of IEEE 118-bus system deployed at the background, the openECA-based analytic of online VSA applications can receive the voltage measurement from openECA platform and perform voltage security analysis using the ensemble model of decision trees initialized by offline

training dataset and updated by online supplement dataset. A changed topology scenario is introduced to the system model in order to test the robustness of proposed online periodic update methodology. Simulation results show that the proposed method is able to reduce the computation burden and have a lower misclassification error compared to the traditional decision tree training method.

The contributions of this thesis work is detailed as follows:

- A framework of online voltage security assessment using pure data-driven method — ensemble decision-tree model is presented.
- The periodic update for the ensemble model utilized in online VSA applications is proposed. By using VFDT system, a new tree induction system based on Hoeffding bound, the ensemble model of decision trees can be further induced incrementally, and reweighted according to the accuracy for each modified classifier.
- Shadow System Simulator, a PSS®E-powered testbed for static control analytics is developed using openECA platform.
- The analytic of online VSA applications is developed using openECA platform in order to perform voltage security analysis using pure data-driven method and to provide feasible control strategies when an insecure voltage profile is detected. The closed-loop control architecture is completed by deploying Shadow System as testbed.

5.2 Future Works

From the standpoint of online assessment, a probable extension of proposed framework using VFDT system would include not only static analysis like voltage security assessment, but also dynamic study of power systems like transient stability assessment. In [9], the ensemble model of decision trees are utilized for online dynamic security assessment. With the use of VFDT system, online dynamic security assessment would update the tree models adaptively not only with the voting weights, but also, the tree structure, which is more importantly engaged in the classification tasks.

From the perspective of pure data-driven methodology, the decision-tree learning is a supervised learning method that allows deeper tree induction based on the existing models. The enhancement of Hoeffding-tree-based method would be further extended with the use of option trees in [32]. Another appealing extensions of this work would be to include different machine learning techniques like support vector machine and neural networks in order to allow model update with unknown data incoming in a batch mode.

For a real-time simulator, the Shadow System Simulator utilizes the power flow solver powered by PSS®E; however, such analytic can be further enhanced by using other the time-series-based simulators in order to perfect the simulations with stronger solvers designed for power systems to accomplish in-depth analysis tasks like dynamic analysis with outstanding time-savings.

Bibliography

[1] J. D. L. Ree, V. Centeno, J. S. Thorp, and A. G. Phadke, "Synchronized phasor measurement applications in power systems," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 20–27, June 2010.

[2] E. E. Bernabeu, J. S. Thorp, and V. Centeno, "Methodology for a security / dependability adaptive protection scheme based on data mining," *IEEE Transactions on Power Delivery*, vol. 27, no. 1, pp. 104–111, Jan 2012.

[3] R. Diao, K. Sun, V. Vittal, R. J. O'Keefe, M. R. Richardson, N. Bhatt, D. Stradford, and S. K. Sarawgi, "Decision tree-based online voltage security assessment using pmu measurements," *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 832–839, May 2009.

[4] K. D. Jones, J. S. Thorp, and R. M. Gardner, "Three-phase linear state estimation using phasor measurements," in *2013 IEEE Power Energy Society General Meeting*, July 2013, pp. 1–5.

[5] J. S. Thorp, M. Ilic-Spong, and M. Varghese, "Optimal secondary voltage-var control using pilot point information structure," in *The 23rd IEEE Conference on Decision and Control*, Dec 1984, pp. 462–466.

[6] Q. Guo, H. Sun, M. Zhang, J. Tong, B. Zhang, and B. Wang, "Optimal voltage control of pjm smart transmission grid: Study, implementation, and evaluation," *IEEE Transactions on Smart Grid*, vol. 4, no. 3, pp. 1665–1674, Sept 2013.

[7] D. Yang, R. Sun, J. D. L. Ree, and M. Mcvey, "Capacitor bank model validation with particle swarm optimization algorithm," in *2016 IEEE Power and Energy Society General Meeting (PESGM)*, July 2016, pp. 1–5.

[8] C. Liu, K. Sun, Z. H. Rather, Z. Chen, C. L. Bak, P. Thgersen, and P. Lund, "A systematic approach for dynamic security assessment and the corresponding preventive control scheme based on decision trees," in *2014 IEEE PES General Meeting — Conference Exposition*, July 2014, pp. 1–1.

[9] M. He, J. Zhang, and V. Vittal, "Robust online dynamic security assessment using adaptive ensemble decision-tree learning," *IEEE Transactions on Power Systems*, vol. 28, no. 4, pp. 4089–4098, Nov 2013.

[10] P. E. Utgoff, N. C. Berkman, and J. A. Clouse, "Decision tree induction based on efficient tree restructuring," *Machine Learning*, vol. 29, no. 1, pp. 5–44, 1997.

[11] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar 1986. [Online]. Available: http://dx.doi.org/10.1007/BF00116251

[12] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, "On-line random forests," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, Sept 2009, pp. 1393–1400.

[13] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 71–80.

[14] H. Yang, S. Fong, G. Sun, and R. Wong, "A very fast decision tree algorithm for real-time data mining of imperfect data streams in a distributed wireless sensor network," *International Journal of Distributed Sensor Networks*, 2012.

[15] C. P. Steinmetz, "Power control and stability of electric generating stations," *Transactions of the American Institute of Electrical Engineers*, vol. 39, no. 2, pp. 1215–1287, 1920.

[16] P. Kundur, N. J. Balu, and M. G. Lauby, *Power system stability and control*. McGraw-hill New York, 1994, vol. 7.

[17] P. Kundur, J. Paserba, V. Ajjarapu, G. Andersson, A. Bose, C. Canizares, N. Hatziargyriou, D. Hill, A. Stankovic, C. Taylor, T. V. Cutsem, and V. Vittal, "Definition and classification of power system stability ieee/cigre joint task force on stability terms and definitions," *IEEE Transactions on Power Systems*, vol. 19, no. 3, pp. 1387–1401, Aug 2004.

[18] D. Barber, *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[19] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning.* Springer series in statistics New York, 2001, vol. 1.

[20] T. V. Cutsem, L. Wehenkel, M. Pavella, B. Heilbronn, and M. Goubin, "Decision tree approaches to voltage security assessment," *IEE Proceedings C - Generation, Transmission and Distribution*, vol. 140, no. 3, pp. 189–198, May 1993.

[21] L. Wehenkel, T. V. Cutsem, and M. Ribbens-Pavella, "An artificial intelligence framework for online transient stability assessment of power systems," *IEEE Transactions on Power Systems*, vol. 4, no. 2, pp. 789–800, May 1989.

[22] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *European conference on computational learning theory.* Springer, 1995, pp. 23–37.

[23] C. Taylor, N. Balu, and D. Maratukulam, *Power system voltage stability*, ser. The EPRI Power System Engineering Series. McGraw-Hill Ryerson, Limited, 1994. [Online]. Available: https://books.google.com/books?id=CPtSAAAAMAAJ

[24] T. Van Cutsem and C. Vournas, *Voltage stability of electric power systems.* Springer Science & Business Media, 1998, vol. 441.

[25] P. Kessel and H. Glavitsch, "Estimating the voltage stability of a power system," *IEEE Power Engineering Review*, vol. PER-6, no. 7, pp. 72–72, July 1986.

[26] S. Shukla and L. Mili, "A hierarchical decentralized coordinated voltage instability detection scheme for svc," in *2015 North American Power Symposium (NAPS)*, Oct 2015, pp. 1–6.

[27] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2000, pp. 71–80.

[28] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Data stream mining - a practical approach. [Online]. Available: http://www.cs.waikato.ac.nz/ abifet/MOA/StreamMining.pdf

[29] D. G. Denison, B. K. Mallick, and A. F. Smith, "A bayesian cart algorithm," *Biometrika*, vol. 85, no. 2, pp. 363–377, 1998.

[30] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, no. 1, pp. 105–139, 1999.

[31] H. Mehrjerdi, S. Lefebvre, M. Saad, and D. Asber, "A decentralized control of partitioned power networks for voltage regulation and prevention against disturbance propagation," *IEEE Transactions on Power Systems*, vol. 28, no. 2, pp. 1461–1469, 2013.

[32] R. Kohavi and C. Kunz, "Option decision trees with majority votes," in *ICML*, vol. 97, 1997, pp. 161–169.

Appendix A

Python Implementations of Hoeffding Trees

A.1 main()

```
import sys, csv
import numpy as np
import scipy.io as sio
import time
import matplotlib.pyplot as plt
# import pydotplus
from sklearn import tree
from sklearn.metrics import accuracy_score
from hoeffdingtree import *

def main():
    InitFileName = 'IEEE300_load_init100.csv'
    TrainFileName = 'IEEE300_load_train3100.csv'
    TestFileName = 'IEEE300_test_last356.csv'

    InitDataset = open_dataset(InitFileName, 0, probe_instances=10000)
    TrainDataset = open_dataset(TrainFileName, 0, probe_instances=10000)
    TestDataset = open_dataset(TestFileName, 0, probe_instances=10000)

    elapsed_time = []
```

```
err = []

vfdt = HoeffdingTree()
# Set some of the algorithm parameters
vfdt.set_grace_period(50)
vfdt.set_hoeffding_tie_threshold(0.4)
vfdt.set_split_confidence(0.0000001)
# Split criterion, for now, can only be set on hoeffdingtree.py file.
# This is only relevant when Information Gain is chosen as the split criterion
vfdt.set_minimum_fraction_of_weight_info_gain(0.01)

vfdt.build_classifier(InitDataset)
print(vfdt)
NumInstances = 0

# Simulate a data stream
with open(TrainFileName) as f:
    stream = csv.reader(f)
    # Ignore the CSV headers
    next(stream)
    # for item in stream:
    for item in stream:
        NumInstances = NumInstances + 1
        inst_values = list(item)
        for i in range(len(inst_values)):
            if TrainDataset.attribute(index=i).type() == 'Nominal':
                inst_values[i] = int(TrainDataset.attribute(index=i)
                                     .index_of_value(str(inst_values[i])))
            else:
                inst_values[i] = float(inst_values[i])
        new_instance = Instance(att_values=inst_values)
        new_instance.set_dataset(TrainDataset)

        # Record the time duration while updating the model
        tic = time.time()
        vfdt.update_classifier(new_instance)
        elapsed_time.append([NumInstances, time.time() - tic])
        print('[Hoeffding Training] HT model is trained with {0} instances in the
            dataset...'.format(NumInstances))


        # Simulate another data stream to test the Model
```

```
            # Test the accuracy every 10 instances
            if (NumInstances % 10 == 0):
                NumTestInstances = 0
                err_count = 0

                with open(TestFileName) as f:
                    stream = csv.reader(f)
                    # Ignore the CSV headers
                    next(stream)
                    # for item in stream:
                    for item in stream:
                        NumTestInstances = NumTestInstances + 1
                        inst_values = list(item)
                        for i in range(len(inst_values)):
                            if TestDataset.attribute(index=i).type() == 'Nominal':
                                inst_values[i] = int(TestDataset.attribute(index=i)
                                                .index_of_value(str(inst_values[i])))
                            else:
                                inst_values[i] = float(inst_values[i])
                        new_instance = Instance(att_values=inst_values)
                        new_instance.set_dataset(TestDataset)
                        err_count = err_count + vfdt.predict_classifier(new_instance)
                    print('\tTesting Error Rate: {0}/{1} = {2}%'.format(err_count,
                        NumTestInstances,
                                                            100 * err_count /
                                                                NumTestInstances))

                # Calculate the training error for the new model
                err.append([NumInstances+100, err_count / NumTestInstances])


    elapsed_time = np.array(elapsed_time)
    err = np.array(err)
    return (vfdt,elapsed_time,err)
```

A.2 *class* HoeffdingTree()

```
import math
from operator import attrgetter
```

```
from core import utils
from core.attribute import Attribute
from core.instance import Instance
from core.dataset import Dataset

from ht.activehnode import ActiveHNode
from ht.ginisplitmetric import GiniSplitMetric
from ht.hnode import HNode
from ht.inactivehnode import InactiveHNode
from ht.infogainsplitmetric import InfoGainSplitMetric
from ht.leafnode import LeafNode
from ht.splitcandidate import SplitCandidate
from ht.splitmetric import SplitMetric
from ht.splitnode import SplitNode

class HoeffdingTree(object):
    """Main class for a Hoeffding Tree, also known as Very Fast Decision Tree
        (VFDT)."""
    def __init__(self):
        self._header = None
        self._root = None
        self._grace_period = 200
        self._split_confidence = 0.0000001
        self._hoeffding_tie_threshold = 0.05
        self._min_frac_weight_for_two_branches_gain = 0.01

        # Split metric stuff goes here
        self.GINI_SPLIT = 0
        self.INFO_GAIN_SPLIT = 1

        self._selected_split_metric = self.INFO_GAIN_SPLIT
        self._split_metric =
            InfoGainSplitMetric(self._min_frac_weight_for_two_branches_gain)
        #self._selected_split_metric = self.GINI_SPLIT
        #self._split_metric = GiniSplitMetric()

        # Leaf prediction strategy stuff goes here

        # Only used when the leaf prediction strategy is based on Naive Bayes, not
            useful right now
        #self._nb_threshold = 0
```

```
        self._active_leaf_count = 0
        self._inactive_leaf_count = 0
        self._decision_node_count = 0

        # Print out leaf models in the case of naive Bayes or naive Bayes adaptive
            leaves
        self._print_leaf_models = False

    def __str__(self):
        if self._root is None:
            return 'No model built yet!'
        return self._root.__str__(self._print_leaf_models)

    def reset(self):
        """Reset the classifier and set all node/leaf counters to zero."""
        self._root = None
        self._active_leaf_count = 0
        self._inactive_leaf_count = 0
        self._decision_node_count = 0

    def set_minimum_fraction_of_weight_info_gain(self, m):
        self._min_frac_weight_for_two_branches_gain = m

    def get_minimum_fraction_of_weight_info_gain(self):
        return self._min_frac_weight_for_two_branches_gain

    def set_grace_period(self, grace):
        self._grace_period = grace

    def get_grace_period(self):
        return self._grace_period

    def set_hoeffding_tie_threshold(self, ht):
        self._hoeffding_tie_threshold = ht

    def get_hoeffding_tie_threshold(self):
        return self._hoeffding_tie_threshold

    def set_split_confidence(self, sc):
        self._split_confidence = sc

    def get_split_confidence(self):
```

```
    return self._split_confidence

def compute_hoeffding_bound(self, max_value, confidence, weight):
    """Calculate the Hoeffding bound.

    Args:
        max_value (float):
        confidence (float):
        weight (float):

    Returns:
        (float): The Hoeffding bound.
    """
    return math.sqrt(((max_value * max_value) * math.log(1.0 / confidence)) /
        (2.0 * weight))

def build_classifier(self, dataset):
    """Build the classifier.

    Args:
        dataset (Dataset): The data to start training the classifier.
    """
    self.reset()
    self._header = dataset
    if self._selected_split_metric is self.GINI_SPLIT:
        self._split_metric = GiniSplitMetric()
    else:
        self._split_metric =
            InfoGainSplitMetric(self._min_frac_weight_for_two_branches_gain)

    for i in range(dataset.num_instances()):
        self.update_classifier(dataset.instance(i))

def update_classifier(self, instance):
    """Update the classifier with the given instance.

    Args:
        instance (Instance): The new instance to be used to train the classifier.
    """
    if instance.class_is_missing():
        return
    if self._root is None:
```

```
            self._root = self.new_learning_node()

        l = self._root.leaf_for_instance(instance, None, None)
        actual_node = l.the_node
        if actual_node is None:
            actual_node = ActiveHNode()
            l.parent_node.set_child(l.parent_branch, actual_node)

        # ActiveHNode should be changed to a LearningNode interface if Naive Bayes
            nodes are used
        if isinstance(actual_node, InactiveHNode):
            actual_node.update_node(instance)
        if isinstance(actual_node, ActiveHNode):
            actual_node.update_node(instance)
            total_weight = actual_node.total_weight()
            if total_weight - actual_node.weight_seen_at_last_split_eval >
                self._grace_period:
                self.try_split(actual_node, l.parent_node, l.parent_branch)
                actual_node.weight_seen_at_last_split_eval = total_weight

    def predict_classifier(self, instance):
        """Predict the result of the instance given the trained classifier.

        Args:
            instance (Instance): The instance to be used to make the prediction.
        """
        if instance.class_is_missing():
            return
        if self._root is None:
            self._root = self.new_learning_node()

        l = self._root.leaf_for_instance(instance, None, None)
        actual_node = l.the_node

        if actual_node is None:
            actual_node = ActiveHNode()
            l.parent_node.set_child(l.parent_branch, actual_node)

        # predict/actual_placement class
        ins_class_predict = str(actual_node)

        # true class
```

```
    ins_class_attribute = instance.class_attribute()
    ins_class_true = str(ins_class_attribute.value(instance.class_value()))


    PredictTag = (ins_class_predict == ins_class_true)
    if PredictTag:
        err_increment = 0
    else:
        err_increment = 1

    return err_increment

def distribution_for_instance(self, instance):
    """Return the class probabilities for an instance.

    Args:
        instance (Instance): The instance to calculate the class probabilites for.

    Returns:
        list[float]: The class probabilities.
    """
    class_attribute = instance.class_attribute()
    pred = []

    if self._root is not None:
        l = self._root.leaf_for_instance(instance, None, None)
        actual_node = l.the_node
        if actual_node is None:
            actual_node = l.parent_node
        pred = actual_node.get_distribution(instance, class_attribute)
    else:
        # All class values equally likely
        pred = [1 for i in range(class_attribute.num_values())]
        utils.normalize(pred)

    return pred


def deactivate_node(self, to_deactivate, parent, parent_branch):
    """Prevent supplied node of growing.

    Args:
```

```
            to_deactivate (ActiveHNode): The node to be deactivated.
            parent (SplitNode): The parent of the node.
            parent_branch (str): The branch leading from the parent to the node.
        """
        leaf = InactiveHNode(to_deactivate.class_distribution)

        if parent is None:
            self._root = leaf
        else:
            parent.set_child(parent_branch, leaf)

        self._active_leaf_count -= 1
        self._inactive_leaf_count += 1

    def activate_node(self, to_activate, parent, parent_branch):
        """Allow supplied node to grow.

        Args:
            to_activate (InactiveHNode): The node to be activated.
            parent (SplitNode): The parent of the node.
            parent_branch (str): The branch leading from the parent to the node.
        """
        leaf = ActiveHNode()
        leaf.class_distribution = to_activate.class_distribution

        if parent is None:
            self._root = leaf
        else:
            parent.set_child(parent_branch, leaf)

        self._active_leaf_count += 1
        self._inactive_leaf_count -= 1

    def try_split(self, node, parent, parent_branch):
        """Try a split from the supplied node.

        Args:
            node (ActiveHNode): The node to split.
            parent (SplitNode): The parent of the node.
            parent_branch (str): The branch leading from the parent to the node.
        """
        # Non-pure?
```

```
if node.num_entries_in_class_distribution() > 1:
    best_splits = node.get_possible_splits(self._split_metric)
    best_splits.sort(key=attrgetter('split_merit'))

    do_split = False
    if len(best_splits) < 2:
        do_split = len(best_splits) > 0
    else:
        # Compute Hoeffding bound
        metric_max =
            self._split_metric.get_metric_range(node.class_distribution)
        hoeffding_bound = self.compute_hoeffding_bound(
            metric_max, self._split_confidence, node.total_weight())
        best = best_splits[len(best_splits) - 1]
        second_best = best_splits[len(best_splits) - 2]
        if best.split_merit - second_best.split_merit > hoeffding_bound or
            hoeffding_bound < self._hoeffding_tie_threshold:
            do_split = True

    if do_split:
        best = best_splits[len(best_splits) - 1]
        if best.split_test is None:
            # preprune
            self.deactivate_node(node, parent, parent_branch)
        else:
            new_split = SplitNode(node.class_distribution, best.split_test)

            for i in range(best.num_splits()):
                new_child = self.new_learning_node()
                new_child.class_distribution =
                    best.post_split_class_distributions[i]
                new_child.weight_seen_at_last_split_eval =
                    new_child.total_weight()
                branch_name = ''
                if
                    self._header.attribute(name=best.split_test.split_attributes()[0]).is_nume
                    if i is 0:
                        branch_name = 'left'
                    else:
                        branch_name = 'right'
                else:
                    split_attribute =
```

```
                        self._header.attribute(name=best.split_test.split_attributes()[0])
                    branch_name = split_attribute.value(i)
                new_split.set_child(branch_name, new_child)

            self._active_leaf_count -= 1
            self._decision_node_count += 1
            self._active_leaf_count += best.num_splits()

            if parent is None:
                self._root = new_split
            else:
                parent.set_child(parent_branch, new_split)

    def new_learning_node(self):
        """Create a new learning node. Will always be an ActiveHNode while Naive Bayes
        nodes are not implemented.

        Returns:
            ActiveHNode: The new learning node.
        """
        # Leaf strategy should be handled here if/when the Naive Bayes approach is
            implemented
        return ActiveHNode()
```

A.3 *class* ActiveHNode()

```
from ht.leafnode import LeafNode
from ht.hnode import HNode
from ht.gaussianconditionalsufficientstats import GaussianConditionalSufficientStats
from ht.nominalconditionalsufficientstats import NominalConditionalSufficientStats
from ht.splitcandidate import SplitCandidate

class ActiveHNode(LeafNode):
    """A Hoeffding Tree node that supports growth."""
    def __init__(self):
        super().__init__()
        # The total weight of the instances seen at the last split evaluation.
        self.weight_seen_at_last_split_eval = 0
        # Statistics for the attributes.
        # Dict of tuples (attribute name, ConditionalSufficientStats).
```

```
    self._node_stats = {}

def update_node(self, instance):
    """Update the node with the supplied instance.

    Args:
        instance (Instance): The instance to be used for updating the node.
    """
    self.update_distribution(instance)
    for i in range(instance.num_attributes()):
        a = instance.attribute(i)
        if i is not instance.class_index():
            stats = self._node_stats.get(a.name(), None)
            if stats is None:
                if a.is_numeric():
                    stats = GaussianConditionalSufficientStats()
                else:
                    stats = NominalConditionalSufficientStats()
                self._node_stats[a.name()] = stats

            stats.update(instance.value(attribute=a),
                instance.class_attribute().value(index=instance.class_value()),
                instance.weight())

def get_possible_splits(self, split_metric):
    """Return a list of the possible split candidates.

    Args:
        split_metric (SplitMetric): The splitting metric to be used.

    Returns:
        list[SplitCandidate]: A list of the possible split candidates.
    """
    splits = []
    null_dist = []
    null_dist.append(self.class_distribution)
    null_split = SplitCandidate(None, null_dist,
        split_metric.evaluate_split(self.class_distribution, null_dist))
    splits.append(null_split)

    for attribute_name, stat in self._node_stats.items():
        split_candidate = stat.best_split(split_metric, self.class_distribution,
```

```
                attribute_name)
            if split_candidate is not None:
                splits.append(split_candidate)

        return splits
```

A.4 *class* GiniSplitMetric()

```
from ht.splitmetric import SplitMetric

class GiniSplitMetric(SplitMetric):
    """The Gini split metric."""
    def evaluate_split(self, pre_dist, post_dist):
        total_weight = 0.0
        dist_weights = []
        for i in range(len(post_dist)):
            dist_weights.append(self.sum(post_dist[i]))
            total_weight += dist_weights[i]
        gini_metric = 0
        for i in range(len(post_dist)):
            gini_metric += (dist_weights[i] / total_weight) * self.gini(
                post_dist[i], dist_weights[i])

        return 1.0 - gini_metric

    def gini(self, dist, sum_of_weights=None):
        if sum_of_weights is None:
            sum_of_weights = self.sum(dist)
        gini_metric = 1.0
        for class_value, mass in dist.items():
            frac = mass.weight / sum_of_weights
            gini_metric -= frac * frac
        return gini_metric

    def get_metric_range(self, pre_dist):
        return 1.0
```

Appendix B

Management of Data Structure in openECA

B.1 Manage Data Structure using SQL scripts

```
USE openECA;

INSERT INTO Node(Name, CompanyID, Description, Settings, MenuType, MenuData, Master,
    LoadOrder, Enabled)
    VALUES('Default', NULL, 'Default node',
        'RemoteStatusServerConnectionString={server=localhost:8525;integratedSecurity=true};dataPubl
        'File', 'Menu.xml', 1, 0, 1);

UPDATE Node SET ID='e57b4a6d-ca9e-403c-ad2e-9a1db9a8a707' WHERE Name='Default';

INSERT INTO Historian(NodeID, Acronym, Name, AssemblyName, TypeName,
    ConnectionString, IsLocal, Description, LoadOrder, Enabled)
    VALUES('e57b4a6d-ca9e-403c-ad2e-9a1db9a8a707', 'PPA', 'Primary Phasor Archive',
    'TestingAdapters.dll', 'TestingAdapters.VirtualOutputAdapter', '', 1, 'Primary
    Phasor Archive', 0, 1);
INSERT INTO Device(NodeID, Acronym, Name, IsConcentrator, CompanyID, HistorianID,
    AccessID, VendorDeviceID, ProtocolID, Longitude, Latitude, InterconnectionID,
    ConnectionString, MeasuredLines, LoadOrder, Enabled)
    VALUES('e57b4a6d-ca9e-403c-ad2e-9a1db9a8a707', 'TESTDEVICE', 'Test Device', 0,
    30, 1, 2, 2, 3, -89.8038, 35.3871, 1, 'transportProtocol=File;
```

```
    file=Sample1344.PmuCapture; useHighResolutionInputTimer=True', 3, 0, 1);

INSERT INTO Phasor(DeviceID, Label, Type, Phase, SourceIndex) VALUES(1, '500 kV Bus
    1', 'V', '+', 1);
INSERT INTO Phasor(DeviceID, Label, Type, Phase, SourceIndex) VALUES(1, '500 kV Bus
    2', 'V', '+', 2);
INSERT INTO Phasor(DeviceID, Label, Type, Phase, SourceIndex) VALUES(1, 'Cordova',
    'I', '+', 3);
INSERT INTO Phasor(DeviceID, Label, Type, Phase, SourceIndex) VALUES(1, 'Dell', 'I',
    '+', 4);
INSERT INTO Phasor(DeviceID, Label, Type, Phase, SourceIndex) VALUES(1, 'Lagoon
    Creek', 'I', '+', 5);


-- Shadow System using 118 bus system(ShadowSys118) for Online VSA Applications
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:LOADINCRE', 9, NULL, 'SS118-LOADINCRE', 'Shadow System for 118-bus system
    - Load Increment in percentage', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:RESET', 9, NULL, 'SS118-RESET', 'Shadow System for 118-bus system - Reset
    Signal to read initial system configuration', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB34CLOSE', 9, NULL, 'SS118-ACTSNB34CLOSE', 'Shadow System for
    118-bus system - Action flag of closing Capacitor Bank at bus 34 ActSnB34Close',
    1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB44CLOSE', 9, NULL, 'SS118-ACTSNB44CLOSE', 'Shadow System for
    118-bus system - Action flag of closing Capacitor Bank at bus 44 ActSnB44Close',
    1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB45CLOSE', 9, NULL, 'SS118-ACTSNB45CLOSE', 'Shadow System for
    118-bus system - Action flag of closing Capacitor Bank at bus 45 ActSnB45Close',
    1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB48CLOSE', 9, NULL, 'SS118-ACTSNB48CLOSE', 'Shadow System for
    118-bus system - Action flag of closing Capacitor Bank at bus 48 ActSnB48Close',
```

```
    1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB74CLOSE', 9, NULL, 'SS118-ACTSNB74CLOSE', 'Shadow System for
    118-bus system - Action flag of closing Capacitor Bank at bus 74 ActSnB74Close',
    1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB105CLOSE', 9, NULL, 'SS118-ACTSNB105CLOSE', 'Shadow System for
    118-bus system - Action flag of closing Capacitor Bank at bus 105
    ActSnB105Close', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB34TRIP', 9, NULL, 'SS118-ACTSNB34TRIP', 'Shadow System for 118-bus
    system - Action flag of tripping Capacitor Bank at bus 34 ActSnB34Trip', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB44TRIP', 9, NULL, 'SS118-ACTSNB44TRIP', 'Shadow System for 118-bus
    system - Action flag of tripping Capacitor Bank at bus 44 ActSnB44Trip', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB45TRIP', 9, NULL, 'SS118-ACTSNB45TRIP', 'Shadow System for 118-bus
    system - Action flag of tripping Capacitor Bank at bus 45 ActSnB45Trip', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB48TRIP', 9, NULL, 'SS118-ACTSNB48TRIP', 'Shadow System for 118-bus
    system - Action flag of tripping Capacitor Bank at bus 48 ActSnB48Trip', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB74TRIP', 9, NULL, 'SS118-ACTSNB74TRIP', 'Shadow System for 118-bus
    system - Action flag of tripping Capacitor Bank at bus 74 ActSnB74Trip', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:ACTSNB105TRIP', 9, NULL, 'SS118-ACTSNB105TRIP', 'Shadow System for
    118-bus system - Action flag of tripping Capacitor Bank at bus 105
    ActSnB105Trip', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:STATESNB34CAPBKRV', 9, NULL, 'SS118-STATESNB34CAPBKRV', 'Shadow System
    for 118-bus system - Capacitor Bank at bus 34 circuit breaker state value
    StateSnB34CapBkrV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
```

```
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:STATESNB44CAPBKRV', 9, NULL, 'SS118-STATESNB44CAPBKRV', 'Shadow System
    for 118-bus system - Capacitor Bank at bus 44 circuit breaker state value
    StateSnB44CapBkrV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:STATESNB45CAPBKRV', 9, NULL, 'SS118-STATESNB45CAPBKRV', 'Shadow System
    for 118-bus system - Capacitor Bank at bus 45 circuit breaker state value
    StateSnB45CapBkrV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:STATESNB48CAPBKRV', 9, NULL, 'SS118-STATESNB48CAPBKRV', 'Shadow System
    for 118-bus system - Capacitor Bank at bus 48 circuit breaker state value
    StateSnB48CapBkrV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:STATESNB74CAPBKRV', 9, NULL, 'SS118-STATESNB74CAPBKRV', 'Shadow System
    for 118-bus system - Capacitor Bank at bus 74 circuit breaker state value
    StateSnB74CapBkrV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:STATESNB105CAPBKRV', 9, NULL, 'SS118-STATESNB105CAPBKRV', 'Shadow System
    for 118-bus system - Capacitor Bank at bus 105 circuit breaker state value
    StateSnB105CapBkrV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB1VOLTV', 3, NULL, 'SS118-MEASB1VOLTV', 'Shadow System for 118-bus
    system - Bus 1 Voltage Magnitude MeasB1VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB2VOLTV', 3, NULL, 'SS118-MEASB2VOLTV', 'Shadow System for 118-bus
    system - Bus 2 Voltage Magnitude MeasB2VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB3VOLTV', 3, NULL, 'SS118-MEASB3VOLTV', 'Shadow System for 118-bus
    system - Bus 3 Voltage Magnitude MeasB3VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB4VOLTV', 3, NULL, 'SS118-MEASB4VOLTV', 'Shadow System for 118-bus
    system - Bus 4 Voltage Magnitude MeasB4VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
```

```
    'SS_118:MEASB5VOLTV', 3, NULL, 'SS118-MEASB5VOLTV', 'Shadow System for 118-bus
    system - Bus 5 Voltage Magnitude MeasB5VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB6VOLTV', 3, NULL, 'SS118-MEASB6VOLTV', 'Shadow System for 118-bus
    system - Bus 6 Voltage Magnitude MeasB6VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB7VOLTV', 3, NULL, 'SS118-MEASB7VOLTV', 'Shadow System for 118-bus
    system - Bus 7 Voltage Magnitude MeasB7VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB8VOLTV', 3, NULL, 'SS118-MEASB8VOLTV', 'Shadow System for 118-bus
    system - Bus 8 Voltage Magnitude MeasB8VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB9VOLTV', 3, NULL, 'SS118-MEASB9VOLTV', 'Shadow System for 118-bus
    system - Bus 9 Voltage Magnitude MeasB9VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB10VOLTV', 3, NULL, 'SS118-MEASB10VOLTV', 'Shadow System for 118-bus
    system - Bus 10 Voltage Magnitude MeasB10VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB11VOLTV', 3, NULL, 'SS118-MEASB11VOLTV', 'Shadow System for 118-bus
    system - Bus 11 Voltage Magnitude MeasB11VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB12VOLTV', 3, NULL, 'SS118-MEASB12VOLTV', 'Shadow System for 118-bus
    system - Bus 12 Voltage Magnitude MeasB12VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB13VOLTV', 3, NULL, 'SS118-MEASB13VOLTV', 'Shadow System for 118-bus
    system - Bus 13 Voltage Magnitude MeasB13VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB14VOLTV', 3, NULL, 'SS118-MEASB14VOLTV', 'Shadow System for 118-bus
    system - Bus 14 Voltage Magnitude MeasB14VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB15VOLTV', 3, NULL, 'SS118-MEASB15VOLTV', 'Shadow System for 118-bus
    system - Bus 15 Voltage Magnitude MeasB15VoltV', 1);
```

```
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB16VOLTV', 3, NULL, 'SS118-MEASB16VOLTV', 'Shadow System for 118-bus
    system - Bus 16 Voltage Magnitude MeasB16VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB17VOLTV', 3, NULL, 'SS118-MEASB17VOLTV', 'Shadow System for 118-bus
    system - Bus 17 Voltage Magnitude MeasB17VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB18VOLTV', 3, NULL, 'SS118-MEASB18VOLTV', 'Shadow System for 118-bus
    system - Bus 18 Voltage Magnitude MeasB18VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB19VOLTV', 3, NULL, 'SS118-MEASB19VOLTV', 'Shadow System for 118-bus
    system - Bus 19 Voltage Magnitude MeasB19VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB20VOLTV', 3, NULL, 'SS118-MEASB20VOLTV', 'Shadow System for 118-bus
    system - Bus 20 Voltage Magnitude MeasB20VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB21VOLTV', 3, NULL, 'SS118-MEASB21VOLTV', 'Shadow System for 118-bus
    system - Bus 21 Voltage Magnitude MeasB21VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB22VOLTV', 3, NULL, 'SS118-MEASB22VOLTV', 'Shadow System for 118-bus
    system - Bus 22 Voltage Magnitude MeasB22VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB23VOLTV', 3, NULL, 'SS118-MEASB23VOLTV', 'Shadow System for 118-bus
    system - Bus 23 Voltage Magnitude MeasB23VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB24VOLTV', 3, NULL, 'SS118-MEASB24VOLTV', 'Shadow System for 118-bus
    system - Bus 24 Voltage Magnitude MeasB24VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB25VOLTV', 3, NULL, 'SS118-MEASB25VOLTV', 'Shadow System for 118-bus
    system - Bus 25 Voltage Magnitude MeasB25VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
```

```
    'SS_118:MEASB26VOLTV', 3, NULL, 'SS118-MEASB26VOLTV', 'Shadow System for 118-bus
    system - Bus 26 Voltage Magnitude MeasB26VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB27VOLTV', 3, NULL, 'SS118-MEASB27VOLTV', 'Shadow System for 118-bus
    system - Bus 27 Voltage Magnitude MeasB27VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB28VOLTV', 3, NULL, 'SS118-MEASB28VOLTV', 'Shadow System for 118-bus
    system - Bus 28 Voltage Magnitude MeasB28VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB29VOLTV', 3, NULL, 'SS118-MEASB29VOLTV', 'Shadow System for 118-bus
    system - Bus 29 Voltage Magnitude MeasB29VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB30VOLTV', 3, NULL, 'SS118-MEASB30VOLTV', 'Shadow System for 118-bus
    system - Bus 30 Voltage Magnitude MeasB30VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB31VOLTV', 3, NULL, 'SS118-MEASB31VOLTV', 'Shadow System for 118-bus
    system - Bus 31 Voltage Magnitude MeasB31VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB32VOLTV', 3, NULL, 'SS118-MEASB32VOLTV', 'Shadow System for 118-bus
    system - Bus 32 Voltage Magnitude MeasB32VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB33VOLTV', 3, NULL, 'SS118-MEASB33VOLTV', 'Shadow System for 118-bus
    system - Bus 33 Voltage Magnitude MeasB33VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB34VOLTV', 3, NULL, 'SS118-MEASB34VOLTV', 'Shadow System for 118-bus
    system - Bus 34 Voltage Magnitude MeasB34VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB35VOLTV', 3, NULL, 'SS118-MEASB35VOLTV', 'Shadow System for 118-bus
    system - Bus 35 Voltage Magnitude MeasB35VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB36VOLTV', 3, NULL, 'SS118-MEASB36VOLTV', 'Shadow System for 118-bus
    system - Bus 36 Voltage Magnitude MeasB36VoltV', 1);
```

```
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB37VOLTV', 3, NULL, 'SS118-MEASB37VOLTV', 'Shadow System for 118-bus
    system - Bus 37 Voltage Magnitude MeasB37VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB38VOLTV', 3, NULL, 'SS118-MEASB38VOLTV', 'Shadow System for 118-bus
    system - Bus 38 Voltage Magnitude MeasB38VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB39VOLTV', 3, NULL, 'SS118-MEASB39VOLTV', 'Shadow System for 118-bus
    system - Bus 39 Voltage Magnitude MeasB39VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB40VOLTV', 3, NULL, 'SS118-MEASB40VOLTV', 'Shadow System for 118-bus
    system - Bus 40 Voltage Magnitude MeasB40VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB41VOLTV', 3, NULL, 'SS118-MEASB41VOLTV', 'Shadow System for 118-bus
    system - Bus 41 Voltage Magnitude MeasB41VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB42VOLTV', 3, NULL, 'SS118-MEASB42VOLTV', 'Shadow System for 118-bus
    system - Bus 42 Voltage Magnitude MeasB42VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB43VOLTV', 3, NULL, 'SS118-MEASB43VOLTV', 'Shadow System for 118-bus
    system - Bus 43 Voltage Magnitude MeasB43VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB44VOLTV', 3, NULL, 'SS118-MEASB44VOLTV', 'Shadow System for 118-bus
    system - Bus 44 Voltage Magnitude MeasB44VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB45VOLTV', 3, NULL, 'SS118-MEASB45VOLTV', 'Shadow System for 118-bus
    system - Bus 45 Voltage Magnitude MeasB45VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB46VOLTV', 3, NULL, 'SS118-MEASB46VOLTV', 'Shadow System for 118-bus
    system - Bus 46 Voltage Magnitude MeasB46VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
```

```
    'SS_118:MEASB47VOLTV', 3, NULL, 'SS118-MEASB47VOLTV', 'Shadow System for 118-bus
    system - Bus 47 Voltage Magnitude MeasB47VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB48VOLTV', 3, NULL, 'SS118-MEASB48VOLTV', 'Shadow System for 118-bus
    system - Bus 48 Voltage Magnitude MeasB48VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB49VOLTV', 3, NULL, 'SS118-MEASB49VOLTV', 'Shadow System for 118-bus
    system - Bus 49 Voltage Magnitude MeasB49VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB50VOLTV', 3, NULL, 'SS118-MEASB50VOLTV', 'Shadow System for 118-bus
    system - Bus 50 Voltage Magnitude MeasB50VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB51VOLTV', 3, NULL, 'SS118-MEASB51VOLTV', 'Shadow System for 118-bus
    system - Bus 51 Voltage Magnitude MeasB51VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB52VOLTV', 3, NULL, 'SS118-MEASB52VOLTV', 'Shadow System for 118-bus
    system - Bus 52 Voltage Magnitude MeasB52VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB53VOLTV', 3, NULL, 'SS118-MEASB53VOLTV', 'Shadow System for 118-bus
    system - Bus 53 Voltage Magnitude MeasB53VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB54VOLTV', 3, NULL, 'SS118-MEASB54VOLTV', 'Shadow System for 118-bus
    system - Bus 54 Voltage Magnitude MeasB54VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB55VOLTV', 3, NULL, 'SS118-MEASB55VOLTV', 'Shadow System for 118-bus
    system - Bus 55 Voltage Magnitude MeasB55VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB56VOLTV', 3, NULL, 'SS118-MEASB56VOLTV', 'Shadow System for 118-bus
    system - Bus 56 Voltage Magnitude MeasB56VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB57VOLTV', 3, NULL, 'SS118-MEASB57VOLTV', 'Shadow System for 118-bus
    system - Bus 57 Voltage Magnitude MeasB57VoltV', 1);
```

```
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB58VOLTV', 3, NULL, 'SS118-MEASB58VOLTV', 'Shadow System for 118-bus
    system - Bus 58 Voltage Magnitude MeasB58VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB59VOLTV', 3, NULL, 'SS118-MEASB59VOLTV', 'Shadow System for 118-bus
    system - Bus 59 Voltage Magnitude MeasB59VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB60VOLTV', 3, NULL, 'SS118-MEASB60VOLTV', 'Shadow System for 118-bus
    system - Bus 60 Voltage Magnitude MeasB60VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB61VOLTV', 3, NULL, 'SS118-MEASB61VOLTV', 'Shadow System for 118-bus
    system - Bus 61 Voltage Magnitude MeasB61VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB62VOLTV', 3, NULL, 'SS118-MEASB62VOLTV', 'Shadow System for 118-bus
    system - Bus 62 Voltage Magnitude MeasB62VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB63VOLTV', 3, NULL, 'SS118-MEASB63VOLTV', 'Shadow System for 118-bus
    system - Bus 63 Voltage Magnitude MeasB63VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB64VOLTV', 3, NULL, 'SS118-MEASB64VOLTV', 'Shadow System for 118-bus
    system - Bus 64 Voltage Magnitude MeasB64VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB65VOLTV', 3, NULL, 'SS118-MEASB65VOLTV', 'Shadow System for 118-bus
    system - Bus 65 Voltage Magnitude MeasB65VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB66VOLTV', 3, NULL, 'SS118-MEASB66VOLTV', 'Shadow System for 118-bus
    system - Bus 66 Voltage Magnitude MeasB66VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB67VOLTV', 3, NULL, 'SS118-MEASB67VOLTV', 'Shadow System for 118-bus
    system - Bus 67 Voltage Magnitude MeasB67VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
```

```
    'SS_118:MEASB68VOLTV', 3, NULL, 'SS118-MEASB68VOLTV', 'Shadow System for 118-bus
    system - Bus 68 Voltage Magnitude MeasB68VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB69VOLTV', 3, NULL, 'SS118-MEASB69VOLTV', 'Shadow System for 118-bus
    system - Bus 69 Voltage Magnitude MeasB69VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB70VOLTV', 3, NULL, 'SS118-MEASB70VOLTV', 'Shadow System for 118-bus
    system - Bus 70 Voltage Magnitude MeasB70VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB71VOLTV', 3, NULL, 'SS118-MEASB71VOLTV', 'Shadow System for 118-bus
    system - Bus 71 Voltage Magnitude MeasB71VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB72VOLTV', 3, NULL, 'SS118-MEASB72VOLTV', 'Shadow System for 118-bus
    system - Bus 72 Voltage Magnitude MeasB72VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB73VOLTV', 3, NULL, 'SS118-MEASB73VOLTV', 'Shadow System for 118-bus
    system - Bus 73 Voltage Magnitude MeasB73VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB74VOLTV', 3, NULL, 'SS118-MEASB74VOLTV', 'Shadow System for 118-bus
    system - Bus 74 Voltage Magnitude MeasB74VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB75VOLTV', 3, NULL, 'SS118-MEASB75VOLTV', 'Shadow System for 118-bus
    system - Bus 75 Voltage Magnitude MeasB75VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB76VOLTV', 3, NULL, 'SS118-MEASB76VOLTV', 'Shadow System for 118-bus
    system - Bus 76 Voltage Magnitude MeasB76VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB77VOLTV', 3, NULL, 'SS118-MEASB77VOLTV', 'Shadow System for 118-bus
    system - Bus 77 Voltage Magnitude MeasB77VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB78VOLTV', 3, NULL, 'SS118-MEASB78VOLTV', 'Shadow System for 118-bus
    system - Bus 78 Voltage Magnitude MeasB78VoltV', 1);
```

```
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB79VOLTV', 3, NULL, 'SS118-MEASB79VOLTV', 'Shadow System for 118-bus
    system - Bus 79 Voltage Magnitude MeasB79VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB80VOLTV', 3, NULL, 'SS118-MEASB80VOLTV', 'Shadow System for 118-bus
    system - Bus 80 Voltage Magnitude MeasB80VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB81VOLTV', 3, NULL, 'SS118-MEASB81VOLTV', 'Shadow System for 118-bus
    system - Bus 81 Voltage Magnitude MeasB81VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB82VOLTV', 3, NULL, 'SS118-MEASB82VOLTV', 'Shadow System for 118-bus
    system - Bus 82 Voltage Magnitude MeasB82VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB83VOLTV', 3, NULL, 'SS118-MEASB83VOLTV', 'Shadow System for 118-bus
    system - Bus 83 Voltage Magnitude MeasB83VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB84VOLTV', 3, NULL, 'SS118-MEASB84VOLTV', 'Shadow System for 118-bus
    system - Bus 84 Voltage Magnitude MeasB84VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB85VOLTV', 3, NULL, 'SS118-MEASB85VOLTV', 'Shadow System for 118-bus
    system - Bus 85 Voltage Magnitude MeasB85VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB86VOLTV', 3, NULL, 'SS118-MEASB86VOLTV', 'Shadow System for 118-bus
    system - Bus 86 Voltage Magnitude MeasB86VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB87VOLTV', 3, NULL, 'SS118-MEASB87VOLTV', 'Shadow System for 118-bus
    system - Bus 87 Voltage Magnitude MeasB87VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB88VOLTV', 3, NULL, 'SS118-MEASB88VOLTV', 'Shadow System for 118-bus
    system - Bus 88 Voltage Magnitude MeasB88VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
```

```
    'SS_118:MEASB89VOLTV', 3, NULL, 'SS118-MEASB89VOLTV', 'Shadow System for 118-bus
    system - Bus 89 Voltage Magnitude MeasB89VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB90VOLTV', 3, NULL, 'SS118-MEASB90VOLTV', 'Shadow System for 118-bus
    system - Bus 90 Voltage Magnitude MeasB90VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB91VOLTV', 3, NULL, 'SS118-MEASB91VOLTV', 'Shadow System for 118-bus
    system - Bus 91 Voltage Magnitude MeasB91VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB92VOLTV', 3, NULL, 'SS118-MEASB92VOLTV', 'Shadow System for 118-bus
    system - Bus 92 Voltage Magnitude MeasB92VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB93VOLTV', 3, NULL, 'SS118-MEASB93VOLTV', 'Shadow System for 118-bus
    system - Bus 93 Voltage Magnitude MeasB93VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB94VOLTV', 3, NULL, 'SS118-MEASB94VOLTV', 'Shadow System for 118-bus
    system - Bus 94 Voltage Magnitude MeasB94VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB95VOLTV', 3, NULL, 'SS118-MEASB95VOLTV', 'Shadow System for 118-bus
    system - Bus 95 Voltage Magnitude MeasB95VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB96VOLTV', 3, NULL, 'SS118-MEASB96VOLTV', 'Shadow System for 118-bus
    system - Bus 96 Voltage Magnitude MeasB96VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB97VOLTV', 3, NULL, 'SS118-MEASB97VOLTV', 'Shadow System for 118-bus
    system - Bus 97 Voltage Magnitude MeasB97VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB98VOLTV', 3, NULL, 'SS118-MEASB98VOLTV', 'Shadow System for 118-bus
    system - Bus 98 Voltage Magnitude MeasB98VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB99VOLTV', 3, NULL, 'SS118-MEASB99VOLTV', 'Shadow System for 118-bus
    system - Bus 99 Voltage Magnitude MeasB99VoltV', 1);
```

```
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB100VOLTV', 3, NULL, 'SS118-MEASB100VOLTV', 'Shadow System for
    118-bus system - Bus 100 Voltage Magnitude MeasB100VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB101VOLTV', 3, NULL, 'SS118-MEASB101VOLTV', 'Shadow System for
    118-bus system - Bus 101 Voltage Magnitude MeasB101VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB102VOLTV', 3, NULL, 'SS118-MEASB102VOLTV', 'Shadow System for
    118-bus system - Bus 102 Voltage Magnitude MeasB102VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB103VOLTV', 3, NULL, 'SS118-MEASB103VOLTV', 'Shadow System for
    118-bus system - Bus 103 Voltage Magnitude MeasB103VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB104VOLTV', 3, NULL, 'SS118-MEASB104VOLTV', 'Shadow System for
    118-bus system - Bus 104 Voltage Magnitude MeasB104VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB105VOLTV', 3, NULL, 'SS118-MEASB105VOLTV', 'Shadow System for
    118-bus system - Bus 105 Voltage Magnitude MeasB105VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB106VOLTV', 3, NULL, 'SS118-MEASB106VOLTV', 'Shadow System for
    118-bus system - Bus 106 Voltage Magnitude MeasB106VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB107VOLTV', 3, NULL, 'SS118-MEASB107VOLTV', 'Shadow System for
    118-bus system - Bus 107 Voltage Magnitude MeasB107VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB108VOLTV', 3, NULL, 'SS118-MEASB108VOLTV', 'Shadow System for
    118-bus system - Bus 108 Voltage Magnitude MeasB108VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB109VOLTV', 3, NULL, 'SS118-MEASB109VOLTV', 'Shadow System for
    118-bus system - Bus 109 Voltage Magnitude MeasB109VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
```

```
    'SS_118:MEASB110VOLTV', 3, NULL, 'SS118-MEASB110VOLTV', 'Shadow System for
    118-bus system - Bus 110 Voltage Magnitude MeasB110VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB111VOLTV', 3, NULL, 'SS118-MEASB111VOLTV', 'Shadow System for
    118-bus system - Bus 111 Voltage Magnitude MeasB111VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB112VOLTV', 3, NULL, 'SS118-MEASB112VOLTV', 'Shadow System for
    118-bus system - Bus 112 Voltage Magnitude MeasB112VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB113VOLTV', 3, NULL, 'SS118-MEASB113VOLTV', 'Shadow System for
    118-bus system - Bus 113 Voltage Magnitude MeasB113VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB114VOLTV', 3, NULL, 'SS118-MEASB114VOLTV', 'Shadow System for
    118-bus system - Bus 114 Voltage Magnitude MeasB114VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB115VOLTV', 3, NULL, 'SS118-MEASB115VOLTV', 'Shadow System for
    118-bus system - Bus 115 Voltage Magnitude MeasB115VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB116VOLTV', 3, NULL, 'SS118-MEASB116VOLTV', 'Shadow System for
    118-bus system - Bus 116 Voltage Magnitude MeasB116VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB117VOLTV', 3, NULL, 'SS118-MEASB117VOLTV', 'Shadow System for
    118-bus system - Bus 117 Voltage Magnitude MeasB117VoltV', 1);
INSERT INTO Measurement(HistorianID, DeviceID, PointTag, SignalTypeID,
    PhasorSourceIndex, SignalReference, Description, Enabled) VALUES(1, 1,
    'SS_118:MEASB118VOLTV', 3, NULL, 'SS118-MEASB118VOLTV', 'Shadow System for
    118-bus system - Bus 118 Voltage Magnitude MeasB118VoltV', 1);


INSERT INTO ApplicationRole (Name, Description, NodeID) VALUES ('Administrator',
    'Administrator Role', 'e57b4a6d-ca9e-403c-ad2e-9a1db9a8a707');
INSERT INTO ApplicationRole (Name, Description, NodeID) VALUES ('Editor', 'Editor',
    'e57b4a6d-ca9e-403c-ad2e-9a1db9a8a707');
INSERT INTO ApplicationRole (Name, Description, NodeID) VALUES ('Viewer', 'Viewer
    Role', 'e57b4a6d-ca9e-403c-ad2e-9a1db9a8a707');
```

B.2 Manage Data Mappings

Shadow System Simulator

```
SS118Data Inputs SS118Data_InputMapping {
    ResetSignal: PPA:62
    LoadIncrementPercentage: PPA:41
    ActTxRaise: PPA:42
    ActTxLower: PPA:43
    ActSn1Close: PPA:44
    ActSn1Trip: PPA:45
    ActSn2Close: PPA:46
    ActSn2Trip: PPA:47
    ActSnB34Close: PPA:63
    ActSnB44Close: PPA:64
    ActSnB45Close: PPA:65
    ActSnB48Close: PPA:66
    ActSnB74Close: PPA:67
    ActSnB105Close: PPA:68
    ActSnB34Trip: PPA:69
    ActSnB44Trip: PPA:70
    ActSnB45Trip: PPA:71
    ActSnB48Trip: PPA:72
    ActSnB74Trip: PPA:73
    ActSnB105Trip: PPA:74
}

SS118Data Outputs SS118Data_OutputMapping {
    StateTxTapV: PPA:48
    StateSn1CapBkrV: PPA:49
    StateSn2CapBkrV: PPA:50
    StateSn1BusBkrV: PPA:51
    StateSn2BusBkrV: PPA:52
    MeasTxVoltV: PPA:53
    MeasSn1VoltV: PPA:54
    MeasSn2VoltV: PPA:55
    MeasTxMwV: PPA:56
    MeasTxMvrV: PPA:57
    MeasGn1MwV: PPA:58
    MeasGn1MvrV: PPA:59
    MeasGn2MwV: PPA:60
```

```
MeasGn2MvrV: PPA:61
StateSnB34CapBkrV: PPA:75
StateSnB44CapBkrV: PPA:76
StateSnB45CapBkrV: PPA:77
StateSnB48CapBkrV: PPA:78
StateSnB74CapBkrV: PPA:79
StateSnB105CapBkrV: PPA:80
MeasB1VoltV: PPA:81
MeasB2VoltV: PPA:82
MeasB3VoltV: PPA:83
MeasB4VoltV: PPA:84
MeasB5VoltV: PPA:85
MeasB6VoltV: PPA:86
MeasB7VoltV: PPA:87
MeasB8VoltV: PPA:88
MeasB9VoltV: PPA:89
MeasB10VoltV: PPA:90
MeasB11VoltV: PPA:91
MeasB12VoltV: PPA:92
MeasB13VoltV: PPA:93
MeasB14VoltV: PPA:94
MeasB15VoltV: PPA:95
MeasB16VoltV: PPA:96
MeasB17VoltV: PPA:97
MeasB18VoltV: PPA:98
MeasB19VoltV: PPA:99
MeasB20VoltV: PPA:100
MeasB21VoltV: PPA:101
MeasB22VoltV: PPA:102
MeasB23VoltV: PPA:103
MeasB24VoltV: PPA:104
MeasB25VoltV: PPA:105
MeasB26VoltV: PPA:106
MeasB27VoltV: PPA:107
MeasB28VoltV: PPA:108
MeasB29VoltV: PPA:109
MeasB30VoltV: PPA:110
MeasB31VoltV: PPA:111
MeasB32VoltV: PPA:112
MeasB33VoltV: PPA:113
MeasB34VoltV: PPA:114
MeasB35VoltV: PPA:115
```

```
MeasB36VoltV: PPA:116
MeasB37VoltV: PPA:117
MeasB38VoltV: PPA:118
MeasB39VoltV: PPA:119
MeasB40VoltV: PPA:120
MeasB41VoltV: PPA:121
MeasB42VoltV: PPA:122
MeasB43VoltV: PPA:123
MeasB44VoltV: PPA:124
MeasB45VoltV: PPA:125
MeasB46VoltV: PPA:126
MeasB47VoltV: PPA:127
MeasB48VoltV: PPA:128
MeasB49VoltV: PPA:129
MeasB50VoltV: PPA:130
MeasB51VoltV: PPA:131
MeasB52VoltV: PPA:132
MeasB53VoltV: PPA:133
MeasB54VoltV: PPA:134
MeasB55VoltV: PPA:135
MeasB56VoltV: PPA:136
MeasB57VoltV: PPA:137
MeasB58VoltV: PPA:138
MeasB59VoltV: PPA:139
MeasB60VoltV: PPA:140
MeasB61VoltV: PPA:141
MeasB62VoltV: PPA:142
MeasB63VoltV: PPA:143
MeasB64VoltV: PPA:144
MeasB65VoltV: PPA:145
MeasB66VoltV: PPA:146
MeasB67VoltV: PPA:147
MeasB68VoltV: PPA:148
MeasB69VoltV: PPA:149
MeasB70VoltV: PPA:150
MeasB71VoltV: PPA:151
MeasB72VoltV: PPA:152
MeasB73VoltV: PPA:153
MeasB74VoltV: PPA:154
MeasB75VoltV: PPA:155
MeasB76VoltV: PPA:156
MeasB77VoltV: PPA:157
```

```
    MeasB78VoltV: PPA:158
    MeasB79VoltV: PPA:159
    MeasB80VoltV: PPA:160
    MeasB81VoltV: PPA:161
    MeasB82VoltV: PPA:162
    MeasB83VoltV: PPA:163
    MeasB84VoltV: PPA:164
    MeasB85VoltV: PPA:165
    MeasB86VoltV: PPA:166
    MeasB87VoltV: PPA:167
    MeasB88VoltV: PPA:168
    MeasB89VoltV: PPA:169
    MeasB90VoltV: PPA:170
    MeasB91VoltV: PPA:171
    MeasB92VoltV: PPA:172
    MeasB93VoltV: PPA:173
    MeasB94VoltV: PPA:174
    MeasB95VoltV: PPA:175
    MeasB96VoltV: PPA:176
    MeasB97VoltV: PPA:177
    MeasB98VoltV: PPA:178
    MeasB99VoltV: PPA:179
    MeasB100VoltV: PPA:180
    MeasB101VoltV: PPA:181
    MeasB102VoltV: PPA:182
    MeasB103VoltV: PPA:183
    MeasB104VoltV: PPA:184
    MeasB105VoltV: PPA:185
    MeasB106VoltV: PPA:186
    MeasB107VoltV: PPA:187
    MeasB108VoltV: PPA:188
    MeasB109VoltV: PPA:189
    MeasB110VoltV: PPA:190
    MeasB111VoltV: PPA:191
    MeasB112VoltV: PPA:192
    MeasB113VoltV: PPA:193
    MeasB114VoltV: PPA:194
    MeasB115VoltV: PPA:195
    MeasB116VoltV: PPA:196
    MeasB117VoltV: PPA:197
    MeasB118VoltV: PPA:198
}
```

Online VSA Applications

```
RVC118Data Inputs RVC118Data_InputMapping {
    StateSnB34CapBkrV: PPA:75
    StateSnB44CapBkrV: PPA:76
    StateSnB45CapBkrV: PPA:77
    StateSnB48CapBkrV: PPA:78
    StateSnB74CapBkrV: PPA:79
    StateSnB105CapBkrV: PPA:80
    MeasB1VoltV: PPA:81
    MeasB2VoltV: PPA:82
    MeasB3VoltV: PPA:83
    MeasB4VoltV: PPA:84
    MeasB5VoltV: PPA:85
    MeasB6VoltV: PPA:86
    MeasB7VoltV: PPA:87
    MeasB8VoltV: PPA:88
    MeasB9VoltV: PPA:89
    MeasB10VoltV: PPA:90
    MeasB11VoltV: PPA:91
    MeasB12VoltV: PPA:92
    MeasB13VoltV: PPA:93
    MeasB14VoltV: PPA:94
    MeasB15VoltV: PPA:95
    MeasB16VoltV: PPA:96
    MeasB17VoltV: PPA:97
    MeasB18VoltV: PPA:98
    MeasB19VoltV: PPA:99
    MeasB20VoltV: PPA:100
    MeasB21VoltV: PPA:101
    MeasB22VoltV: PPA:102
    MeasB23VoltV: PPA:103
    MeasB24VoltV: PPA:104
    MeasB25VoltV: PPA:105
    MeasB26VoltV: PPA:106
    MeasB27VoltV: PPA:107
    MeasB28VoltV: PPA:108
    MeasB29VoltV: PPA:109
    MeasB30VoltV: PPA:110
    MeasB31VoltV: PPA:111
    MeasB32VoltV: PPA:112
    MeasB33VoltV: PPA:113
```

```
MeasB34VoltV: PPA:114
MeasB35VoltV: PPA:115
MeasB36VoltV: PPA:116
MeasB37VoltV: PPA:117
MeasB38VoltV: PPA:118
MeasB39VoltV: PPA:119
MeasB40VoltV: PPA:120
MeasB41VoltV: PPA:121
MeasB42VoltV: PPA:122
MeasB43VoltV: PPA:123
MeasB44VoltV: PPA:124
MeasB45VoltV: PPA:125
MeasB46VoltV: PPA:126
MeasB47VoltV: PPA:127
MeasB48VoltV: PPA:128
MeasB49VoltV: PPA:129
MeasB50VoltV: PPA:130
MeasB51VoltV: PPA:131
MeasB52VoltV: PPA:132
MeasB53VoltV: PPA:133
MeasB54VoltV: PPA:134
MeasB55VoltV: PPA:135
MeasB56VoltV: PPA:136
MeasB57VoltV: PPA:137
MeasB58VoltV: PPA:138
MeasB59VoltV: PPA:139
MeasB60VoltV: PPA:140
MeasB61VoltV: PPA:141
MeasB62VoltV: PPA:142
MeasB63VoltV: PPA:143
MeasB64VoltV: PPA:144
MeasB65VoltV: PPA:145
MeasB66VoltV: PPA:146
MeasB67VoltV: PPA:147
MeasB68VoltV: PPA:148
MeasB69VoltV: PPA:149
MeasB70VoltV: PPA:150
MeasB71VoltV: PPA:151
MeasB72VoltV: PPA:152
MeasB73VoltV: PPA:153
MeasB74VoltV: PPA:154
MeasB75VoltV: PPA:155
```

```
MeasB76VoltV: PPA:156
MeasB77VoltV: PPA:157
MeasB78VoltV: PPA:158
MeasB79VoltV: PPA:159
MeasB80VoltV: PPA:160
MeasB81VoltV: PPA:161
MeasB82VoltV: PPA:162
MeasB83VoltV: PPA:163
MeasB84VoltV: PPA:164
MeasB85VoltV: PPA:165
MeasB86VoltV: PPA:166
MeasB87VoltV: PPA:167
MeasB88VoltV: PPA:168
MeasB89VoltV: PPA:169
MeasB90VoltV: PPA:170
MeasB91VoltV: PPA:171
MeasB92VoltV: PPA:172
MeasB93VoltV: PPA:173
MeasB94VoltV: PPA:174
MeasB95VoltV: PPA:175
MeasB96VoltV: PPA:176
MeasB97VoltV: PPA:177
MeasB98VoltV: PPA:178
MeasB99VoltV: PPA:179
MeasB100VoltV: PPA:180
MeasB101VoltV: PPA:181
MeasB102VoltV: PPA:182
MeasB103VoltV: PPA:183
MeasB104VoltV: PPA:184
MeasB105VoltV: PPA:185
MeasB106VoltV: PPA:186
MeasB107VoltV: PPA:187
MeasB108VoltV: PPA:188
MeasB109VoltV: PPA:189
MeasB110VoltV: PPA:190
MeasB111VoltV: PPA:191
MeasB112VoltV: PPA:192
MeasB113VoltV: PPA:193
MeasB114VoltV: PPA:194
MeasB115VoltV: PPA:195
MeasB116VoltV: PPA:196
MeasB117VoltV: PPA:197
```

```
    MeasB118VoltV: PPA:198
}

RVC118Data Outputs RVC118Data_OutputMapping {
    ActSnB34Close: PPA:63
    ActSnB44Close: PPA:64
    ActSnB45Close: PPA:65
    ActSnB48Close: PPA:66
    ActSnB74Close: PPA:67
    ActSnB105Close: PPA:68
    ActSnB34Trip: PPA:69
    ActSnB44Trip: PPA:70
    ActSnB45Trip: PPA:71
    ActSnB48Trip: PPA:72
    ActSnB74Trip: PPA:73
    ActSnB105Trip: PPA:74
}
```