Towards Safety-critical Commercial-grade Artificial Intelligence to Enhance Grid Reliability and Cybersecurity

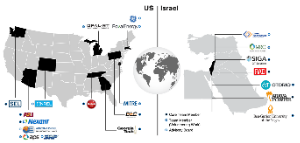
BIRD Presentation—Tasks 5 and 8

Lalitha Sankar
Professor, ECEE Dept.
Arizona State University

October 2023

Arizona State University

ASU Team Members



Obai Bahwal
3rd Year PhD Student

Rajasekhar Anguluri
Postdoctoral Scholar

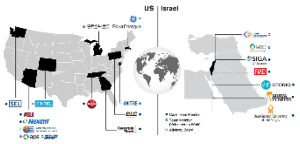
Joel Mathias
Postdoctoral Scholar

Nima T. Bazargani
PhD, Sept. 2023

Avinash Kodali
MS Student

Also collaborating with John Dirkman, Narsi Vempati, Guanji Hou, Roozbeh Emami @ Resource Innovations Inc.

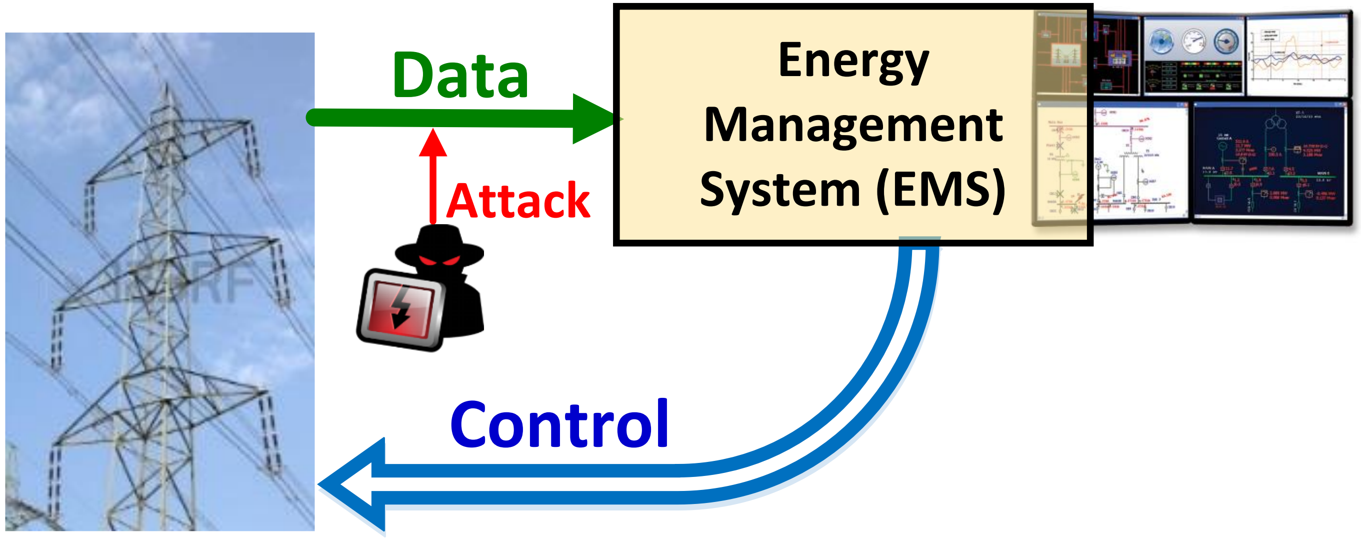
Enhancing Cybersecurity of Grid Operations



- Task 5: Generate event-mimicking attacks
- Task 8: Detect event-mimicking attacks
- Key focus on Commercialization
  - ➢ Commercial-grade software development with Resource Innovations
  - ➢ Load Prediction using Support Vector Regression
  - ➢ Attack Detection and Mitigation using Support Vector Machines

Commercialization: Motivation

False Data Injection Attacks and Countermeasures

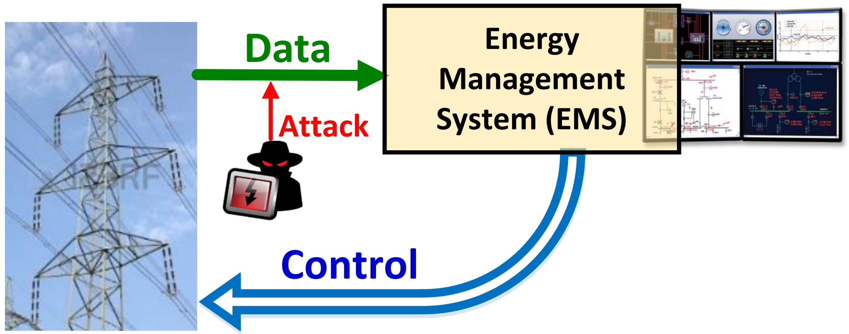


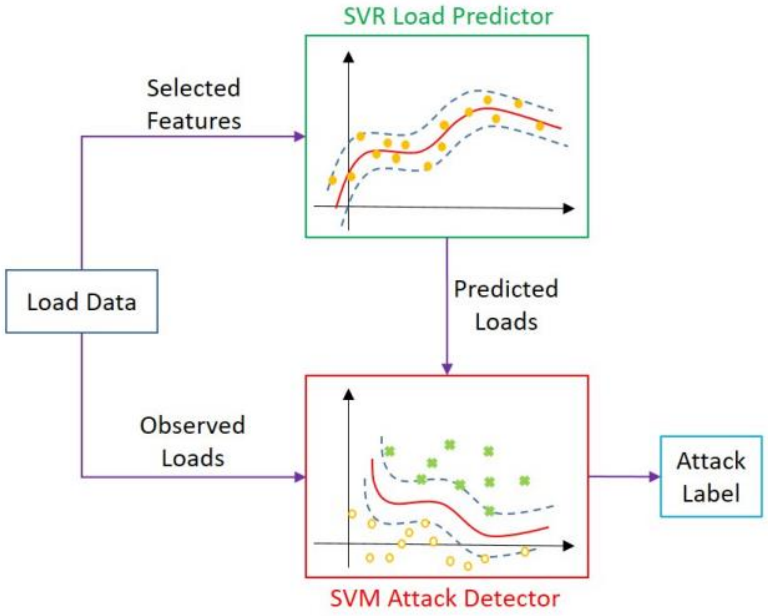
- Knowing network configuration, attackers can maliciously change a subset of measurements with counterfeits before they reach the EMS
- Requires attacker to have access to measurement devices or data concentrators
- Can be unobservable and result in physical [2] / economic [3] consequences

[1] Zhang, J., Sankar, L.: ‘Physical system consequences of unobservable state-and-topology cyber-physical attacks’, IEEE Transactions on Smart Grid, 2016, 7, (4), pp. 2016–2025
[2] Moslemi, R., Mesbahi, A., Velni, J.M.: ‘Design of robust profitable false data injection attacks in multi-settlement electricity markets’, IET Generation, Transmission Distribution, 2018, 12, (6), pp. 1263–1270
[3] Liang, J., Sankar, L., Kosut O.: ‘Vulnerability analysis and consequences of false data injection attack on power system state estimation’, IEEE Transactions on Power Systems, 2015, 31, (5), pp. 3864-72

Detecting Load Redistribution Attacks via Support Vector Models

- Load Redistribution (LR) attacks: redistribute loads across buses without any change in net load
  - Current net load prediction approaches can miss this entire class of false data injection attacks (FDIA)
- Our detection methodology:
  - Grid telemetry including loads follow diurnal and seasonal patterns
  - Historical data can be used to predict such patterns
  - ML algorithms trained on such temporally correlated data can be used to predict loads at the bus-level
- Use multi-output support vector regression (SVR) load predictor
  - predicts loads by exploiting both spatial and temporal correlations
- Combine with a support vector machine (SVM) classifier to classify incoming load estimate as either normative or attacked



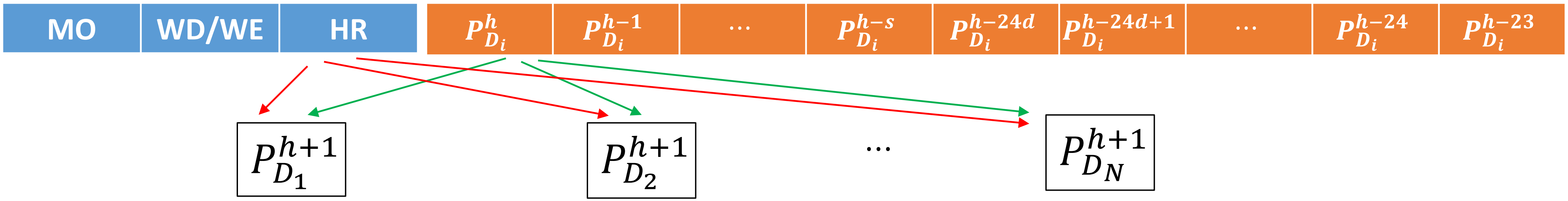


Commercialization: Load Prediction using SVR

Load Prediction using Support Vector Regression

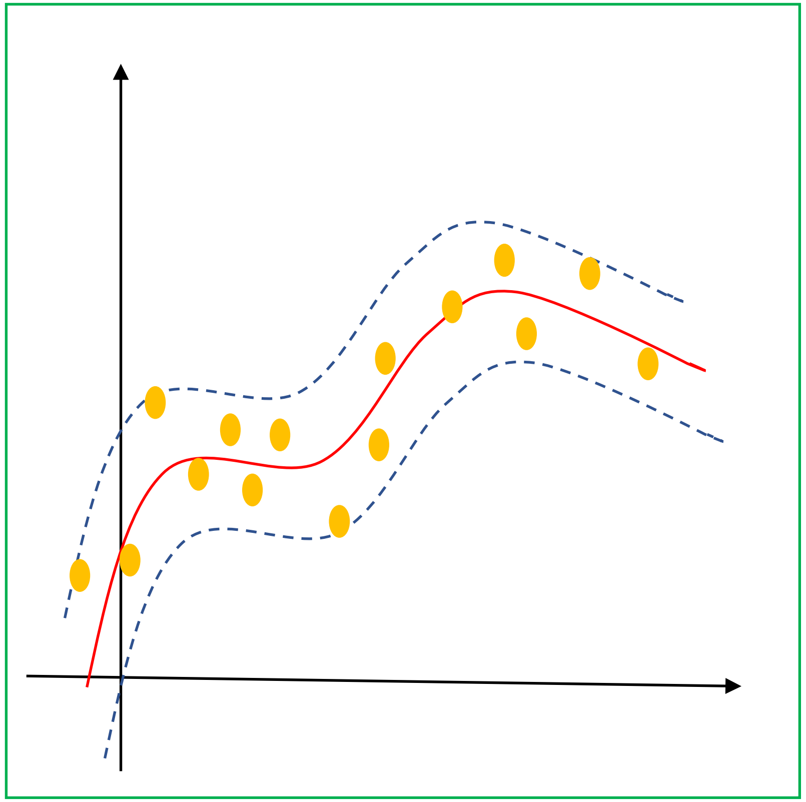
- Learn a support vector regression model for each load bus
  - Feature set can include temporal and spatial/network correlations

- Feature selection to predict load at hour $h+1$
  - Time information
  - Historical load values at past *s* hours, as well as at hour HR and HR+1 at past *d* days
  - Combine these values for multiple loads to capture spatial correlations
  - Can be applied to predict bus level loads



Support Vector Regression (SVR)

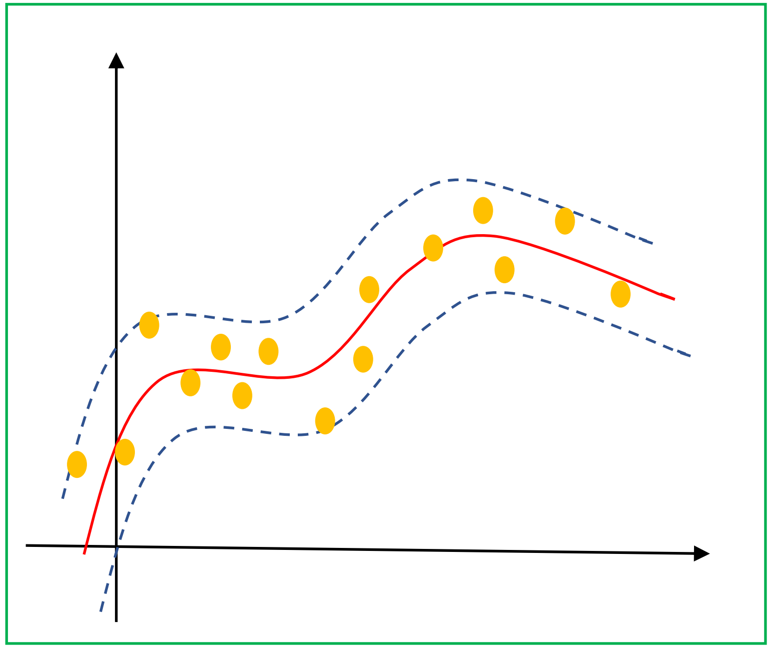
- Find a map between the input variables and a continuous target variable which minimizes the prediction error.
- Involves finding a hyperplane in the higher dimension space that fits the data points in the regression task.
- “Kernel trick” allows for non-linear relationships: maps inputs to a high-dimensional.



Support Vector Regression (SVR)

Finding $\boldsymbol{w}_r$ and b_r that satisfies

$$|y_j - \boldsymbol{w}_r^T \boxed{\phi(\boldsymbol{x}_j)} - b_r| \leq \varepsilon$$



$$\begin{aligned}
\underset{\boldsymbol{w}_r, b_r, \zeta_j, \zeta_j'}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{w}_r^T \boldsymbol{w}_r + M \sum_{j=1}^{n} (\zeta_j + \zeta_j') \\
\text{subject to} \quad & y_j - \boldsymbol{w}_r^T \phi(\boldsymbol{x}_j) - b_r \leq \varepsilon + \zeta_j \quad (\alpha_j) \\
& \boldsymbol{w}_r^T \phi(\boldsymbol{x}_j) + b_r - y_j \leq \varepsilon + \zeta_j' \quad (\alpha_j') \\
& \zeta_j, \zeta_j' \geq 0, \forall j,
\end{aligned}$$

Primal

$$\begin{aligned}
\underset{\boldsymbol{\alpha}, \boldsymbol{\alpha}'}{\text{minimize}} \quad & \frac{1}{2}(\boldsymbol{\alpha} - \boldsymbol{\alpha}')^T \boldsymbol{Q} (\boldsymbol{\alpha} - \boldsymbol{\alpha}') \\
& + \varepsilon \mathbf{1}^T (\boldsymbol{\alpha} + \boldsymbol{\alpha}') - y^T (\boldsymbol{\alpha} - \boldsymbol{\alpha}') \\
\text{subject to} \quad & \mathbf{1}^T (\boldsymbol{\alpha} - \boldsymbol{\alpha}') = 0 \\
& 0 \leq \alpha_j, \alpha_j' \leq M, \forall j
\end{aligned}$$

Dual

RBF Kernel

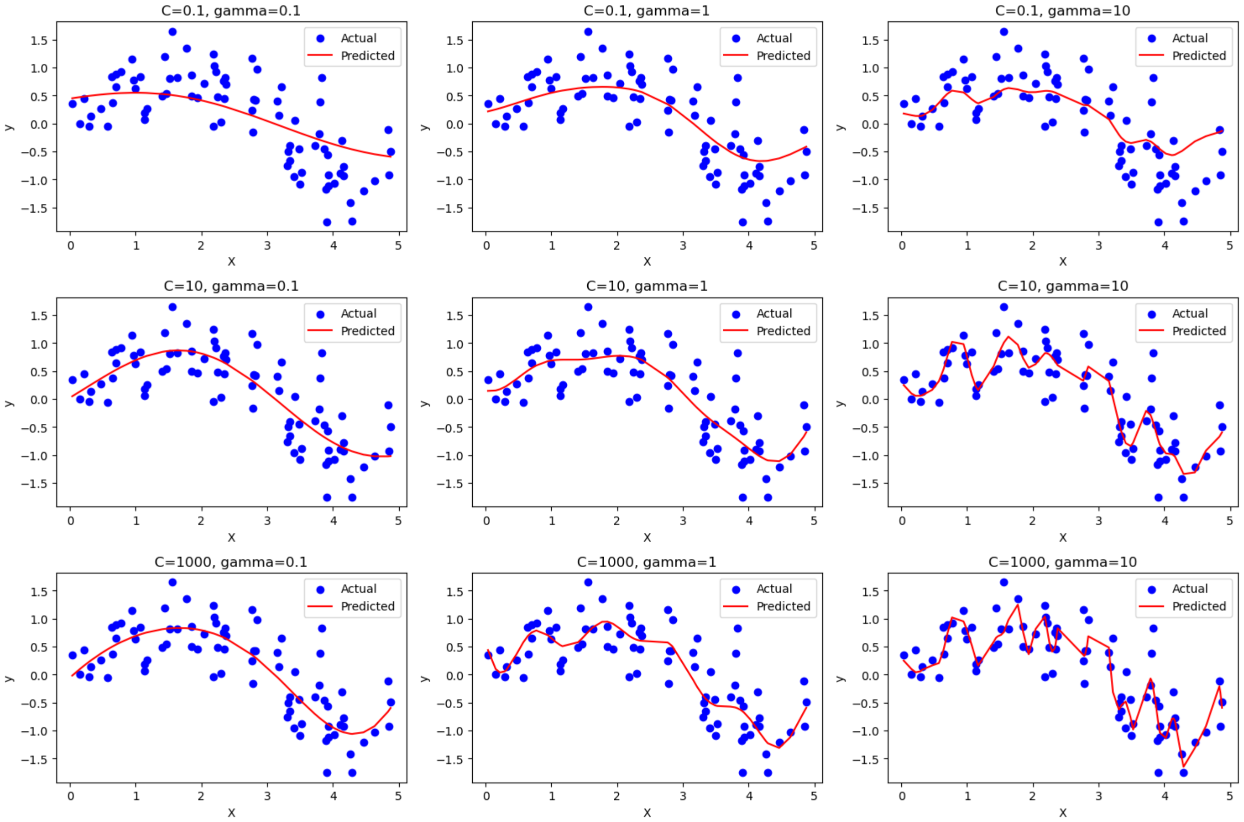
$$Q(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$$

$$Q_{ij} = Q(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$$

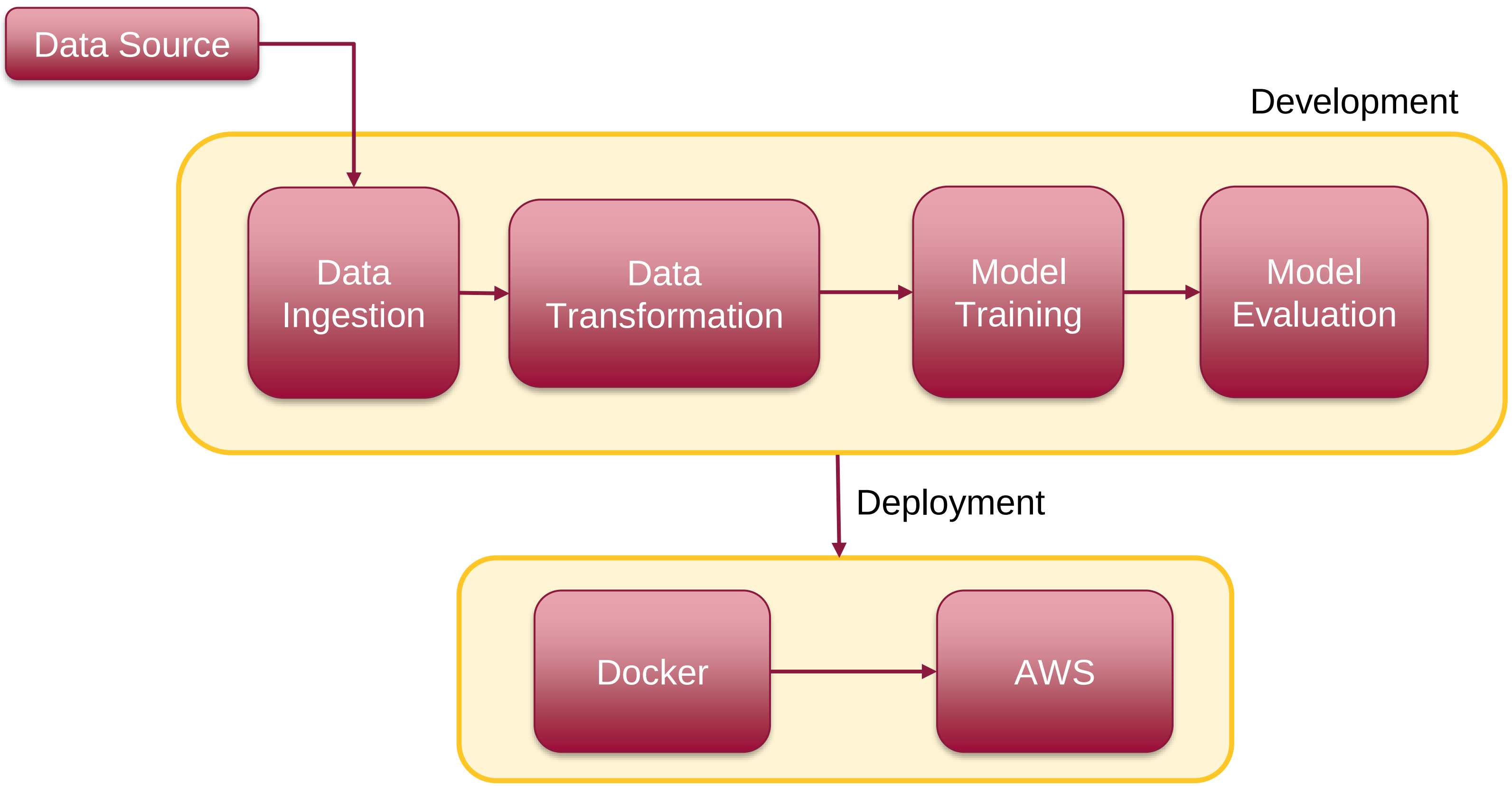
$$y_{\text{new}} = \sum_{j=1}^{n} (\alpha_j^* - \alpha_j'^*) Q(\boldsymbol{x}_j, \boldsymbol{x}_{\text{new}})$$

Support Vector Regression Hyperparameters

- ➢Kernel: Radial Basis Function (RBF)
- ➢*C* or *M* (Regularization parameter): Trade-off between training error and model complexity
- ➢Epsilon (ϵ) : Tolerance for error; higher values allow for more errors, reducing overfitting
- ➢Gamma (γ) : Controls standard deviation of the RBF kernel; higher values for more noisy data



Code Workflow



Code Workflow: Development phase

Data Ingestion
Fetch data from GitHub

Data Transformation
Generate features and standardize the data

MO	WD/WE	HR

$P_{D_i}^{h}$	$P_{D_i}^{h-1}$	…	$P_{D_i}^{h-s}$	$P_{D_i}^{h-24d}$	$P_{D_i}^{h-24d+1}$	…	$P_{D_i}^{h-24}$	$P_{D_i}^{h-23}$

Model Training
80-20 train test split

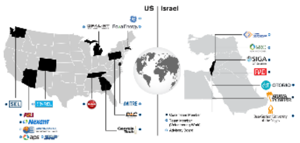
Gridsearch CV using Time series split

Model Evaluation

$$R^2\ Score = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\bar{y}_i - y_i)^2}$$

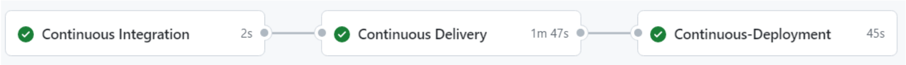
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

Code Workflow: Deployment



Docker

Github CI-CD pipeline



- Continuous Delivery – Create a docker image and push it to AWS ECR (docker container registry service)
  - Docker allows you to package applications and their dependencies into portable containers.
  - It ensures consistent and efficient deployment across different environments.
- Continuous Deployment – Pull the latest docker image and run it on docker container using AWS EC2 (virtual server in cloud).

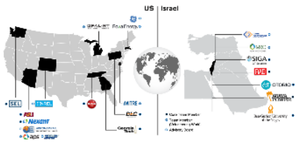
Amazon EC2

Amazon ECR

AWS

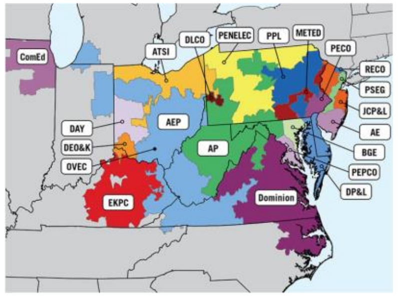
Deployed the Flask web application on AWS

Datasets for Performance Evaluation



PJM

- 19 Load Buses
- Dataset – 2015 to 2018
- Sample frequency – 1hr

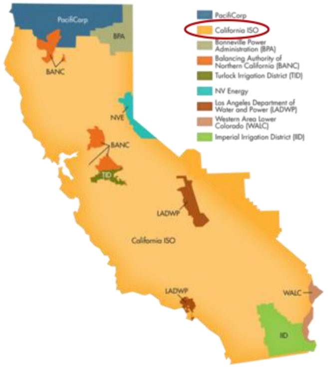


Texas Bus System

- 1347 Load Buses
- Dataset – 2016
- Sample frequency – 1hr

CAISO

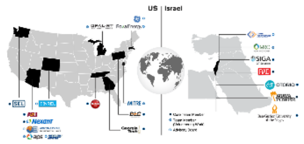
- 30 Load Buses
- Dataset – 2021 to 2023
- Sample frequency - 1hr



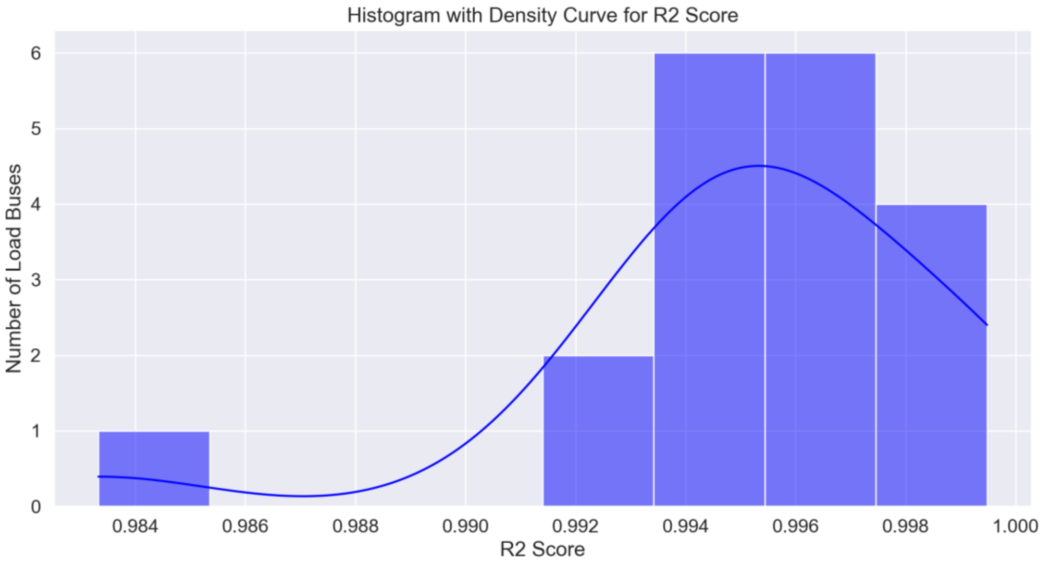
Results (PJM)

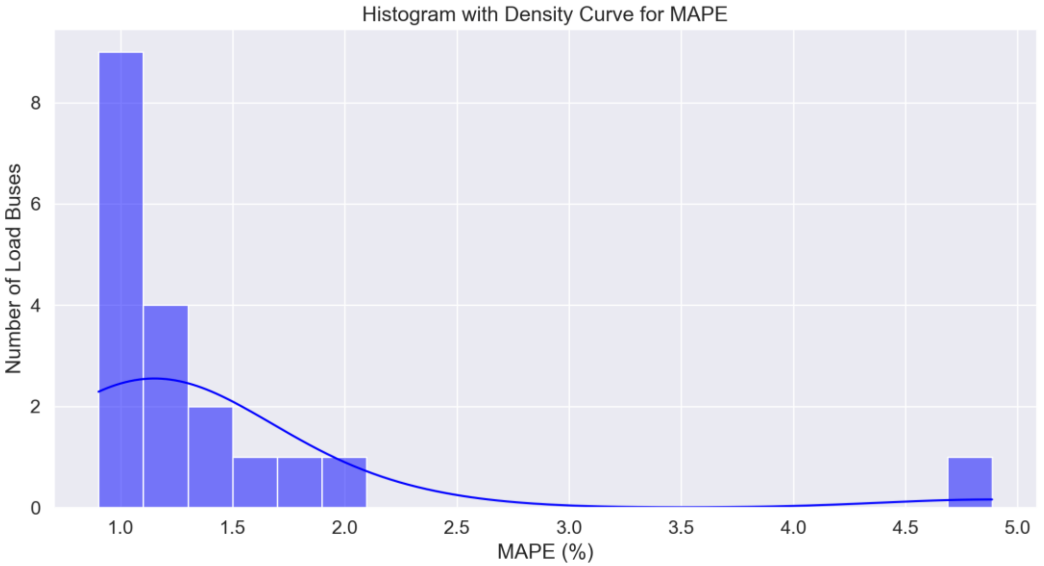
PJM

- 19 Load Buses
- Dataset – 2015 to 2018
- Sample frequency – 1hr



- $R^2\ Score = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(\bar{y}_i - y_i)^2}$
- Ideal R^2 Score is 1.
- R^2 Score for the load buses is above 0.95
- $MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|$
- $MAPE$ for the load buses is concentrated around 1%





Results (Texas)

Texas Bus System

- 1347 Load Buses
- Dataset – 2016
- Sample frequency – 1hr

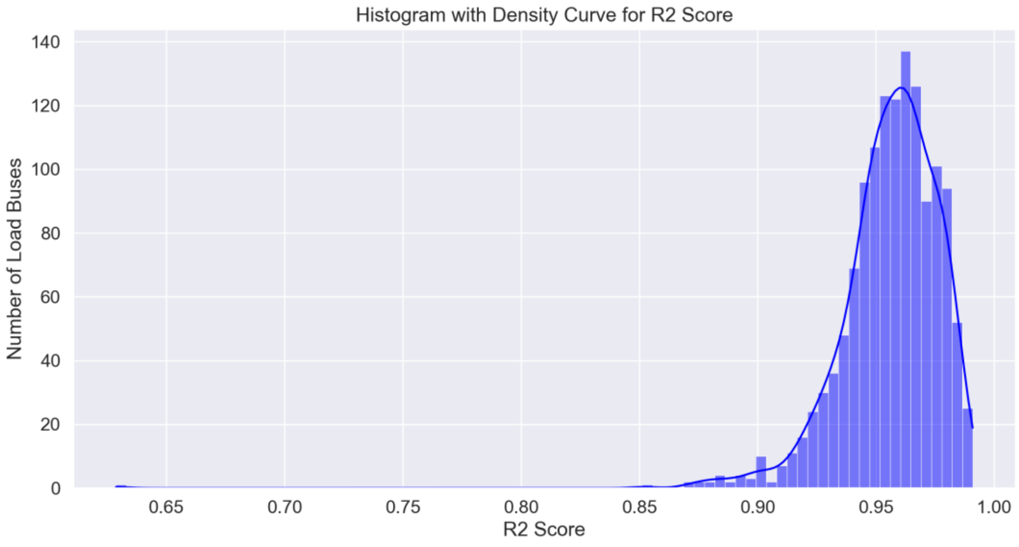
- $R^2\ Score = 1 - \frac{\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2}{\sum_{i=1}^{n}(\overline{y_i} - y_i)^2}$

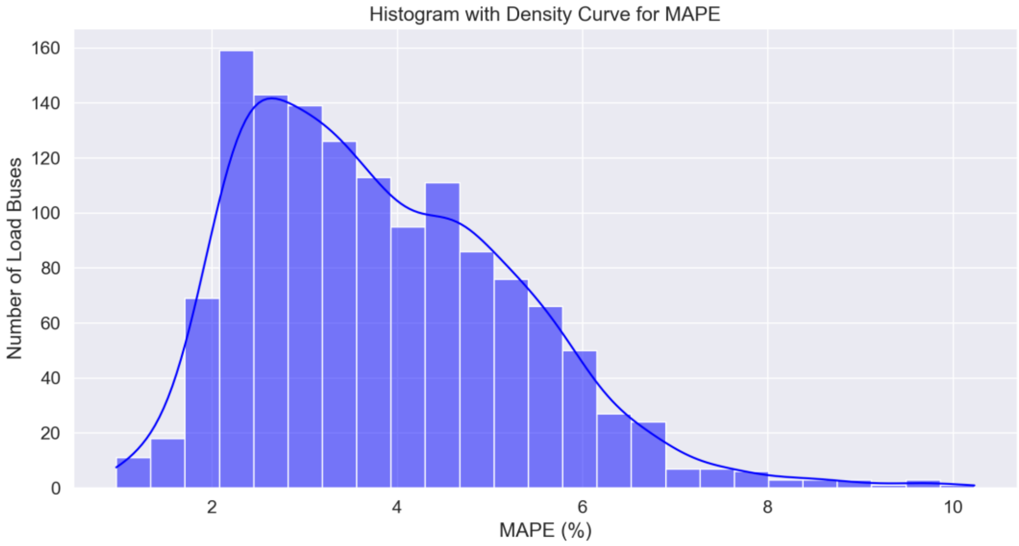
- R^2 Score for majority of load buses is above 0.95

- $MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\widehat{y_i} - y_i}{y_i}\right|$

- $MAPE$ for the load buses is concentrated around 3%

- Reason for higher MAPE: Lower number of training samples, relative to the number of load buses

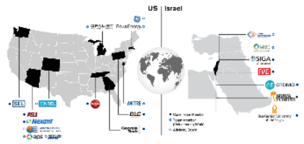




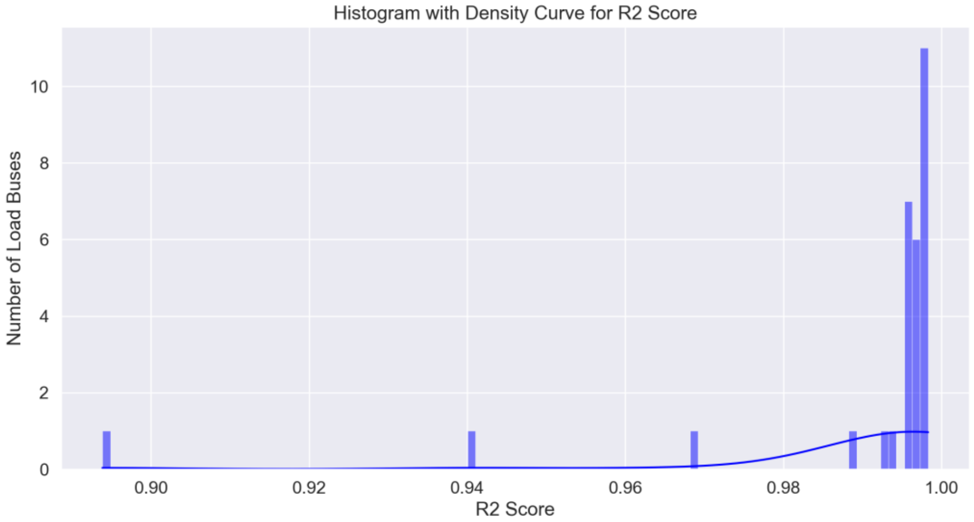
Results (CAISO)

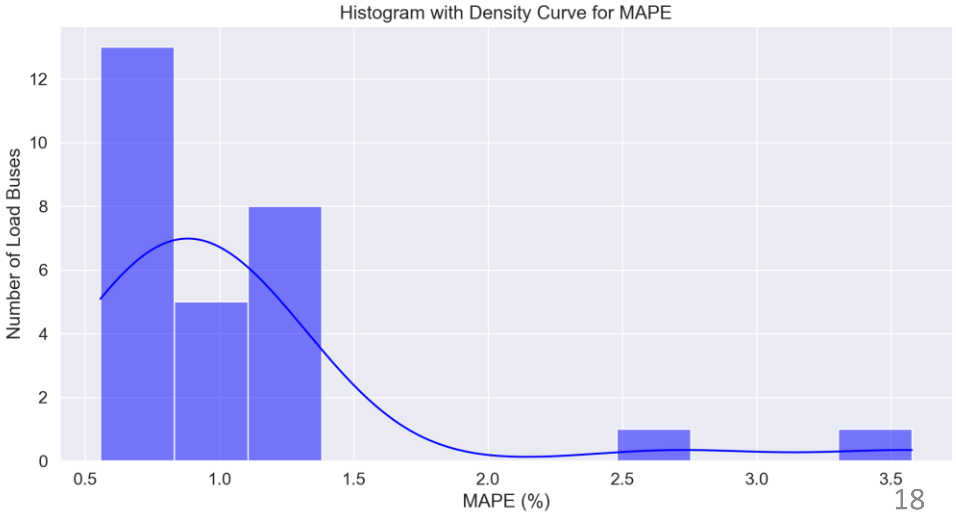
CAISO

- 30 Load Buses
- Dataset – 2021 to 2023
- Sample frequency - 1hr

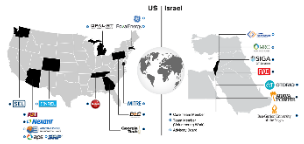


- $R^2\ Score = 1 - \frac{\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2}{\sum_{i=1}^{n}(\overline{y_i} - y_i)^2}$
- R^2 Score for majority of load buses is above 0.95
- $MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\widehat{y_i} - y_i}{y_i}\right|$
- $MAPE$ for majority of the buses is concentrated around 1%

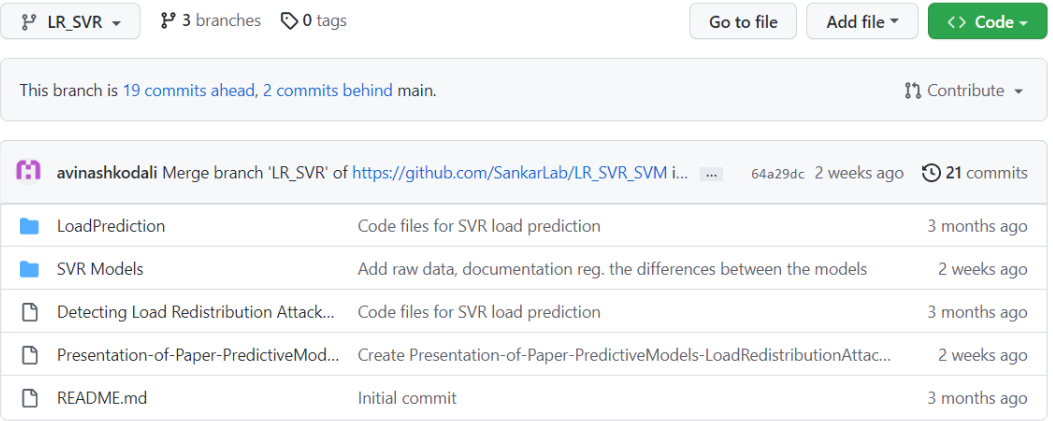


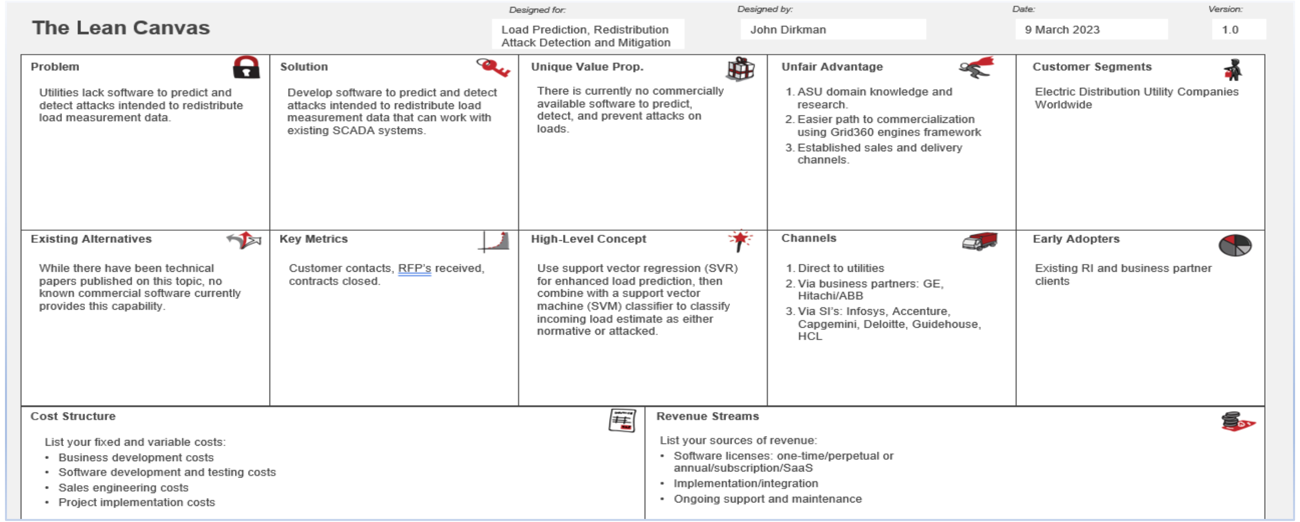


Commercialization: Development with Resource Innovations, Inc.



- ➢ Modularized and documented python code handed-off to RI
- ➢ Version control using GitHub throughout the project, enabling efficient tracking and management of code changes
- ➢ Continuous development methodology for the load prediction and attack detection: bi-weekly progress tracking
- ➢ RI has performed extensive testing on different datasets, including PJM, CAISO, and TX-2000 bus system with highly promising results
- ➢ RI has contacted industry partners and EMS vendors
  - ➢ Bus-level load prediction is crucial with fast-increasing distributed energy resources





The Lean Canvas

Designed for: Load Prediction, Redistribution Attack Detection and Mitigation
Designed by: John Dirkman
Date: 9 March 2023
Version: 1.0

Problem
Utilities lack software to predict and detect attacks intended to redistribute load measurement data.

Solution
Develop software to predict and detect attacks intended to redistribute load measurement data that can work with existing SCADA systems.

Unique Value Prop.
There is currently no commercially available software to predict, detect, and prevent attacks on loads.

Unfair Advantage
1. ASU domain knowledge and research.
2. Easier path to commercialization using Grid360 engines framework
3. Established sales and delivery channels.

Customer Segments
Electric Distribution Utility Companies Worldwide

Existing Alternatives
While there have been technical papers published on this topic, no known commercial software currently provides this capability.

Key Metrics
Customer contacts, RFP's received, contracts closed.

High-Level Concept
Use support vector regression (SVR) for enhanced load prediction, then combine with a support vector machine (SVM) classifier to classify incoming load estimate as either normative or attacked.

Channels
1. Direct to utilities
2. Via business partners: GE, Hitachi/ABB
3. Via SI's: Infosys, Accenture, Capgemini, Deloitte, Guidehouse, HCL

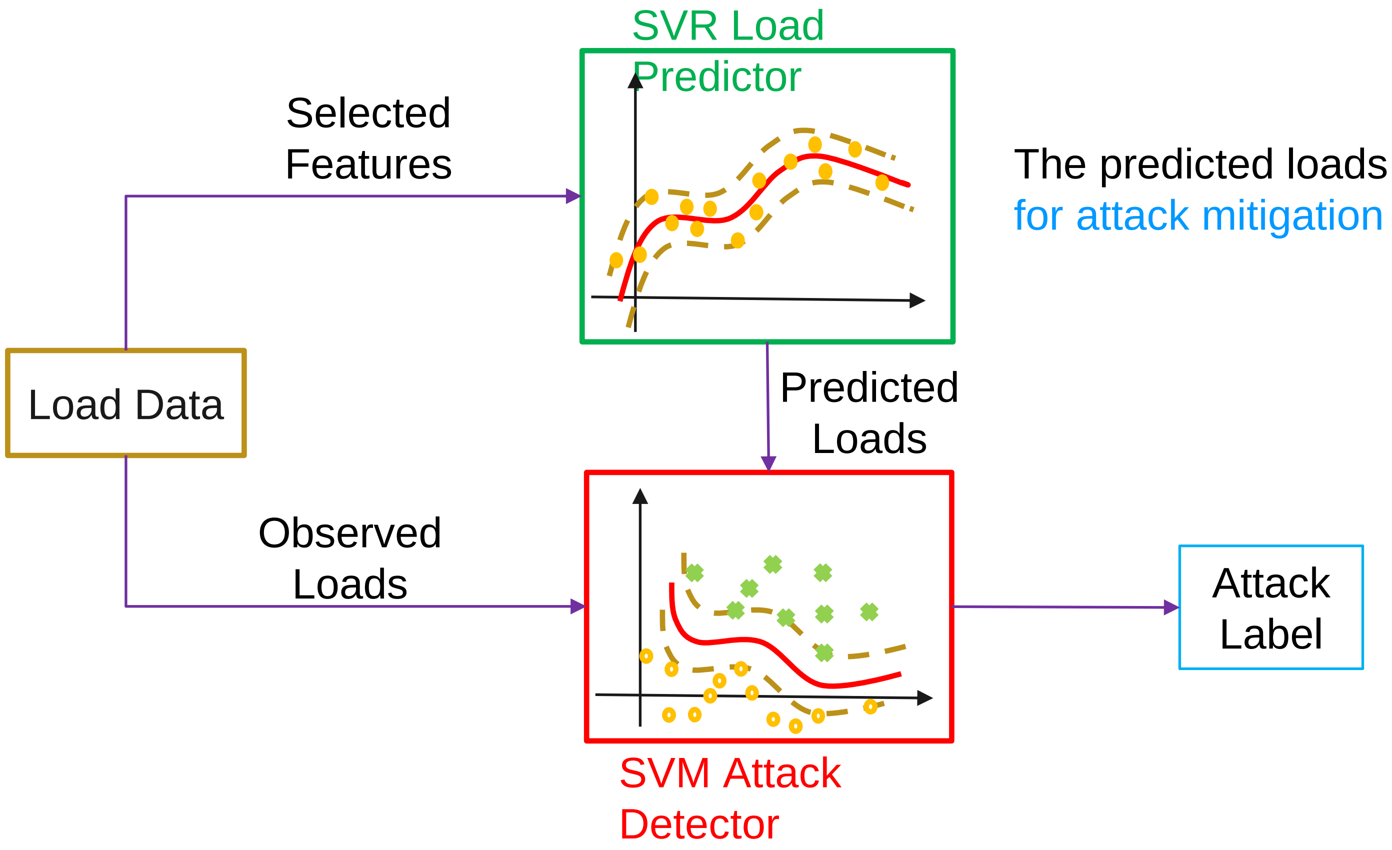
Early Adopters
Existing RI and business partner clients

Cost Structure
List your fixed and variable costs:
- Business development costs
- Software development and testing costs
- Sales engineering costs
- Project implementation costs

Revenue Streams
List your sources of revenue:
- Software licenses: one-time/perpetual or annual/subscription/SaaS
- Implementation/integration
- Ongoing support and maintenance

Commercialization: Attack Detection using SVM (Ongoing Development)

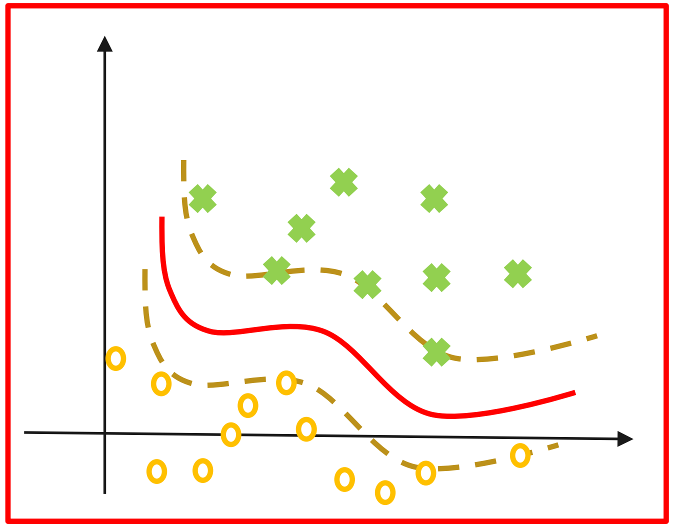
Attack Detection Framework



The predicted loads can be directly used for attack mitigation

Support Vector Machine

Find the separating hyperplane with largest margin that separates the two classes



Primal

$$\begin{aligned} \underset{\boldsymbol{w}_m, b_m, \lambda_j}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{w}_m^T \boldsymbol{w}_m + C\sum_{j=1}^{n}\lambda_j \\ \text{subject to} \quad & v_j(\boldsymbol{w}_m^T \phi(\boldsymbol{u}_j) + b_m) \geq 1 - \lambda_j \quad (\beta_j) \\ & \lambda_j \geq 0, \forall j. \end{aligned}$$

Dual

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimize}} \quad & \frac{1}{2}\boldsymbol{\beta}^T \boldsymbol{Q}\boldsymbol{\beta} - \boldsymbol{1}^T\boldsymbol{\beta} \\ \text{subject to} \quad & \boldsymbol{v}^T\boldsymbol{\beta} = 0 \\ & 0 \leq \beta_j \leq C, \forall j. \end{aligned}$$

$$Q_{ij} = v_i v_j Q(\boldsymbol{u}_i, \boldsymbol{u}_j) = v_i v_j \phi(\boldsymbol{u}_i)^T \phi(\boldsymbol{u}_j)$$

$$v_{\text{new}} = \text{sgn}\left(\sum_{j=1}^{n} v_j \beta_j^* Q(\boldsymbol{u}_j, \boldsymbol{u}_{\text{new}})\right)$$

SVM Attack Detector

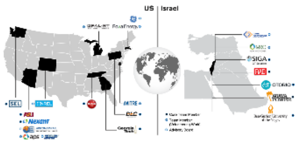
- Feature selection

Time information			Predicted loads			Observed loads			
MO	WD/WE		$\widehat{P}_{D_1}$	$\widehat{P}_{D_2}$	…	P_{D_1}	P_{D_2}	…	P_{D_N}

- Train the detector using normal data and random LR attacks to maximally explore the attack space

- Test the performance with random attack and intelligently designed attacks
  - Line overflow (LO) and cost maximization (CM) attacks
  - Map the 20 PJM zones into the 20 loads in the IEEE 30-bus system

Random Attack Generation

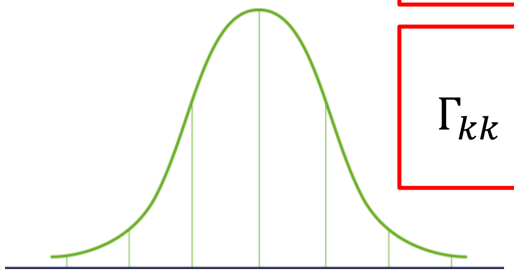


LR attacks : $P_{D,Atk} = P_D + \Delta P_D, \quad \sum_i \Delta P_{D_i} = 0$

P_D
P_{D_1}
P_{D_2}
$\cdots$
P_{D_N}

Select k loads at random $\longrightarrow$

$$\begin{bmatrix} P_{D_2} \\ P_{D_4} \\ P_{D_7} \end{bmatrix} + \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix}$$



$$\boxed{\gamma \sim N(0, \Gamma)}$$

$$\boxed{\Gamma_{kk} = \left(\frac{1}{2}\tau P_{D_{\mathrm{K}(k)}}\right)^2}$$

$$\mathbf{1}^T\gamma = 0$$

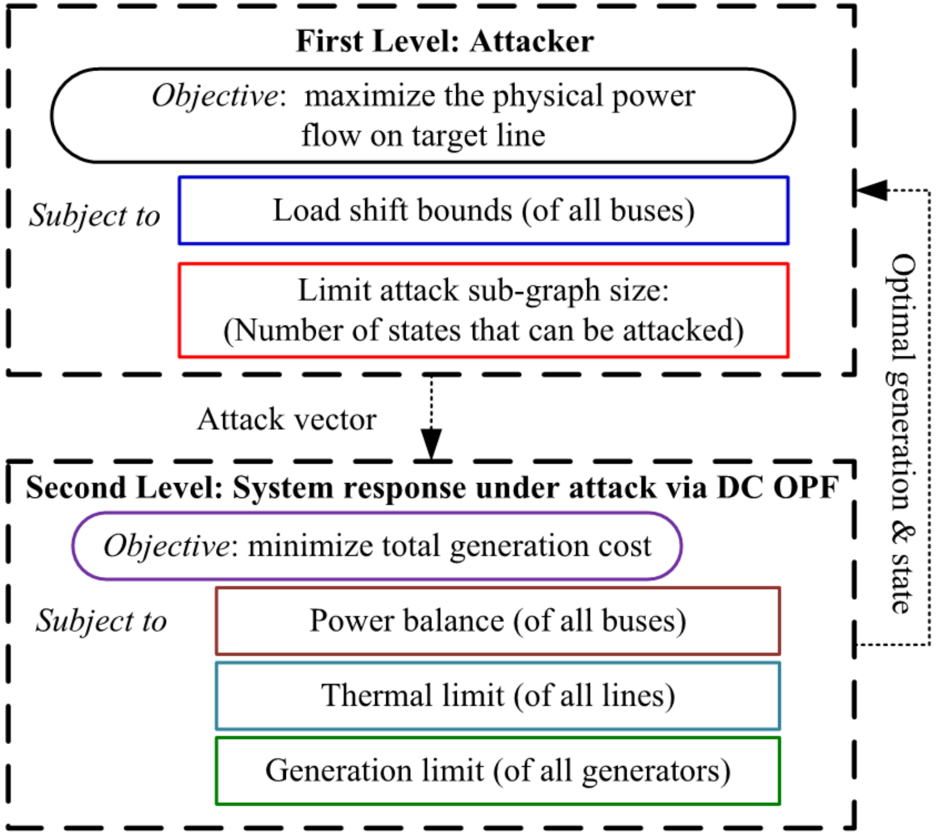
$$E[(\mathbf{1}^T\gamma)^2] = E[\mathbf{1}^T\gamma\gamma^T\mathbf{1}] = \mathbf{1}^{\mathrm{T}}\Gamma\mathbf{1} = 0$$

$$\boxed{\mathbf{1}^{\mathrm{T}}\Gamma\mathbf{1} = 0}$$

$$\boxed{\Gamma \succcurlyeq 0}$$

Intelligent LR attacks: Line Overflow (Case 1)

- Line overflow (LO) attacks
  - Bi-level optimization
  - Upper-level: manipulate measurements to generate a malicious load pattern
  - Lower level: solve DC-OPF using the manipulated data
  - This dispatch in turn causes a **line overflow**



[3] J. Liang, L. Sankar and O. Kosut, "Vulnerability Analysis and Consequences of False Data Injection Attack on Power System State Estimation," in *IEEE Transactions on Power Systems*, vol. 31, no. 5, pp. 3864-3872, Sept. 2016.

Intelligent LR attacks: Cost Maximization (Case 2)

- Cost Maximization (CM) attacks
  - Goal is to find the malicious load pattern that maximizes the cost of generation
  - Change measurements to cause such a malicious load pattern via solution to an optimization problem

Optimization Problem –

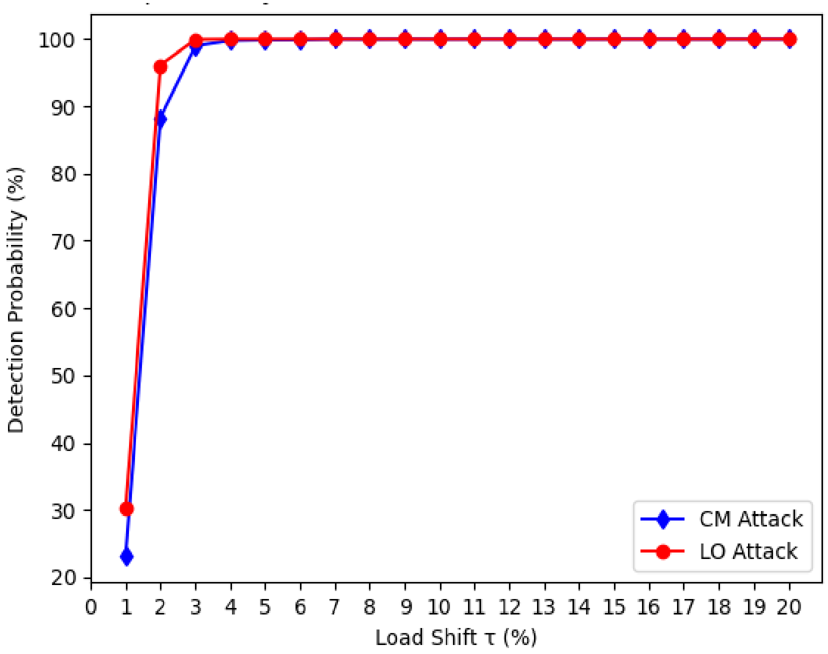
Attack vector *c* is obtained by solving

$$
\begin{aligned}
\underset{\boldsymbol{c}}{\text{maximize}} \quad & \boldsymbol{a}^T \boldsymbol{G}^* \\
\text{subject to} \quad & -\tau \boldsymbol{P} \leq \boldsymbol{B c} \leq \tau \boldsymbol{P} \\
& \{\boldsymbol{G}^*, \boldsymbol{P}_{\boldsymbol{L}}^*\} = \arg \left\{ \min_{\boldsymbol{G}, \boldsymbol{P}_{\boldsymbol{L}}} \ \boldsymbol{a}^T \boldsymbol{G} \right\} \\
& \text{subject to} \quad \sum \boldsymbol{G} = \sum \boldsymbol{P} \\
& \qquad \boldsymbol{P}_{\boldsymbol{L}} = \boldsymbol{R}(\boldsymbol{G} - \boldsymbol{P} + \boldsymbol{B c}) \\
& \qquad -\boldsymbol{P}_{\boldsymbol{L}}^{\max} \leq \boldsymbol{P}_{\boldsymbol{L}} \leq \boldsymbol{P}_{\boldsymbol{L}}^{\max} \\
& \qquad \boldsymbol{G}^{\min} \leq \boldsymbol{G} \leq \boldsymbol{G}^{\max}
\end{aligned}
$$

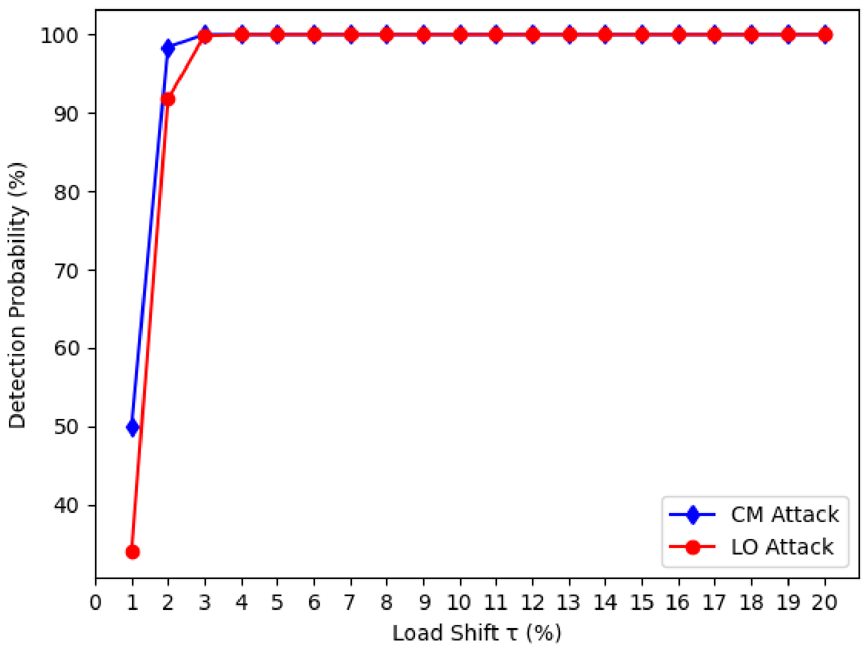
[4] Z. Chu, L. Sankar and O. Kosut, “Detecting Load Redistribution Attacks via Support Vector Models," in *IET Smart Grid*, vol. 3, no. 5, pp. 551-560, Oct 2020.

LR Attack Detection: Evaluation (On-going Efforts)

- Evaluation on PJM dataset: illustrations for τ_{min} = 3% and C = 2000 (τ_{min} is the smallest load shift used in training)
- CM attacks with consequences are those that increase the operating cost by more than 1%
- LO attacks with consequences are those that result in physical overflows
- **Next Step: Evaluation required on different datasets: e.g., CAISO, TX-2000 bus system**



All Attacks

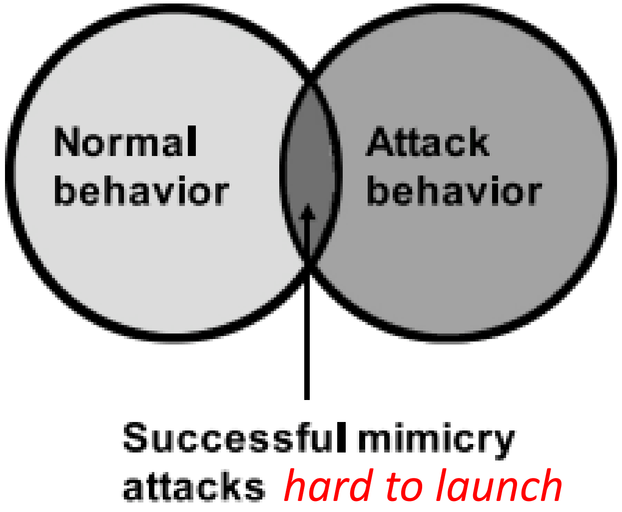


Attacks with consequences

Event-Mimicking Attacks on PMU Data: Design and Mitigation

Event-mimicking Attacks and Countermeasures

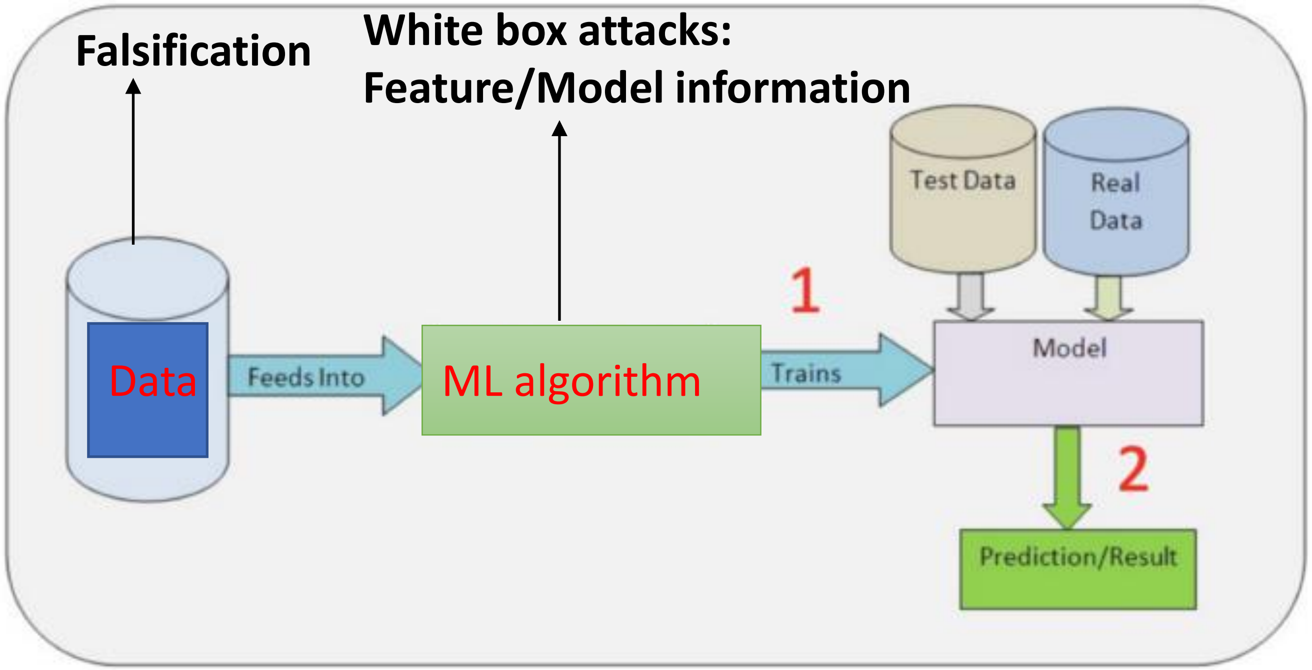
- Modern grid with renewables is more stochastic in operations and requires real-time monitoring to detect/identify real events (oscillations/outages) and attacks.
- ML-based detectors can be easily evaded by attacks that mimic events, ultimately, causing significant damage on grid operations.



mimicry attack: a careful cyberattack on data that throws off ML detector

Source: https://towardsdatascience.com/evasion-attacks-on-machine-learning-or-adversarial-examples-12f2283e06a1

Where Can Attackers Target OT Systems?



PMU data can be falsified but for mimicking event attacks

- how to tamper data?

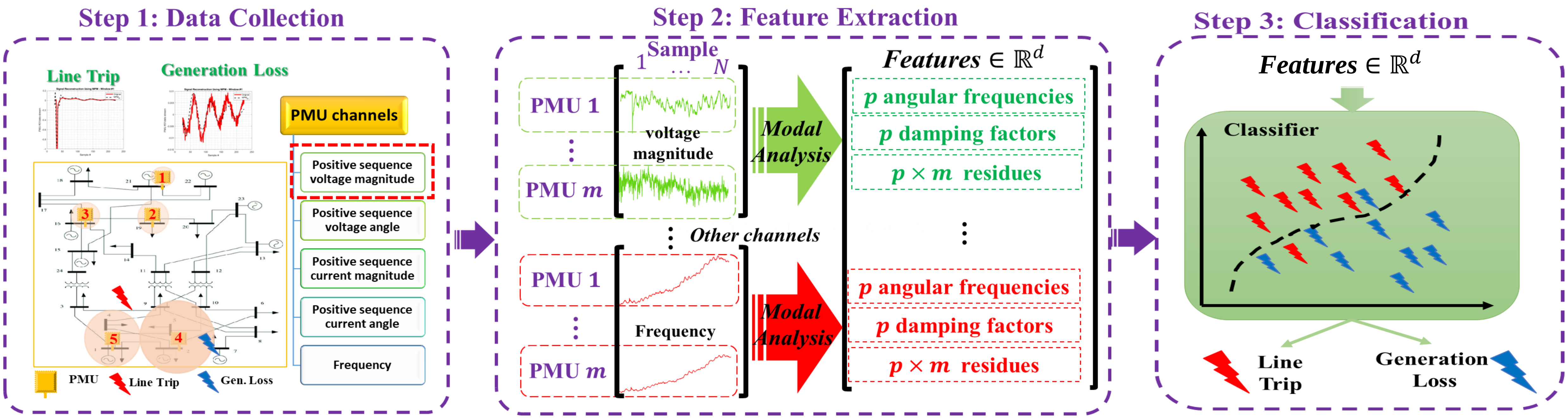
- how many PMUs to tamper?

- how long to tamper?

extract and exploit
signal physics (modes)

Source: https://towardsdatascience.com/will-my-machine-learning-be-attacked-6295707625d8

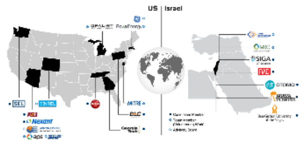
Event ID: Learn Event Signatures from Measurements



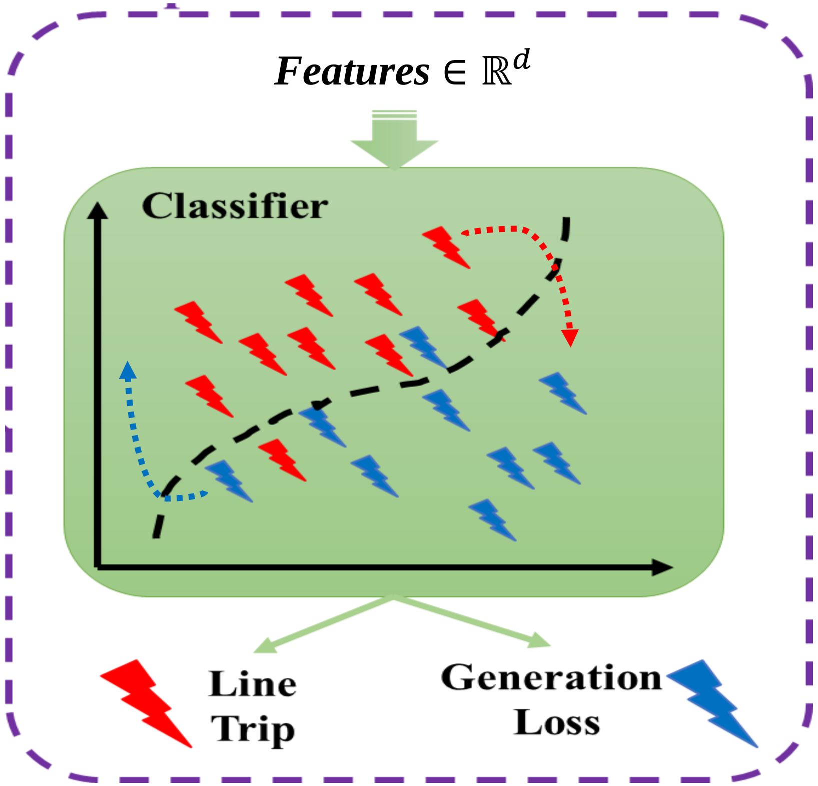
- ✓ Characterizing events based on a set of physically interpretable features
- ✓ Finding the most informative sparse set of features
- ✓ Learning a set of robust classification models to identify the events

[5] N. Taghipourbazargani, G. Dasarathy, L. Sankar and O. Kosut, "A Machine Learning Framework for Event Identification via Modal Analysis of PMU Data," in IEEE Transactions on Power Systems, 2022.

Attack Design: Threat Model



- Start with White Box Attack Model: Attacker has full information of the event classifier (LR)
- Untampered Features:
  - Angular Frequency
  - Damping
  - Residual Amplitude
  - Residual Angle
  - Channels: Voltage magnitude, voltage angle, frequency
- Tamper features just enough for the event to be misclassified
  - Move feature sample across decision boundary

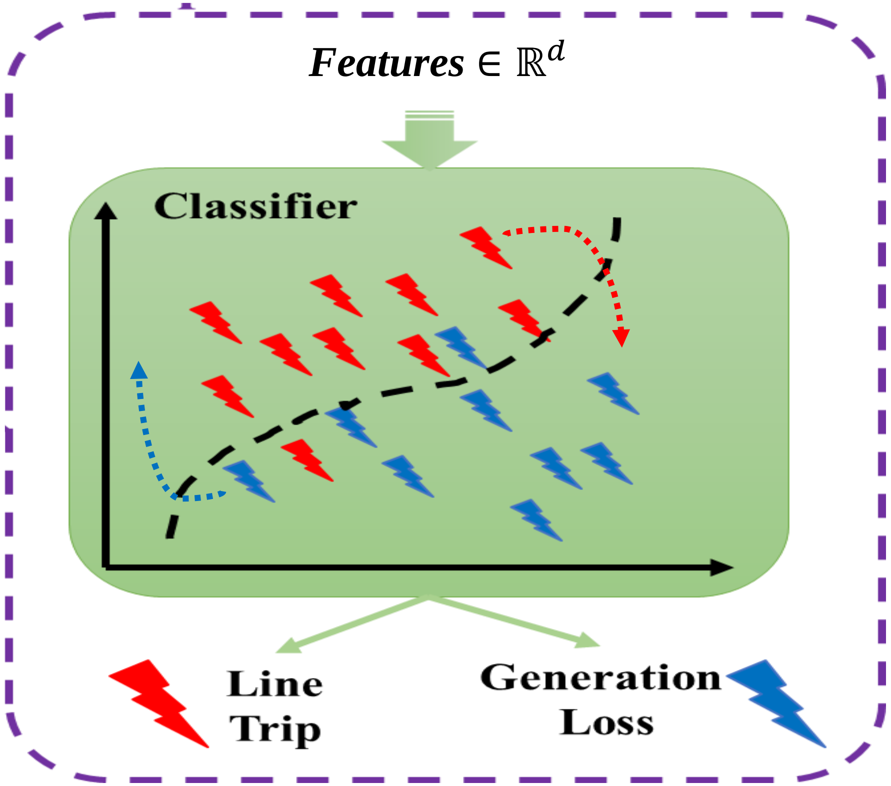


Event Mimicking Attack Algorithm

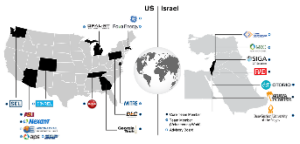
Inputs: LR classifier, attack parameters, PMU data

1. Tamper features until the event is misclassified by employing the knowledge of LR parameters
2. Reconstruct time signals of the tampered data
3. Replace the time domain signals for only the PMUs under the attacker's control
4. Extract features of the new signals set
5. Classify using LR model
6. Repeat 1 through 5 until misclassification

Output: tampered PMU measurements



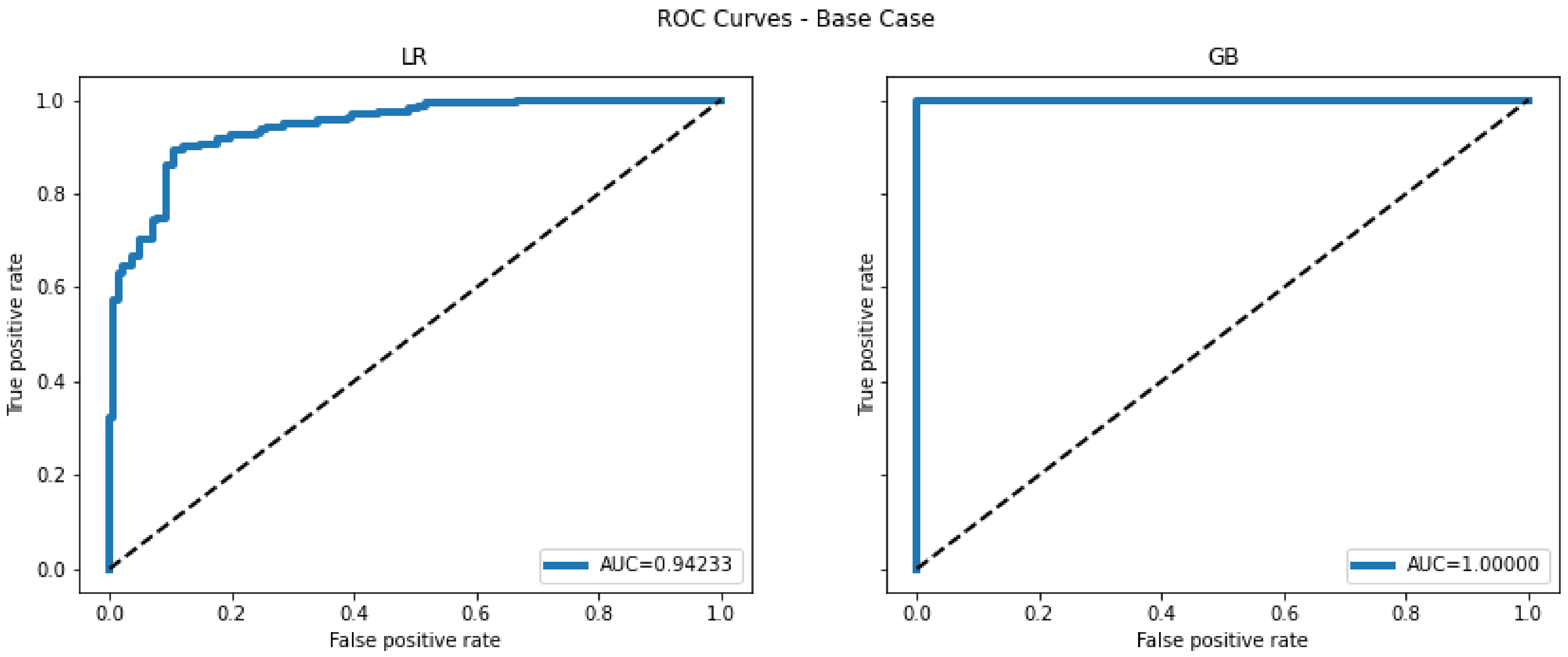
Setup and Assumptions for Illustrations



- Network and data: synthetic PMU data generated using PSS\E for South Carolina 500-bus system
  - 750 generation loss and 750 load trip events
  - Voltage magnitude, voltage angle, and frequency measurements are collected from 95 PMUs across the system
- Classifiers: Logistic regression (LR) and gradient boosting (GB) algorithms
  - Training data: 591 generation loss and 609 load trip events
  - Test data: 159 generation loss and 141 load trip events
  - Modal analysis is used for feature extraction

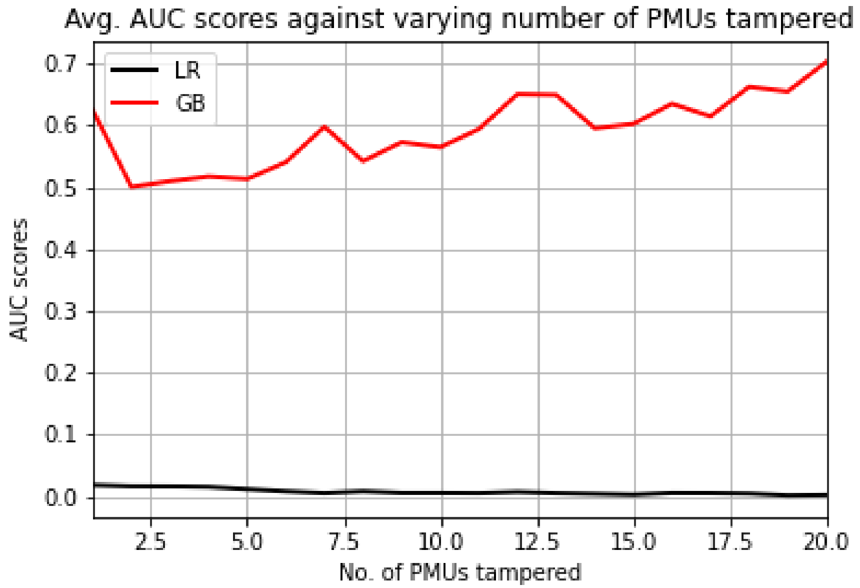
Classification of Untampered Events

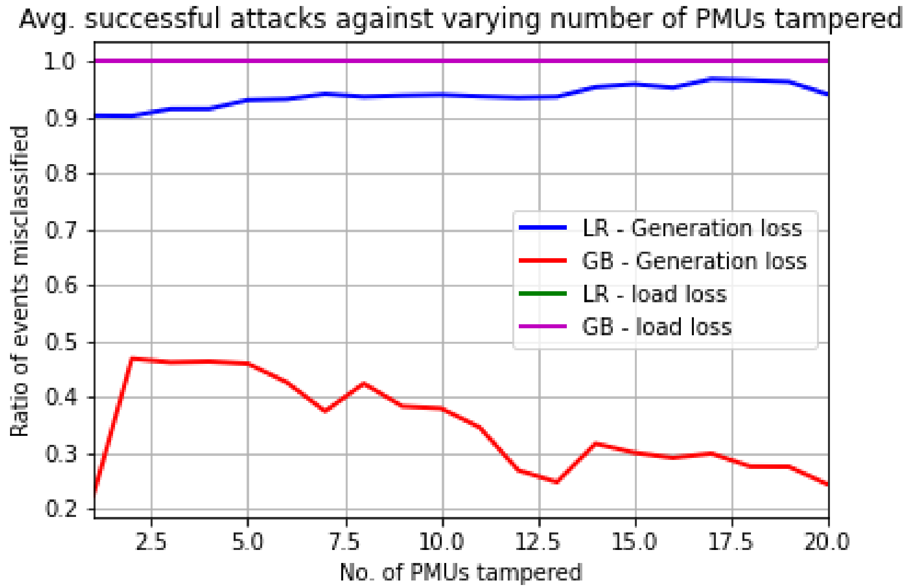
- ➢ Event classifier is applied to 300 test data (159 generation loss and 141 load trip events)
- ➢ LR and GB classifiers are used to classify untampered test data to establish a base case
  - ➢ Both models are trained on the same dataset
- ➢ Both models classify the events with very high accuracy



Attack Illustration

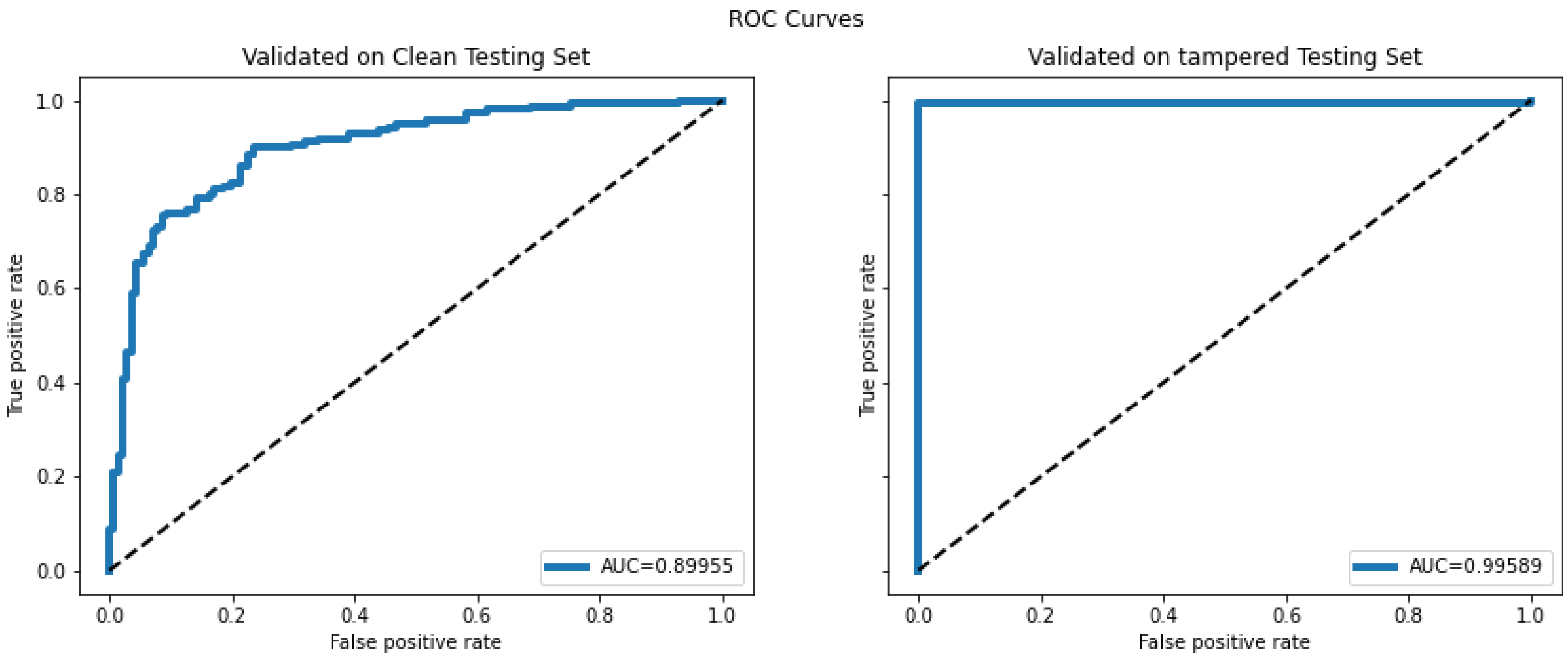
- Attack Assumptions:
  - Attacker has full knowledge of LR classifier model
  - Attacker has access to a subset of system PMUs (no more than 20)
  - Tampers 1200 events (training set) comprised of 591 generation loss and 609 load loss events
- Efficacy of tampered data also evaluated on GB classifier (trained on clean data)
- Results: overall successful attack with LR having a higher success rate as expected
  - Generation loss detection using GB has higher robustness against the attack
  - Load loss attack has a 100% success rate against LR and GB classifiers



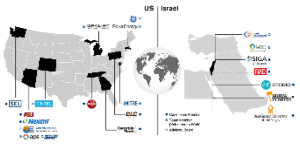


Mitigation: Adversarial Approach

- Attack algorithm generates adversarial examples that are likely to be misclassified
- The generated adversarial examples are used in combination with clean data to train new classifier
  - Robust classifier should be able to identify tampered and untampered data with their true label with high accuracy
  - How do we know the attack can't be applied again with this classifier?!



Robust Classifier Training Algorithm



Inputs: LR classifier, attack parameters, PMU data

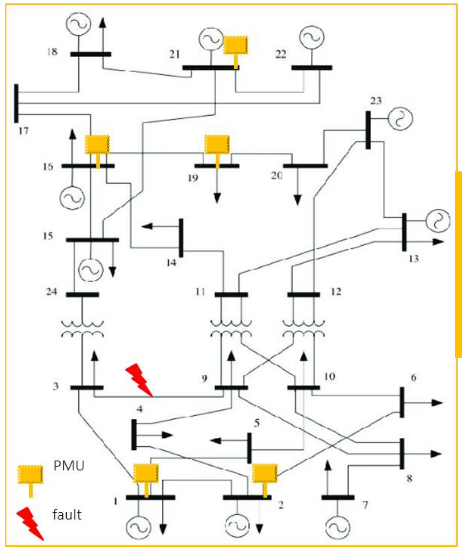
1. Apply the attack employing the knowledge of LR classifier
2. Train new LR classifier using combined adversarial examples and clean data
3. Update the LR classifier
4. Validate the attack by applying it on unseen clean data using the updated classifier
5. Repeat 1 through 4 until the success rate of the attack on the unseen data diminishes

Output: Robust Classifier

➢ *Performance evaluation in progress*: preliminary results are encouraging

Broader Utility: API for generation of eventful PMU data

Generation of Synthetic Eventful PMU data



Input:
.raw file
.dyr file

Python API

PSSE

Initialization: Get the list of loads, generators, lines, and buses

For different loading conditions:

For any component:

1. Apply the **new loading** condition
2. Run the **power flow**
3. Initialize dynamic simulation
4. Flat run for **1 second**
5. Apply disturbance on the component at **t=1 second**
6. If the component is a **bus**:
   clear the disturbance after **5 cycles**
7. Run the dynamic simulation for additional **10 second**
8. Record the **Vm, Va, F** measurements

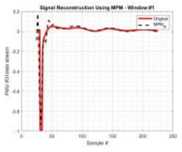
Network:
South-Carolina 500 bus system

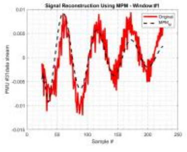
No. of Generated events:
Load loss: 500
Generation Loss: 500
Line Trip: 500
Bus Fault: 327

Output

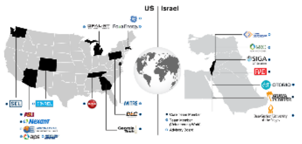
Vm, Va, F channels .out, .xlsx., and .mat files and

$$\mathbf{D} = \{\boldsymbol{X}_D, \boldsymbol{Y}_D\}$$


Line Trip

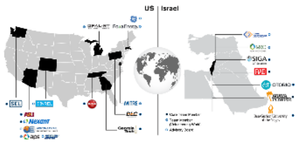

Gen Loss

Summary



- Commercial-grade software for bus-level load prediction
  - Evaluated extensively on PJM, CAISO, and TX-2000 network data
  - Remarkable prediction accuracy
  - CI-CD pipeline demonstrated on AWS
  - Code hand-off to Resource Innovations, Inc.
- Commercial-grade software for generation of intelligent attacks
  - On-going development for line overflow and cost maximization attacks
  - Attack detection using support vector machines
  - Evaluated on PJM dataset: needs evaluation on additional datasets
- Event-mimicking attack generation via physics-informed ML
- Robust classifiers designed via adversarial ML (on-going)
  - Promising initial results for logistic regression and gradient boosting
  - Extensions to different classifiers including GANs.
- Python API for creation of synthetic eventful PMU data

Select References



- Z. Chu, L. Sankar and O. Kosut, “Detecting load redistribution attacks via support vector models,” in *IET Smart Grid*, 3 (5), pp. 551-560, Oct 2020.
- J. Liang, L. Sankar and O. Kosut, “Vulnerability analysis and consequences of false data injection attack on power system state estimation,” in *IEEE Transactions on Power Systems*, 31 (5), pp. 3864-3872, Sept. 2016.
- Z. Chu et al., “A verifiable framework for cyber-physical attacks and countermeasures in a resilient electric power grid,” *arXiv preprint arXiv:2104.13908*, 2021
- N. Taghipourbazargani, G. Dasarathy, L. Sankar and O. Kosut, “A machine learning framework for event identification via modal analysis of PMU data,” in *IEEE Transactions on Power Systems,* 38 (5), pp. 4165-4176, Sept. 2023
- N. Taghipourbazargani, L. Sankar and O. Kosut, “A semi-supervised approach for power system event identification,” *arXiv preprint arXiv:2309.10095*, Sept. 2023
- O. Bahwal, L. Sankar, and O. Kosut, “An adversarial approach for evaluating the robustness of event identification models”, *in preparation,* Oct. 2023