energies

MDPI

Article

Improving Detection of False Data Injection Attacks Using Machine Learning with Feature Selection and Oversampling

Ajit Kumar [1], Neetesh Saxena [2], Souhwan Jung [3] and Bong Jun Choi [1,*]

[1] School of Computer Science and Engineering, Soongsil University, Seoul 06978, Korea; ajitkumar.pu@gmail.com
[2] School of Computer Science and Informatics, Cardiff University, Cardiff CF10 3AT, UK; nsaxena@ieee.org
[3] School of Electronic Engineering, Soongsil University, Seoul 06978, Korea; souhwanj@ssu.ac.kr
* Correspondence: davidchoi@soongsil.ac.kr

Abstract: Critical infrastructures have recently been integrated with digital controls to support intelligent decision making. Although this integration provides various benefits and improvements, it also exposes the system to new cyberattacks. In particular, the injection of false data and commands into communication is one of the most common and fatal cyberattacks in critical infrastructures. Hence, in this paper, we investigate the effectiveness of machine-learning algorithms in detecting False Data Injection Attacks (FDIAs). In particular, we focus on two of the most widely used critical infrastructures, namely power systems and water treatment plants. This study focuses on tackling two key technical issues: (1) finding the set of best features under a different combination of techniques and (2) resolving the class imbalance problem using oversampling methods. We evaluate the performance of each algorithm in terms of time complexity and detection accuracy to meet the time-critical requirements of critical infrastructures. Moreover, we address the inherent skewed distribution problem and the data imbalance problem commonly found in many critical infrastructure datasets. Our results show that the considered minority oversampling techniques can improve the Area Under Curve (AUC) of GradientBoosting, AdaBoost, and kNN by 10–12%.

Keywords: Data Injection Attack; machine learning; critical infrastructure; smart grid; water treatment plant; power system

Citation: Kumar, A.; Saxena, N.; Jung, S.; Choi, B.J. Improving Detection of False Data Injection Attacks Using Machine Learning with Feature Selection and Oversampling. *Energies* **2022**, *15*, 212. https://doi.org/10.3390/en15010212

Communicated by: Silvio Simani and Zbigniew Leonowicz

Received: 9 November 2021
Accepted: 27 December 2021
Published: 29 December 2021

1. Introduction

Today, the umbrella term 'Industry 4.0' represents the integration of digital control, Information and Communications Technology (ICT), and intelligent decision-making into critical infrastructures. This upgrade is possible due to the amalgamation of information and industrial technologies into standard components and processes [1,2]. This shift, from the traditional system to Industry 4.0, has helped improve the overall performance and productivity of critical infrastructures that have become the fundamental building blocks of modern society. For instance, electricity distribution and usage can be optimized in smart grids. In water systems, in-time data about usage and plant treatment capacity can reduce water wastage. Along with various benefits and improvements, the addition of new components into critical infrastructures presents new vulnerabilities [3–5]. This critical infrastructure is especially sensitive to cyberattacks. Even a low-scale attack that causes a few critical infrastructure components to malfunction can impact the whole system. For example, even a short disruption in the power grid can halt the functioning of many industries and infrastructures, from food processing plants to hospitals. The attack on the Ukraine grid infrastructure and a recent ransomware attack on a colonial pipeline are some of the many alarming examples that call for improvements to be made to the defense techniques which protect critical infrastructures [6,7].

The injection of data or commands at the source or during communication is collectively called False Data Injection Attack (FDIA). Data injection refers to the manipulation

of the value generated from sensors, actuators, and other devices, while the command 'injection' refers to changing server-issued instructions. The FDIA is one of the most common attacks and can be launched on any critical infrastructure by penetrating the communication sessions between different devices. FDIAs can damage physical components, induce huge economic losses, and even create life-threatening scenarios [8,9]. Therefore, it is essential to prevent and detect FDIAs in any critical infrastructure. However, most of the existing solutions are only theoretical or adopt techniques from the cyber environment, such as Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), which are used to protect conventional computer networks. These existing approaches are missing the specific security requirements and properties of critical infrastructure, such as a higher rate of event, the need for real-time interaction and detection, the need for proactive defense, and complex physical and cyber interfaces. Therefore, this study addresses these limitations by applying machine learning techniques to detect injection attacks.

This study work addresses two key issues to fulfill the requirements of proactive detection, low response time, and detection of minority classes of critical infrastructures: (1) finding the set of best features under different selection techniques and (2) resolving the class imbalance problem using oversampling. Our work uses a smart power system [10] and a water treatment plant [11] as case studies. We found the best features using different filter and wrapper selection methods (top- and bottom-ranked features). We also tested the performance of various machine-learning algorithms with different sets (with varying sizes) of the best features. Further, to improve the detection performance, especially for the minority class, i.e., attack class, we applied different oversampling methods to increase the sample for the minority class. We also solved common issues in the datasets, such as missing and corrupted values.

The contribution of this study is summarized below.

1. We provide a comprehensive analysis of machine-learning algorithms for FDIA detection using the two representative datasets, namely power system and water treatment datasets.
2. We determine the subset of features which can be used to achieve the best performance using different filter and wrapper approaches.
3. We mitigate performance bias in imbalanced datasets using four different oversampling methods.

The remainder of the paper is organized as follows. The related works are presented in Section 2. A detailed explanation of both critical infrastructures (power system and water treatment plant) from which data about events have been recorded for datasets creation is presented in Section 3. Section 4 presents the three feature selection approaches and provides a ranking of the features. The class imbalance issue and oversampling methods are discussed in Section 5. The results of training and testing and the outcome of oversampling are presented in Section 6. Lastly, Section 7 provides the conclusion and future research directions.

2. Related Work

In the section, we present the details of FDIA and other attacks targeted to the Cyber-Physical System (CPS) and provide a summary of existing FDIA detection methods based on machine learning. Further, the limitations and research gaps in the existing literature, which motivated the current study, are discussed.

With the fast transition of the traditional grid to the smart grid, the effective implementation of FDIA is critical to the success of the smart grid. There have been many FDIA attacks demonstrated in the literature. In the last five years (2015–2020), some surveys provided discussion and a comprehensive summary of challenges and countermeasures regarding FDIA. The role and importance of Artificial Intelligence (AI) and big data technologies for FDIA detection were also highlighted [9,12–14]. The financial impact of FDIAs was demonstrated in [8]. The authors assumed an insider attack and simulated an injection attack by changing the value of the memory location of the Programmable Logic Controller

(PLC). Experimental results showed that the injection attack could directly impact the electric usage billing system, generating a manipulated final bill.

Traditionally, state estimation and time-series analysis are the main methods used for FDIA detection. Recently, many AI-based approaches have been adapted to improve detection performance [15]. Class labeling and class-balanced datasets are two critical challenges for developing a machine learning-based FDIA detection system for the smart grid because of the small sample size for FDIA class and complex labeling. Maglaras et al. [16] used a One-Class Support Vector Machine (OCSVM) with *normal events* to resolve these two challenges for a Supervisory Control and Data Acquisition (SCADA)-based critical infrastructure. Due to the challenges involved in dataset preparation, FDIA detection with minimum training and prediction time is required to handle the high rate of data generation in the smart grid. Reducing the vectors of features using Principal Component Analysis (PCA) and speeding up the training time using Distributed SVM are used to achieve low computation requirements of the smart grid [17]. Further, FDIAs in the smart grid are grouped into 'direct' and 'stealth', where 'stealth FDIAs' are more challenging to detect than 'direct FDIAs'. Yan et al. [18] used supervised machine learning to build FDIA detection systems by formulating the detection as binary classification (direct and stealth). The authors also tested detection performance for balanced vs. imbalanced class distribution using the IEEE 30-bus simulation dataset. More recently, the Artificial Neural Network (ANN) has been applied for FDIA detection. Khanna et al. [4] used ANN and Extreme Learning Machine (ELM) to detect Data Injection Attacks on the consumer side of the smart grid and classified electric meters as either benign malicious. The NYISO load data was mapped to an IEEE 14-bus system for performing simulation, experiments, and validation of results. Data generation sources in the smart grid can be grouped into cyber or physical space. Wang et al. [19] have collected simulated and real-world measurements of synchronized PMUs and applied the Margin Setting Algorithm (MSA) for detection. The ensemble of Machine Learning (ML) algorithms was shown to improve the detection performance in [15]. In this direction, the performance of ensemble learning for multi-class classification was tested for a total of 37 classes, including FDIA in [20]. The experiments were executed using a dataset containing measurements of four Phasor Measurement Units (PMUs) and network communication data to and from the firewall and IDS of the experimental power system [10]. FDIA detection is also formulated as a three-class problem, rather than a binary classification, in the literature. Panthi et al. [21] used machine-learning algorithms and the publicly available power dataset [10] to build a classifier to group events into natural, no-event, or attack classes.

A fingerprinting-based detection of stealthy cyber-attacks in water treatment plants was proposed in [22]. An IDS using a semi-supervised system for attack localization and deep neural network learning for anomaly detection was proposed in [23]. More recently, a two-level attack-detection framework using a decision tree for detection and deep learning for attribution was proposed in [24].

Based on the summary of existing literature, we observe that machine learning-based FDIA detection approaches can improve detection performance and address some of the key requirements, such as real-time large-scale data generation in the smart grid. Such improvement will promote machine learning models for FDIA detection in smart grids and other critical infrastructure. Our literature also indicates that most existing research works have used the power system dataset and formulated FDIA classification as a multi-class problem. So, to explore a novel dimension, we consider FDIA classification as a binary problem and made pre-processing necessary to the dataset to experiment under various environments. The power system dataset [10] used also contains a binary version that was formulated the classification as 'Attack' and 'Normal'. In contrast, the classification problem was formulated as 'FDIA' and 'non-FDIA' in this study. Feature selection is useful and required to reduce time complexity but is seldom used with critical infrastructure datasets. Therefore, we experimented with feature selection methods and machine-learning algorithms. We aim to find the best performance of classifiers given the selected features.

As shown in Table 1, the data imbalance issue is rarely addressed. Therefore, we also performed minority class oversampling to balance the class distribution beyond identifying the imbalanced dataset's effect on detection accuracy.

Table 1. Summary of related work on FDIA detection using machine learning

Ref.	Method	Dataset	Samples Ratio (Normal, Attack)	Feature Selection
[18]	Supervised Learning (SVM, kNN, and ENN)	Simulations IEEE 30-bus system	0.1	No
[4]	ANN and ELM	NYISO load data IEEE 14-bus system	NA	NA
[19]	Data Centric (Big Data and MSA)	Simulated (6 bus power system) and real-world (Texas synchrophasor network)	100 K, 0.334, 0.196, 0.086	No
[20]	Voting on ML-classifier, dataset divided per PMUs	4 PMUs events and firewall log [10]	Balance	Yes
[16]	OCSVM	Network traces	1570, NA	NA
[15]	Ensemble Learning	Measurement data and power system audit logs	Balance	Yes
[17]	Distributed SVM and PCA	IEEE standard test systems	NA	Yes
[21]	Machine Learning (One R, J-Ripper, NB, RF)	Power system [10]	NA	No
[22]	Fingerprinting and OCSVM	Water treatment (SWaT)	NA	NA
[23]	AutoEncoder	SWaT	NA	Yes
[24]	DT and Deep learning	SWaT and gas pipeline	214 K, 21.86%	Yes

3. Critical Infrastructure Experimental Framework

The critical infrastructure is any physical infrastructure, such as a power system, healthcare [25], or gas pipeline, that is essential to support our daily lives [26]. Hence, disturbance to these systems has a huge impact on society, the economy, and the environment [27]. Recently, the development in computer and network technologies has enabled the fast adoption of ICT in such critical infrastructures. For example, the traditional power grid is now controlled, operated, and monitored using ICT, migrating the century-old power grid into a smart grid. Such integration is also evident in almost all critical infrastructures. However, cyber involvement in the physical system makes it vulnerable to various cyber attacks such as FDIA, unauthorized access, etc. In our study, we considered FDIA detection in power systems and water treatment plants using two very popular open datasets.

3.1. The Experimental Framework for Power System Data

Power generation, storage, and distribution tasks are continuously performed in the smart grid. Moreover, the complexity and large scale make it infeasible to experiment with the real infrastructure. Moreover, data access in the smart grid environment is highly restricted due to privacy and security concerns. Research is often performed on a reduced scale or using simulated datasets, such as IEEE 14/30 buses, to accommodate the abovementioned limitations. In 2015, the Mississippi State University and Oak Ridge National Labora-

tory dataset (https://www.sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets (accessed on 8 November 2021)) produced a scaled-down version of the power system and recorded a dataset with various simulated attacks in addition to normal events [10]. The experimental power system has two power generators (*G*1 and *G*2), four Intelligent Electronic Devices (IEDs) (R1 to R4), and four breakers (BR1 to BR4). Two lines were created in the power system using the pairs of breakers (BR1 and BR2; BR3 and BR4). The four IEDs, R1 to R4, were configured to open or close the four breakers, BR1 to BR4, respectively. A server controls the physical part of the power framework in the control room, and these cyber and physical parts are connected using a switch and a Power Distribution Center (PDC).

3.1.1. Dataset Pre-Processing

The power system environment discussed in Section 3.1 helped to create a suitable dataset for conducting machine learning-based detection experiments [10]. The complete dataset is available and being distributed as 15 sets. The dataset comprises 37 power system events that can be grouped into three main scenarios: (1) *Natural*, (2) *No Events*, and (3) *Attack Events (data injection and command injection)* containing 8, 1, and 28 events, respectively. Regrouping and resampling are performed using these three types of events, and three datasets are created for binary, three-class, and multi-class classifications. For the multi-class classification, each event type is considered a class therefore, it has 37 classes. The binary and three-class datasets are distributed in CSV file format. However, the multi-class dataset is available only as an ARFF (an Attribute-Relation File Format (ARFF) is a file format created to be used by the Waikato Environment for Knowledge Analysis (WEKA) tool. It is a Graphical User Interface (GUI) tool for performing machine learning tasks such as per-processing, training, exporting models, and creating an ML pipeline.).

Building a multi-class machine learning classifier is complex and resource-consuming. It also creates a dataset's class imbalance problem. Considering this, we reformulated the FDIA detection as a *binary classification* with 'FDIA' and 'Non-FDIA' classes. However, the existing dataset was unsuitable for this study, so we grouped samples based on the type of events. Before resampling, we converted the multi-class ARFF to CSV format to simplify further pre-processing, training, and testing. Filtering and merging were performed on all 15 sets to group all scenarios into two predefined classes; Normal/non-FDIA classes were 1–6, 13, 14, and 41, while "FDIA" classes were 7–12. Further, the Normal/non-FDIA sample was labeled as 0, and the FDIA sample was labeled as 1. The total number of samples that the final pre-processed and resampled dataset contained was 32,296. Figure 1 shows the class distribution. There are 22,714 samples in the 'Normal/non-FDIA' class and 9582 in the 'FDIA' class. As shown in Figure 1 and listed in Table 2, it is clear that the power system dataset is an imbalanced dataset where 'Normal/non-FDIA' is the majority class. All four features related to impedance for IEDs relays such as 'R1-PA:Z' had infinite value, and so, as a pre-processing step, they were replaced with 0.

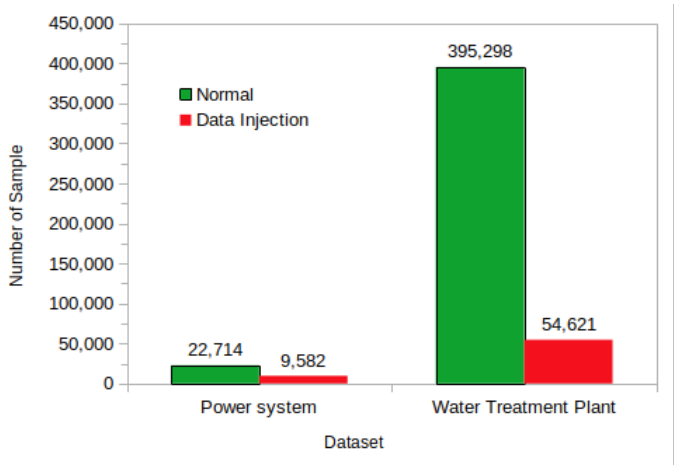


Figure 1. Number of samples per class in the dataset.

Table 2. Distribution of the classes (normal and FDIA) in the dataset.

Class Type	Number of Samples	
	Power System	Water Treatment
Normal	22,714	395,298
FDIA	9582	54,621
Total	32,296	449,919

3.1.2. Description of Features

There is a total of 128 features in the dataset, consisting of PMUs measurement and software logs. A total of 29 measurements were recorded for each PMU, so a total of 116 features were collected from 4 PMUs. The logs were recorded from three sources: snort, relay, and control panel. Each had 4 values, so logs contributed a total of 12 features. Each feature was given a name based on the combination of a source of data and type of value. For example, PMUs features begin with *R#−Signal Reference* and log features start with the source of logs such as snort, control_panel, and relay. The # for PMU features was a number between 1–4, indicating the PMUs number, while postfix *Signal Reference* was the type of measurement. These measurements fall into two groups: first, phase angle and magnitude for voltage and current, and, second, frequency, frequency delta, appearance impedance, and appearance impedance angle for relays. Details of these features are presented and explained in the original dataset description document (http://www.ece.uah.edu/~thm0009/icsdatasets/PowerSystem_Dataset_README.pdf (accessed on 8 November 2021)).

It is important to understand the impact of *false data injection* on individual features. We used a data distribution approach and created an overlapped histogram for individual features. For example, a histogram for *R1-PA1:VH* features is plotted for all normal and FDI samples. We can observe from Figure 2 that the count for the specific value range is higher in the FDI sample, which indicates data injection.

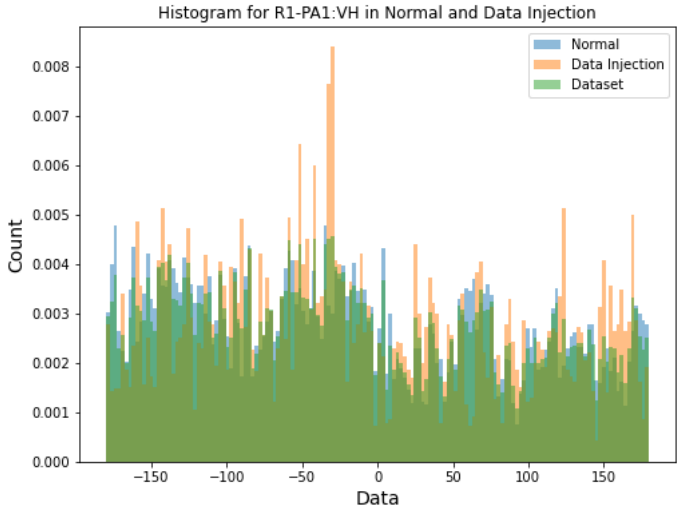


Figure 2. Histogram comparison of normal, injection, and all dataset events for *R1-PA1:VH*.

3.2. Water Treatment Plant

For the same reason as for a power system, a testbed, i.e., a scaled-down version of a real water treatment plant or pipeline, is normally created to experiment and collect data. In this study, we used a similar dataset, the Secure Water Treatment (SWaT) testbed [11] (a fully operational scaled-down water treatment plant), for FDIA detection. The configuration and framework of the experimental water treatment plant are depicted in Figure 3. It has six processing stages for water treatment labeled P1–P6. In total, the testbed has 24 sensors, 27 actuators, and 6 PLCs (one for each stage). The count for each type of sensor and actuator is listed in Table 3.

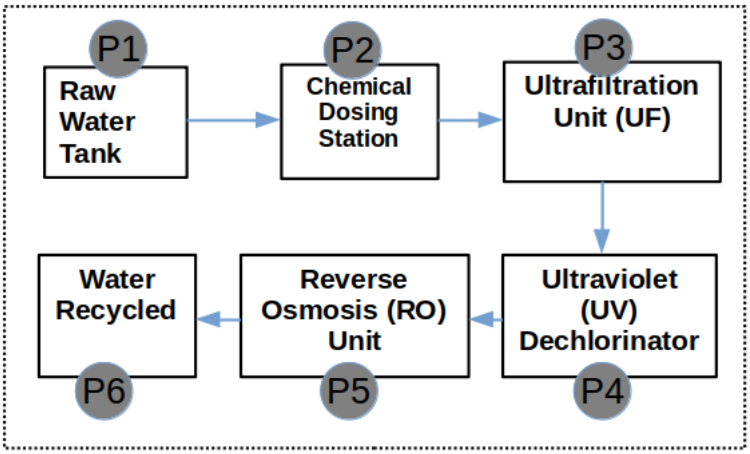


Figure 3. The water treatment plant in [11], where P1–P6 denotes total six stages of processing in the plant.

3.2.1. Dataset Pre-Processing

In the SWaT testbed, there are PLCs, Human Machine Interfaces (HMIs), SCADA, and a Historian in a layered communication network. Data from field devices are available to SCADA via PLCs and transferred to the Historian for analysis. The dataset contains events from *physical* and *network activities* against the 36 predefined attacks. The complete dataset was collected during a period of 11 days, during which the plant was running continuously for 24 h each day.

Table 3. Description of different sensors and actuators used for data generation. Features are named as combination of type (i.e., MV, P, FIT etc.) and suffix (process number and device number). For example, FIT-101 can be read as being a first flow meter sensor of process stage 1.

Field Device	Type	Description	Total (51)
Actuators (27)	MV	Motorized Valve	6
	P	Pump	19
	LIT	Level Transmitter	1
	UV	Dechlorinator	1
Sensors (24)	FIT	Flow Meter	9
	LIT	Level Transmitter	2
	AIT	Analyzer	9
	DPIT	Differential Pressure Indicating transmitter	1
	PIT	Pressure meter	3

3.2.2. Feature Description

The collected dataset contains 449,919 physical events and 51 features mainly generated from 24 sensors and 27 actuators. Table 3 provides the details of the sensors and actuators used in the water treatment process. The network data are packets communication between PLCs and SCADA. They have 18 features based on network attributes such as date, time, IPs, etc. This sub-part of the dataset is not used in this study. The dataset was collected, stored, and distributed on CSV files. The attacks on both physical and network were *injection* type attacks, i.e., on either the value of sensor or actuators.

The power and water system datasets were obtained using different field devices and operational environments. In the power system dataset, the majority of features are measurements of PMUs, whereas, in the water treatment dataset the events were collected by sensors and actuators. Differences in the data source provide varying data types: PMUs provide voltage, current phase angles, and magnitude, while sensors and actuators provide numerical or Boolean values.

4. Feature Selection

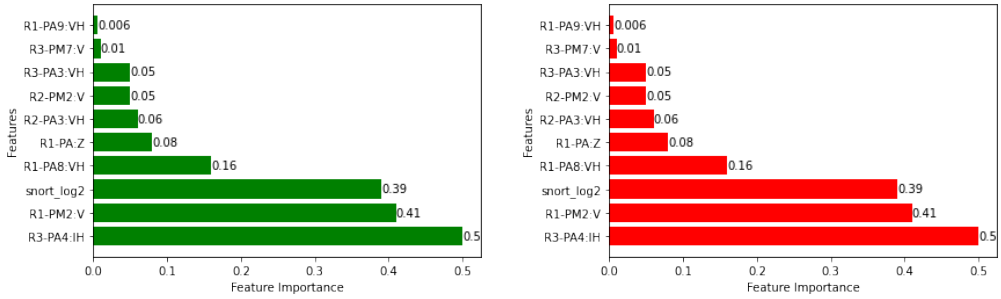
A feature represents a characteristic of any object. In ML, a sample is decomposed into a set of features before training and testing for tasks, such as classification, prediction, or clustering. The dimension of a feature vector can be small to large, and each feature has unequal discriminative potential. So, there is a need to select the best possible set of features without significantly impacting the model's performance. The different feature-selection approaches provide various ways to rank and select a set of features. The selection of features is performed in relation to the output variable that can be a class for classification or the predictive variable. Feature selection provides two key benefits. First, it helps to improve the model's performance in terms of accuracy, precision, and recall. Second, it reduces the computation cost (time and space) for the training, testing, and deployment of ML models. As a result of these two benefits, feature selection (as a part of feature engineering) is critical in the ML model. Based on the technique of feature ranking and selection, various feature selection methods are grouped into three main classes: the filter, wrapper, and embedded methods [28].

4.1. Filter Method

The filter method examines the dependency relationship of features X and class labels Y to select features based on their strength level with Y. The dependency strength level of the variables is calculated using traditional statistical tests, such as ANOVA, Z-test, T-test, chi-square, and Pearson Correlation Coefficient. Due to the individual evaluation of each feature, the filter method is also called univariate selection; it also speedily calculates and

easily interprets results [28]. In this study, under the filter method, the ANOVA F-value was used as the statistical test; the dataset features were ranked, and the best set of features was selected.

Figure 4 and the left four columns of Table 4 show the results of feature selection using a filter-for-power system and water treatment plant (SWaT dataset) respectively. The results are represented as feature name and score from the top (ten features) and bottom (ten features) of the feature rank list. In the case of the power system dataset, from Figure 4, we can observe that *magnitude*-related measurement of PMUs achieves larger scores and is top-ranked using the filter approach. Based on the value of the *magnitude* features in the dataset, we can observe that the larger values influence the statistical test. In contrast, *angle*-based features fall into a smaller value range (negative to a small positive value), and the statistical test was given a low score and was lower-ranked.



(a) Top features **(b)** Bottom features

Figure 4. Ten top and bottom features with individual score using filter method.

Table 4. Water Treatment Plant: Ten top and bottom features with their importance values using the filter and wrapper methods.

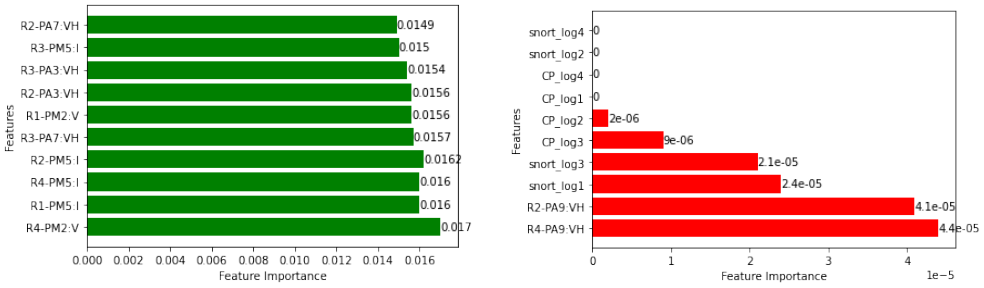
Filter Method				**Wrapper Method**			
Top Features		**Bottom Features**		**Top Features**		**Bottom Features**	
Feature	**Value**	**Feature**	**Value**	**Feature**	**Value**	**Feature**	**Value**
FIT401	6.281	FIT601	0.00066	FIT504	0.223181	P204	0.000403
FIT504	6.218	P602	0.00058	FIT401	0.125924	P206	0.000313
FIT503	6.105	P403	0.00008	P501	0.105114	P402	0.000114
UV401	6.076	P202	0.0	PIT502	0.070205	P403	0.000035
P501	6.075	P301	0.0	FIT503	0.063296	P202	0.000000
PIT501	5.972	P401	0.0	P102	0.040979	P301	0.000000
FIT501	5.906	P404	0.0	LIT301	0.040890	P401	0.000000
PIT503	5.899	P502	0.0	LIT101	0.030181	P404	0.000000
FIT502	5.860	P601	0.0	LIT401	0.027320	P502	0.000000
P402	5.550	P603	0.0	DPIT301	0.022423	P601	0.000000

In the SWaT dataset, the top three features are FIT401, FIT504, and FIT503, and all these are flow control sensors placed in the crucial stages, i.e., the 4-th and 5-th stages of a 6-stage process. Similarly, other top features also have critical roles and are found in later stages of the plant process. From Table 4, we can observe that the two bottom features are P601 and P60. These are two actuators placed in the last stage. Interestingly, these two were not implemented in SWaT, and the features selection correctly placed these at last. Other bottom features, P401, P404, and P502, are actuators. These were implemented as backups, and so, for this reason, they are not considered during attack events.

4.2. Wrapper Method

Compared to the filter method, features ranking is performed concerning a particular algorithm in the wrapper method. So, the best-selected feature set works well with the

machine-learning algorithm, and the feature set differs when the selection is made using another algorithm. Unlike the filter method, the wrapper feature selection process is costly in terms of time and space. Most wrapper methods use greedy search, which is not optimal, and suffer from false starts (wrongly choosing the first best feature) [28]. Figure 5 and the right four columns of Table 4 show the best-selected feature sets and their importance for the wrapper method for the power system dataset and the water treatment plant dataset (SWaT dataset), respectively.



(a) Top features **(b)** Bottom features

Figure 5. Ten top and bottom features with individual score using wrapper method.

In the case of the power system dataset, as shown in Figure 5, the features based on *magnitude* and *angle* are top-ranked in a nearly equal proportion, i.e., 6 and 4, respectively. So, the feature rank list differs from the filter method in which *magnitude*-related features were dominating. This study used a tree-based classifier as a wrapper method. In this approach, features are selected based on their impact on classification accuracy, rather than *number of features*. Similar behavior can be observed in the control log features listed at the bottom of the feature list. These features are Boolean and sparse, and their contribution to the classification is negligible, i.e., with an importance value of zero.

In the SWaT dataset, FIT401 and FIT504 are ranked as top features. The other top features, i.e., P501 and PIT502, are the actuator and the sensor for a pump and a pressure meter, respectively. The bottom features are similar to those from the filter methods, which verify the importance of ranking these features.

4.3. Embedded Method

The embedded method combines the techniques of the filter and wrapper approaches. The purpose of the combination is to take advantage of both approaches in terms of the speed and performance of the filter and wrapper methods, respectively. From the implementation perspective, feature selection becomes part of training in the embedded method. The algorithm starts training with the seed feature set (i.e., all features) and recursively selects a set of best features for the next round of training based on the importance of the features in the trained model [28]. The retraining continues until the predefined termination condition, e.g., based on the algorithm's convergence criteria or expected performance. The commonly used embedded methods are LASSO and RIDGE regression.

5. Imbalance Dataset: Issue and Solution

In a supervised ML case, if the training sample for each classification class is approximately equal, then the given dataset is considered imbalanced [29–31]. If the dataset is imbalanced, the training is highly influenced by the majority class sample (i.e., the class with the largest samples). Hence, the trained model lacks generalization in the real world and misclassifies the minor class.

The imbalance issue is more prominent in the cases such as this study, where the task is to detect a rare event, i.e., an anomaly, maliciousness, an attack, etc. In contrast, the normal events contribute the majority of the dataset. Both of our datasets are of an imbalanced nature, as can be verified from Table 2 and the bar plots in Figure 1. The power system

dataset has an imbalance ratio of 1:2.5. In contrast, the water treatment plant dataset has an imbalance ratio of 1:7.23, meaning that the samples for normal events are 70% and 88% of the total sample, respectively.

Imbalanced datasets are a major issue and create a bottleneck in machine learning, so there have been many methods to address and resolve the problem of training with imbalanced datasets. These techniques mainly work on two principles: oversampling and under-sampling. Oversampling suggests increasing the sample in the minority class, while under-sampling is the process of reducing the sample in the majority class. The under-sampling method goes against the basic principle of machine learning, which mainly aims to obtain more samples to achieve better performance. So, under-sampling is suitable only when the dataset has a very large sample for the majority class, and removing the sample will have a very low or no impact on training. In this study, we adopted the oversampling technique, given the limited number of samples, and focused on increasing the sample of the minority class.

5.1. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE is a minority-class oversampling method that creates synthetic examples. The synthetic examples are created by performing operations in data space and are nearly free from any particular application domain. The synthetic examples are plotted against the minority class samples, and the required samples, denoted as k, are randomly selected as k nearest neighbors [30].

5.2. Borderline-SMOTE

Borderline-SMOTE is also a minority oversampling method. There are two variants [32]: borderline-SMOTE1 and borderline-SMOTE2. Both the methods only oversample those minority samples on the borderline of class separation. The algorithm first finds the borderline samples from minority groups, then generates synthetic examples. It is assumed that borderline samples of minority classes are more prone to misclassification than samples far from the classification line.

5.3. Borderline Oversampling

Borderline Oversampling is similar to other oversampling methods that try to create synthetic samples around the classification line. Support Vector Machine (SVM) can be used to create a classification line and select boundary samples for oversampling [33]. First, the SVM model is trained on the complete dataset. Later, the trained model is used to identify the borderline, and synthetic/new samples for minority class are generated around the borderline. The number of samples, i.e., nearest neighbors, are generated either using interpolation or extrapolation depending upon the density of majority class instances around the borderline. This method differs from SMOTE, mentioned above, by choosing a new sample (i.e., nearest neighbor). SMOTE chooses randomly, while this method chooses the first k nearest neighbors.

5.4. Adaptive Synthetic (ADASYN) Sampling

Adaptive Synthetic (ADASYN) sampling incorporates *weighting* for oversampling of the minority class as per the difficulty level in learning [34]. The method claims to improve learning in two aspects, first, by reducing the bias induced by class imbalance, and second, by adapting to the classification decision boundary as per the difficult examples. The difficult examples are those samples from minority classes close to the decision boundary. They often result in false classifications. The algorithm starts by calculating the degree of class imbalance (d), i.e., $d = m_s/m_l$, where $d \in (0, 1)$ and m_s and m_l are the number of samples in the minority and majority classes respectively. The values of d in the used power system and water treatment plant datasets, respectively, are $0.4, 219$ and $0.1, 382$. The value of d is compared with the preset threshold value d-th for the maximum tolerated degree of class

imbalance. Further, the total required synthetic samples (G) is calculated, and then the algorithm generates those samples.

Apart from the aforementioned oversampling methods, new methods have been proposed in the recent literature for improving model performance with an imbalanced dataset. Elyan et al. [35] have proposed class decomposition-based SMOTE (CDSMOTE). The proposed method improves performance by taking two actions: first, to reduce the dominance of the majority class by applying class decomposition, and second, to increase the representation of the minority class by oversampling. Moreover, a two-step hybridization of minority oversampling (SMOTE) and a novel data cleaning method (Weighted Edited Nearest Neighbor rule, or WENN) was proposed in [36]. Fajardo et al. [37] have applied deep conditional generative models for learning to the distribution of minority classes and then generated synthetic samples for solving the class imbalance in the dataset to improve the model's performance. Similarly, Bellinger et al. [38] have proposed a new training approach of a deep learning model (CNN) which mixes three techniques (batch resampling, instance mixing, and soft labels) to create a robust model from a long-tailed or imbalanced dataset. Krawczyk et al. [39] have studied the issues of the imbalanced dataset for multi-class classification. The authors have proposed a two-step under-sampling approach; in the first step, a one-class SVM is trained for all classes. An evolutionary under-sampling approach is applied to each learned classifier in the second step. Using under-sampling on the set of support vectors instead of on the original dataset, the authors claimed significant computational and performance improvements.

All the methods mentioned above for handling class imbalances in learning are suitable for single-model-based learning algorithms. They can be extended to suit ensemble-based learning algorithms [39]. SVM is a good choice for dealing with imbalanced datasets [33]. All these oversampling methods were tested on the power system dataset, and the results are presented in Section 6.4 along with explanations.

6. Experiments and Results

This section provides details of various experiments conducted to analyze the performance of ML algorithms for feature selection and improvement of minority class detection for imbalance datasets. Figure 6 illustrates the steps, structures, and components of the conducted experiments.

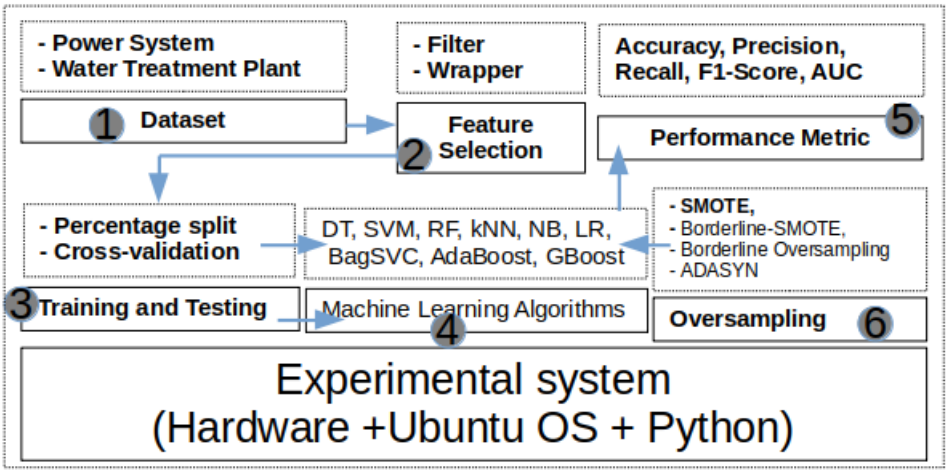


Figure 6. A summary of the steps, structures, and components of the conducted experiments.

These experiments were designed to test and validate the different hypotheses. For example, performance comparison for the train–test split vs. cross-validation, identifying the impact of the imbalanced dataset on performance, testing oversampling techniques to improve the performance, and finding the classifiers' performance on a different subset of features (ALL, Top10, Top20, Top30, Top40, and Top50). All the experiments were tested

against standard performance metrics such as accuracy, precision, recall, and F1-score. The Receiver Operating Characteristic(ROC) and Area Under ROC(AUC) are additionally used to show performance.

6.1. Experimental System

The experimental system was prepared with Ubuntu OS in the Python development environment. The python environment was prepared with required machine learning modules and frameworks such as Pandas, NumPy, matplotlib, CSV, and Scikit-learn [40].

6.2. Machine Learning Algorithms

In our work, ML algorithms were chosen based on their working principles. We tried to keep a diverse set of algorithms for a better understanding and performance comparison. For example, Naive Bayes (NB) works on conditional probability, while kNN applies a distance function to associate a node to a group or cluster [41]. Based on this, we initially selected nine algorithms and later, due to relatively much large training time, dropped the Bagging (SVC) and XBoost algorithms from further experiments. The selected algorithms were Decision Tree (DT), Support Vector Machine (SVM), Random Forest (RF), k-Nearest Neighbour (kNN), Naive Bayes (NB), Regression, Bagging, and Boosting. Training and testing with such a diverse set of algorithms helped the authors to understand and find suitable features and algorithms. All the algorithms were tested with default parameters available in the scikit-learn framework. However, parameter setting is explicitly mentioned wherever the default value changes. Some critical parameters for the best performing model, i.e., random forest, are the number of trees: 100; split method: Gini; and the minimum number of samples required to split: 2. Hyper-parameter-tuning finds the best value for the algorithm's parameters from the search space. This study did not perform hyper-parameter-tuning; however, this is a possible area of research for future work.

6.3. Training and Testing

The power system and SWaT datasets were divided into training and testing sets. Each algorithm was trained on the training set, while the performance evaluation of the model was completed on the testing set. Percentage split and cross-validation are two main methods for splitting the dataset into training and testing sets. The percentage split simply divides the original samples into two sets based on the given *percentage of the sample* to be considered for training and testing. However, the cross-validation divides the original samples into N folds containing equal numbers of samples. We used a 70/30 ratio for percentage split, while 10 folds (i.e., $N = 10$) were used with cross-validation. As cross-validation is an iterative process, the algorithm's performance was taken as the mean of N rounds of training and testing. In each round, $N - 1$ folds were used for training, and the remaining fold was used for testing. Training and testing in multiple folds provide diversity to the dataset, cross-validation provides robust training, and the trained model was generalized well on an unseen sample [42].

6.3.1. Percentage Split (70–30)

As mentioned earlier, based on the required percentage for training and testing samples, the percentage split method divides the samples into two sets. We used 70% for training and the remaining 30% for testing. This split method randomly selects the sample from the dataset for training. Training on a train–test dataset (from the split method) provides an approximate model, because randomly selected training samples do not represent actual data distribution. As such, the trained model suffers from over-fitting, i.e., it performs poorly on unseen samples. We trained and tested all nine algorithms on training and testing sets obtained from a percentage split (70–30%) to measure training time and approximate performance of FDI classification. Tables 5 and 6 show the precision, recall, F1-score, and accuracy of all algorithms for the power system and SWaT dataset, respectively. Except for accuracy, the other three metrics provide results for both classes

(Normal and FDI). From Table 5, we can observe that, with an accuracy score 92%, Random Forest performance is the best performed as an ensemble algorithm, while Decision Tree has an accuracy score of 85% and is the best performer as a single model. The accuracy value is biased towards the majority class, and the model suffers for the minority class. This is evident from the precision, recall, and F1-score value from Table 5 for both classes. The performance reduced by about 6–10% for best-performing classifiers. In the case of the SWaT dataset, all the classifiers have an accuracy above 95%. The precision, recall, and F1-score for the normal class is as per the accuracy but reduced for the FDI class for many classifiers. From Table 6, we can observe that kNN, DT, and RF show a perfect 100% score for all metrics; this is indicating over-fitting. Over-fitting can be attributed to the water treatment dataset having fewer features and many samples. Therefore, kNN, DT, and RF can memorize the class distribution for training data. We have investigated these three algorithms further with cross-validation, and the results are presented in Section 6.3.2.

Table 5. Classifiers' performance in the power dataset.

ML Model	Precision		Recall		F1-Score		Accuracy
	Normal	FDI	Normal	FDI	Normal	FDI	–
NB	0.71	0.33	0.98	0.02	0.82	0.04	0.70
SVM	**0.97**	0.29	0.01	**1.0**	0.01	0.46	0.30
kNN	0.86	0.70	0.88	0.66	0.87	0.68	0.82
DT	0.90	0.75	0.90	0.75	0.90	0.75	0.85
RF	0.91	**0.93**	0.98	0.78	**0.94**	**0.85**	**0.92**
Ada	0.72	0.53	0.96	0.10	0.82	0.16	0.71
LR	0.71	0.49	**1.0**	0.01	0.83	0.02	0.71

Table 6. Classifiers' performance in the water treatment dataset.

ML Model	Precision		Recall		F1-Score		Accuracy
	Normal	FDI	Normal	FDI	Normal	FDI	–
NB	0.96	0.98	1.0	0.70	0.98	0.82	0.96
SVM	0.96	0.99	1.0	0.71	0.98	0.83	0.96
kNN	1	1	1	1	1	1	1
DT	1	1	1	1	1	1	1
RF	1	1	1	1	1	1	1
BagSVC	0.96	0.99	1	0.66	0.98	0.80	0.96
LR	0.95	0.99	1	0.62	0.97	0.76	0.95

In critical infrastructure, decision making needs to be quick. So, a low prediction time is required from the machine learning model. In this study, time is one of the key performance metrics. So, the time taken for training and testing by each algorithm is measured. The training time will help find a suitable algorithm for the power system or other critical infrastructure. Training time is also essential because data generation is fast, and the models often require retraining. A model with a lower training time will be more suitable. Table 7 presents the training and testing times (in seconds) of all nine algorithms. As mentioned earlier, Naive Bayes (NB) is a fast and probabilistic algorithm, because it uses prior probability values to calculate the posterior. Probability values can be calculated in advance, so NB training is faster than others. However, *conditional independence* is one of the critical assumptions that attributes need in order to be satisfied. *Bagging* is an ensemble method that creates multiple base models on the subset of the dataset. These subsets are created using random sampling. Table 7 shows that NB has the lowest training time of all algorithms, while Bagging (with SVC) has the highest training time. Hence, the highest training time is the accumulation of time taken for dataset generation, multiple model training, and testing.

Table 7. Training and testing time of machine-learning algorithms.

Classifiers	NB	SVC	kNN	DT	RF	Ada	BSVC	LR	XGB
Time (S)	0.132	9.770	7.280	3.580	17.900	11.800	533.00	1.140	62.0

6.3.2. Testing with Cross-Validation

The previous section shows the detection performance and training time of algorithms for the training and testing set created using a percentage split. With this initial estimation, the algorithms were further trained and tested with 10-folds cross-validation to know the trained model's generalization capacities. We compared the detection accuracy of all algorithms for the percentage split and cross-validation of the power-system dataset. We have two key observations; first, five algorithms (DT, SVC, kNN, GB, and RF) achieved lower accuracy in the 10-fold cross-validation than the percentage split. Second, four other algorithms (Adaboost, Bagging (ensemble), LR, and NB) had minimum impact, i.e., accuracy either reduced with smaller margin or remained constant. Based on these two observations, we can conclude that the former five algorithms over-fit inherently, while the latter four algorithms have the inbuilt function to overcome over-fitting during training. Considering this outcome, for the SWaT dataset, we performed training with 10-fold cross-validation for the three most over-fitted classifiers, i.e., kNN, DT, and RF. Figure 7 shows the AUC of all classifiers for 10 folds training. For RF, although 4 passes have an AUC value of 1.0, the mean performance of all three classifiers reduced to 0.79, 0.75, and 0.81 from 1.0 for kNN, DT, and RF, respectively.

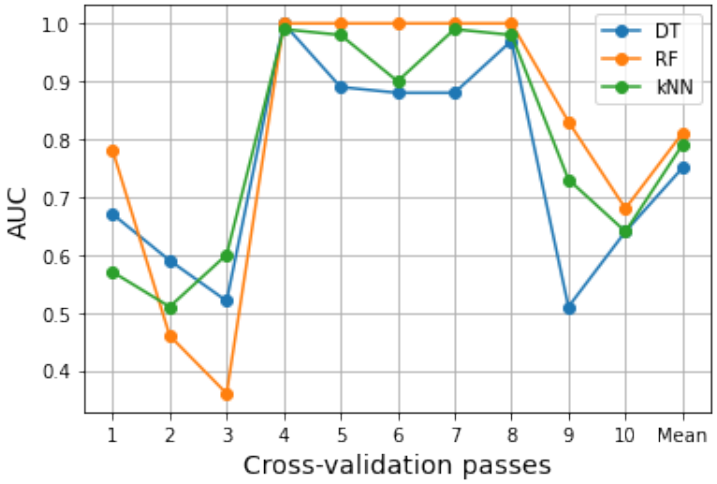
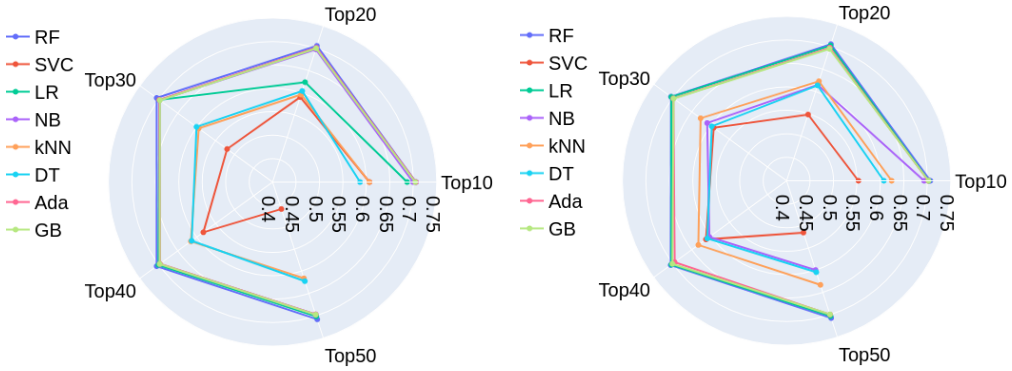


Figure 7. AUC with 10-fold cross-validation for the water treatment plant dataset.

The previous section presented the outcome of the filter and wrapper methods. One further key objective of this study is to test the performance of all algorithms on the selected set of features. For this, we experimented only on the power-system dataset. A total of ten datasets were created to train and test different machine-learning algorithms using sets of the selected top features, five sets each from the filter and wrapper method. The experimental results are shown as Top10, Top20, Top30, Top40, and Top50 for feature selection sets, and ALL represents all features. These 11 sets of the dataset were used to train and test all algorithms in 10-fold cross-validation. Figure 8a shows the results for the filter method sets. The detection accuracy in Adaboost, SVC, and GB decreased, while the detection accuracy in DT, kNN, and LR increased. The performance of RF and NB was unaffected (insignificant change) by the change in the number of features.



(a) Filter method **(b)** Wrapper method

Figure 8. Accuracy of classifiers with selected features for power system dataset.

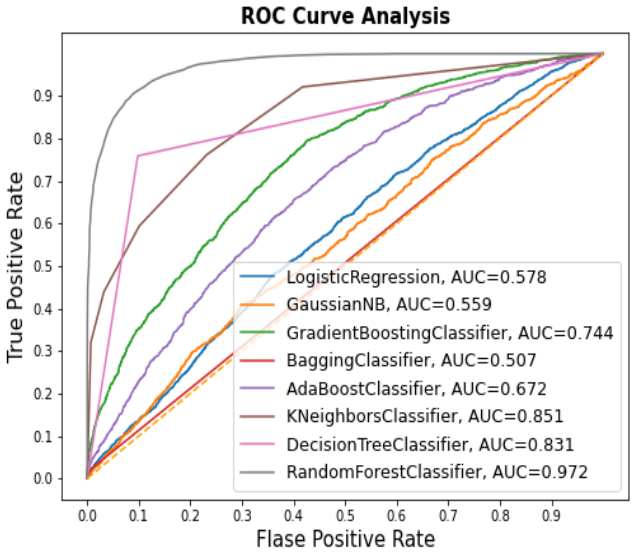
Similarly, five datasets were created using the feature ranking of the wrapper method. Further, using the 10-fold cross-validation approach, all algorithms were trained and tested on all five sets. Figure 8b shows the performance of all algorithms. Unlike the filter method case, there was no change in the performance of Logistic Regression; the performance of Naive Bayes decreased significantly. In either selection method, kNN had a similar pattern, i.e., accuracy increased with the number of selected features. However, there was no clear pattern in the performance change of Adaboost, RF, GB, DT, and SVC. The performance of RF and DT decreased and had the lowest accuracy with thirty top features, while there were no significant changes in accuracy with other sets of features (top10, top20, top40, and top50). In other groups, the accuracy of SVC and Adaboost did not seem related to the number of features.

6.4. Imbalance Dataset and Impact

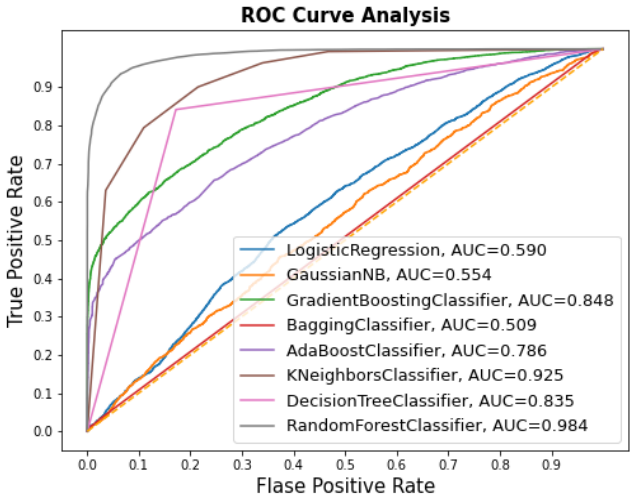
As discussed in Section 5, with an imbalanced dataset, ML models suffer performance degradation while making predictions about minority classes. This is because a model learns mainly from the majority class or is over-fitted to the majority class. Accuracy is the most-used metric for measuring the performance of machine-learning algorithms, but it is not suitable for imbalanced datasets [30]. Values from Tables 5 and 6 verify the performance degradation of the model with more robust metrics such as precision, recall, and F1-score.

The two main approaches for handling the imbalanced dataset are oversampling the minority class and under-sampling the majority class. While oversampling is suitable for maximum use-cases, under-sampling only suits when the majority class has many samples and the minority class also has enough samples to represent the nature of the distribution. In this study, we have adopted the oversampling approach, and the minority class is oversampled using four different sampling techniques. SMOTE is the main technique for oversampling, and the other three, i.e., Borderline-SMOTE, Borderline-SMOTE with SVM, and Adaptive Synthetic Sampling, are variants of SMOTE. As explained in Section 5, in borderline-SMOTE, only the borderline sample from the minority class is used for oversampling. In the original borderline-SMOTE algorithm [32], kNN is used for sample selection, while in the modified version (borderline-SMOTE with SVM [33]), SVM is used for sample selection. To understand and highlight the impact of the imbalanced dataset, we have used AUC as a performance metric. After applying each oversampling technique, we trained all ML algorithms on the imbalanced dataset and again trained the algorithms. Figure 9 depicts the algorithms' AUC values with the imbalanced dataset, i.e., Figure 9a and after balancing the dataset (making an equal sample for both classes by oversampling minority class, i.e., FDI class) using SMOTE, shown in Figure 9b. From Figure 9, it is evident that imbalance datasets have varying impacts on the different types of ML algorithms. These are obvious, because each performs training differently. Further, this observation can be broken down into two: first, some algorithms are not impacted (they can have high performance, i.e., DT and RF, or low-performance, i.e., NB and LR) by the ratio of samples

for each class, so oversampling also fails to impact performance. Second, some algorithms (GradientBoost, AdaBoost, and kNN) have a high impact on the imbalanced dataset, and so the performance of these algorithms improves after oversampling. Table 8 shows the AUC values of different classifiers for the imbalanced dataset and after applying four selected oversampling methods. In Table 8, borderline-SMOTE and SVM-based borderline-SMOTE is coded as BSMOTE and BSMOTE-SVM. From Table 8 we can observe that all oversampling techniques improve the AUC values of almost all classifiers. The *magnitude* of improvement depends upon the type of algorithm used. As mentioned previously, the best improvement, i.e., 10–12%, was observed for GradientBoost, and AdaBoost algorithms, while kNN had a 6–8% improvement with different oversampling techniques. DT and RF are considered robust against imbalanced datasets, but these algorithms achieved a 2–3% performance improvement after applying oversampling.



(**a**) ROC with imbalance dataset



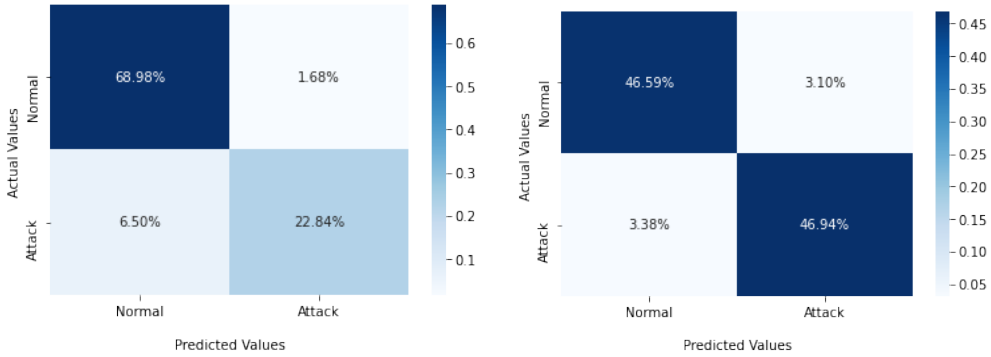
(**b**) ROC after SMOTE

Figure 9. ROC for classifiers with imbalance and SMOTE balance power system dataset.

Table 8. Comparison of classifiers (on AUC %) for imbalanced datasets and different oversampling methods for the power system dataset.

Classifiers	Imbalance	Smote	Bsmote	Bsmote-Svm	Adasyn
LR	0.592	0.590	0.550	0.595	0.565
GNB	0.559	0.548	0.578	0.544	0.596
GBoost	0.747	**0.847**	**0.839**	**0.839**	**0.845**
BagSVC	0.504	0.505	0.512	0.515	0.513
AdaBoost	0.672	**0.783**	0.777	0.747	0.785
kNN	0.855	**0.924**	**0.917**	**0.926**	**0.910**
DT	0.826	0.830	0.835	0.847	0.831
RF	**0.974**	**0.984**	0.982	0.988	0.983

In this study, we used robust performance metrics such as precision, recall, and F1-score calculated using a confusion matrix. In addition to this, to visualize the model's performance, we selected the best performing model in both the imbalanced and balanced dataset and plotted the confusion matrix. Figure 10 shows the results for both cases. As shown in Figure 10a,b, the detection performance of the attack class improved with a balanced dataset but decreased for normal class. However, improved attack detection is critical and required for critical infrastructures.



(a) Imbalance dataset **(b)** Balance dataset (SMOTE)

Figure 10. Confusion matrix for best model (RF) on imbalance and SMOTE balanced power system dataset.

6.5. Comparison with Previous Works

In this section, we compare the performance of our best model with those described in the existing literature. Jingyu wang et al. [43] used deep autoencoder to detect data manipulation attacks in power systems. Adhikari et al. [44] combined Non-Nested Generalized Exemplars (NNGEs) and the STate Extraction Method (STEM) for cyber-attack event detection. Defu Want et al. [20] divided the features as per each PMU and then used an ensemble approach to combine the results of five classifiers (the authors trained four classifiers on four PMUs data and the fifth with combined features). Table 9 shows the performance comparison of the best model of earlier studies and this study. The performance of this study is shown as an AUC value. AUC is a robust metric that represents the total area under ROC (trade-off between True Positive Rate (TPR) and False Positive rate (FPR)). A higher AUC value indicates better classification performance of the model. This study has achieved an AUC value of 0.984, which is better than the existing literature.

Table 9. Comparison with previous works on FDIA detection for the power system dataset.

Work	Algo.	Accuracy	Precision	Recall	F1-Measure	AUC
[43]	DAE	0.941	0.996	0.886	0.938	-
	XGBoost	0.848	0.990	0.703	0.82	-
[44]	–	-	0.96	-	0.95	-
[20]	Ensemble	0.9391	-	-	-	-
Current Study	RF	-	-	-	-	**0.984**

7. Conclusions and Future Scope

We examined and improved the performance of machine-learning algorithms for detecting FDIAs in critical infrastructure by determining the best features and mitigating imbalanced dataset problems. Performance improvement was tested and validated through various experimental results. These experiments included feature selection methods, oversampling techniques, and training and testing ML algorithms on two popular datasets related to power systems and water treatment plants. Our results show that the performance of algorithms varies significantly depending on the feature selection and the number of features. For example, the performance of NB is unaffected by increasing the number of features in the filter method while decreasing the number wrapper features. We also found that selection methods rank features differently. We found that RF is generally suitable for building an FDIA detector based on detection performance and training time trade-offs. Additionally, model training with 10-fold cross-validation is suitable because it highlights the over-fitting issues. Moreover, we analyzed the impact of the imbalanced dataset and applied minority oversampling techniques to improve detection performance.

New sampling techniques based on deep learning and hybrid sampling approaches are proposed in the literature [35–39]. Future studies can explore these recent techniques with the power system and other critical infrastructure datasets. The binary classification formulation in this study can be further divided and reformulated as a multi-class classification for training machine-learning algorithms. Moreover, consideration of the space and computation requirements of critical infrastructures can motivate new research objectives.

Author Contributions: Conceptualization: A.K. and B.J.C.; data curation: A.K.; formal analysis: A.K.; funding acquisition: S.J.; methodology: A.K. and B.J.C.; project administration: S.J. and B.J.C.; resources: A.K.; software: A.K.; supervision: B.J.C.; validation: A.K., N.S. and B.J.C.; visualization: A.K.; writing—original draft: A.K.; writing—review and editing: B.J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the National Research Foundation (NRF), Korea (2019R1C1C1007277) and the ITRC (Information Technology Research Center) support program (IITP-2020-2020-0-01602) supervised by the IITP (Institute for Information and Communications Technology Planning and Evaluation). This research was also supported by the Cardiff University HEFCW GCRF Small Project: Secure, Low-Cost, and Efficient Energy Solution (SP113).

Conflicts of Interest: The authors declare no conflicts of interest.

Sample Availability: The modified version of the Power system dataset is available on Github for comparison, but citing the original authors' work while using the dataset is also suggested. The SWaT dataset is only available on request to the original author.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
AI	Artificial Intelligence
ARFF	Attribute-Relation File Format
AUC	Area Under Curve
CPPS	Cyber-Physical Power System
CPU	Central Processing Unit
CSV	Comma-Separated Value
DT	Decision Tree
FDIA	False Data Injection Attack
GB	Gradient Boost
GUI	Graphical User Interface
HMI	Human–Machine Interfaces
ICT	Information and Communication Technology
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IED	Intelligent Electronic Device
kNN	k Nearest Negibour
LR	Linear Regression
LTS	Long Term Support
ML	Machine Learning
NB	Naive Bayes
OCSVM	One-Class Support Vector Machine
PCA	Principal Component Analysis
PDC	Power Distribution Center
PLC	Programmable Logic Controllers
PMU	Phasor Measurement Unit
RF	Random Forest
ROC	Receiver Operating Characteristic
SCADA	Supervisory Control and Data Acquisition
SVC	Support Vector Classifier
SVM	Support Vector Machine

References

1. Corallo, A.; Lazoi, M.; Lezzi, M. Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts. *Comput. Ind.* **2020**, *114*, 103165. [CrossRef]
2. Griffor, E.R.; Greer, C.; Wollman, D.A.; Burns, M.J. Framework for cyber-physical systems: Volume 1, overview. *NIST SP* **2017**. [CrossRef]
3. Rodofile, N.R.; Radke, K.; Foo, E. Extending the cyber-attack landscape for SCADA-based critical infrastructure. *Int. J. Crit. Infrastruct. Prot.* **2019**, *25*, 14–35. [CrossRef]
4. Khanna, K.; Panigrahi, B.K.; Joshi, A. AI-based approach to identify compromised meters in data integrity attacks on smart grid. *IET Gener. Transm. Distrib.* **2017**, *12*, 1052–1066. [CrossRef]
5. Maleh, Y.; Shojafar, M.; Darwish, A.; Haqiq, A. *Cybersecurity and Privacy in Cyber Physical Systems*; CRC Press: Boca Raton, FL, USA, 2019.
6. Liang, G.; Weller, S.R.; Zhao, J.; Luo, F.; Dong, Z.Y. The 2015 ukraine blackout: Implications for false data injection attacks. *IEEE Trans. Power Syst.* **2016**, *32*, 3317–3318. [CrossRef]
7. Reeder, J.R.; Hall, C.T. *Cybersecurity's Pearl Harbor Moment: Lessons Learned from the Colonial Pipeline Ransomware Attack*; Government Contractor Cybersecurity: Washington, DC, USA, 2021.
8. Gönen, S.; Sayan, H.H.; Yılmaz, E.N.; Üstünsoy, F.; Karacayılmaz, G. False Data Injection Attacks and the Insider Threat in Smart Systems. *Comput. Secur.* **2020**, *97*, 101955. [CrossRef]
9. Aoufi, S.; Derhab, A.; Guerroumi, M. Survey of false data injection in smart power grid: Attacks, countermeasures and challenges. *J. Inf. Secur. Appl.* **2020**, *54*, 102518. [CrossRef]
10. Pan, S.; Morris, T.; Adhikari, U. Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Trans. Smart Grid* **2015**, *6*, 3104–3113. [CrossRef]
11. Goh, J.; Adepu, S.; Junejo, K.N.; Mathur, A. A dataset to support research in the design of secure water treatment systems. In *International Conference on Critical Information Infrastructures Security*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 88–99.

12. Guan, Z.; Sun, N.; Xu, Y.; Yang, T. A comprehensive survey of false data injection in smart grid. *Int. J. Wirel. Mob. Comput.* **2015**, *8*, 27–33. [CrossRef]
13. Liang, G.; Zhao, J.; Luo, F.; Weller, S.R.; Dong, Z.Y. A review of false data injection attacks against modern power systems. *IEEE Trans. Smart Grid* **2016**, *8*, 1630–1638. [CrossRef]
14. Musleh, A.S.; Chen, G.; Dong, Z.Y. A survey on the detection algorithms for false data injection attacks in smart grids. *IEEE Trans. Smart Grid* **2019**, *11*, 2218–2234. [CrossRef]
15. Cao, J.; Wang, D.; Qu, Z.; Cui, M.; Xu, P.; Xue, K.; Hu, K. A Novel False Data Injection Attack Detection Model of the Cyber-Physical Power System. *IEEE Access* **2020**, *8*, 95109–95125. [CrossRef]
16. Maglaras, L.A.; Jiang, J. Intrusion detection in SCADA systems using machine learning techniques. In Proceedings of the 2014 Science and Information Conference, Las Vegas, NV, USA, 25–26 April 2014; pp. 626–631.
17. Esmalifalak, M.; Liu, L.; Nguyen, N.; Zheng, R.; Han, Z. Detecting stealthy false data injection using machine learning in smart grid. *IEEE Syst. J.* **2014**, *11*, 1644–1652. [CrossRef]
18. Yan, J.; Tang, B.; He, H. Detection of false data attacks in smart grid with supervised learning. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 1395–1402.
19. Wang, Y.; Amin, M.M.; Fu, J.; Moussa, H.B. A novel data analytical approach for false data injection cyber-physical attack mitigation in smart grids. *IEEE Access* **2017**, *5*, 26022–26033. [CrossRef]
20. Wang, D.; Wang, X.; Zhang, Y.; Jin, L. Detection of power grid disturbances and cyber-attacks based on machine learning. *J. Inf. Secur. Appl.* **2019**, *46*, 42–52. [CrossRef]
21. Panthi, M. Anomaly Detection in Smart Grids using Machine Learning Techniques. In Proceedings of the 2020 First International Conference on Power, Control and Computing Technologies (ICPC2T), Raipur, India, 3–5 January 2020; pp. 220–222.
22. Ahmed, C.M.; Zhou, J.; Mathur, A.P. Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in cps. In Proceedings of the 34th Annual Computer Security Applications Conference, San Juan, PR, USA, 3–7 December 2018; pp. 566–581.
23. Dutta, A.K.; Negi, R.; Shukla, S.K. *Robust Multivariate Anomaly-Based Intrusion Detection System for Cyber-Physical Systems. International Symposium on Cyber Security Cryptography and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 86–93.
24. Jahromi, A.N.; Karimipour, H.; Dehghantanha, A.; Choo, K.K.R. Toward Detection and Attribution of Cyber-Attacks in IoT-enabled Cyber-physical Systems. *IEEE Internet Things J.* **2021**. [CrossRef]
25. Begli, M.; Derakhshan, F.; Karimipour, H. A layered intrusion detection system for critical infrastructure using machine learning. In Proceedings of the 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE), UOIT, ON, Canada, 12–14 August 2019; pp. 120–124.
26. Dick, K.; Russell, L.; Souley Dosso, Y.; Kwamena, F.; Green, J.R. Deep learning for critical infrastructure resilience. *J. Infrastruct. Syst.* **2019**, *25*, 05019003. [CrossRef]
27. Rodofile, N.R.; Schmidt, T.; Sherry, S.T.; Djamaludin, C.; Radke, K.; Foo, E. Process control cyber-attacks and labelled datasets on S7Comm critical infrastructure. In *Australasian Conference on Information Security and Privacy*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 452–459.
28. Kotsiantis, S. Feature selection for machine learning classification problems: a recent overview. *Artif. Intell. Rev.* **2011**, *42*, 157–176. [CrossRef]
29. He, H.; Ma, Y. *Imbalanced Learning: Foundations, Algorithms, and Applications*; Wiley-IEEE Press: Hoboken, NJ, USA , 2013.
30. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
31. Chawla, N.V. Data mining for imbalanced datasets: An overview. In *Data Mining and Knowledge Discovery Handbook*; Springer: Berlin/Heidelberg, Germany, **2009**; pp. 875–886.
32. Han, H.; Wang, W.Y.; Mao, B.H. Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*; Springer: Berlin/Heidelberg, Germany, **2005**; pp. 878–887.
33. Nguyen, H.M.; Cooper, E.W.; Kamei, K. Borderline over-sampling for imbalanced data classification. *Int. J. Knowl. Eng. Soft Data Paradig.* **2011**, *3*, 4–21. [CrossRef]
34. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008; pp. 1322–1328.
35. Elyan, E.; Moreno-Garcia, C.F.; Jayne, C. CDSMOTE: class decomposition and synthetic minority class oversampling technique for imbalanced-data classification. *Neural Comput. Appl.* **2021**, *33*, 2839–2851. [CrossRef]
36. Guan, H.; Zhang, Y.; Xian, M.; Cheng, H.D.; Tang, X. SMOTE-WENN: Solving class imbalance and small sample problems by oversampling and distance scaling. *Appl. Intell.* **2021**, *51*, 1394–1409. [CrossRef]
37. Fajardo, V.A.; Findlay, D.; Jaiswal, C.; Yin, X.; Houmanfar, R.; Xie, H.; Liang, J.; She, X.; Emerson, D. On oversampling imbalanced data with deep conditional generative models. *Expert Syst. Appl.* **2021**, *169*, 114463. [CrossRef]
38. Bellinger, C.; Corizzo, R.; Japkowicz, N. Calibrated Resampling for Imbalanced and Long-Tails in Deep Learning. In *International Conference on Discovery Science*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 242–252.

39. Krawczyk, B.; Bellinger, C.; Corizzo, R.; Japkowicz, N. Undersampling with support vectors for multi-class imbalanced data classification. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–7.
40. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
41. Shalev-Shwartz, S.; Ben-David, S. *Understanding Machine Learning: From theory to Algorithms*; Cambridge University Press: Cambridge, UK, 2014.
42. Shao, J. Linear model selection by cross-validation. *J. Am. Stat. Assoc.* **1993**, *88*, 486–494. [CrossRef]
43. Wang, J.; Shi, D.; Li, Y.; Chen, J.; Ding, H.; Duan, X. Distributed framework for detecting PMU data manipulation attacks with deep autoencoders. *IEEE Trans. Smart Grid* **2018**, *10*, 4401–4410. [CrossRef]
44. Adhikari, U.; Morris, T.H.; Pan, S. Applying non-nested generalized exemplars classification for cyber-power event and intrusion detection. *IEEE Trans. Smart Grid* **2016**, *9*, 3928–3941. [CrossRef]