Performance analysis of machine learning algorithms and the effect of data preprocessing on the classification of partial discharge in electrical switchgear

Abstract

Insulation failure is one of the most critical faults in any electric switchgear. One of the foremost factors resulting in insulation degradation is partial discharge (PD), which can have multiple sources of origin. Being able to recognize the source of the detected PD signal data will help in the optimal treatment of the respective fault. In this paper, the dataset corresponding to five different types of PD defects is used. Five different types of machine learning algorithms are implemented on the dataset and their performance is compared based on PD pattern recognition accuracy and computation time to find the best among them. After assessing the performance, feature engineering techniques are then implemented, and the performance of the algorithms is further evaluated for the updated dataset. Accuracy percentages as high as 99.97 were obtained in the case of the Random Forest Classifier, which was further improved to 99.99 when the dataset was preprocessed using feature engineering.

1. Introduction

Continuity of power supply is an imperative element in contemporary society's workflow. Any kind of failure in any part of the system that can result in discontinuity of power supply can lead to loss of workflow, almost invariably resulting in loss of revenue and consumer discomfort. One of the most prominent causes of failure is insulation degradation, which can occur in different power system entities like switchgear, cables, and power transformers. Researchers have been incessantly researching the reasons behind it and have deduced multiple reasons for the same. One of the foremost factors resulting in insulation degradation is PD. PD may be defined as a concentration of electrical discharges that, rather than ultimately bridging the insulation, only partially traverses through it [1]. This kind of fault is called an incipient fault [2], slowly and gradually resulting in complete insulation degradation due to repetitive action.

There can be different factors which may result in the initiation of PD. Voids present in the insulation, which can be gas or oil-filled, having a lower dielectric strength than the insulation material can lead to PD initiation. There can be metal protrusions. Electric tree formation may start from sharp conducting particles. Sharp discharges may occur on an oil or liquid surface if the tangential component of the electric field is very high on the dielectric surface. Discharges may be initiated on poorly grounded elements in or near HV circuits. There may be contact discharges, which can occur in cases of bad contact. PD can progressively damage insulation materials, leading to their eventual failure and costly outages. So, it's important to figure out where the PD is coming from, find it, and, if necessary, get rid of it. If these PDs may be detected, their source of origin can be known, and the position of fault occurrence can be localized, then preventive measures can be taken well in advance. The information is useful at the time of commissioning of the equipment and during running

conditions as it allows one to anticipate continuous service behaviour under similar operating conditions.

PD can be detected and measured by analysing the various physical phenomena which accompany its production, like optical effects, acoustic effects, and electrical signals. The signals can be measured using appropriate sensors. The measured signals are then preprocessed appropriately to make them suitable for use as input data in the classification algorithm. The data can be visually analyzed to get a sense of how different sources lead to different patterns. The patterns, though mostly unique, are difficult to segregate manually as they have unclear boundaries where two or more sources may have a few identical patterns. Therefore, for this purpose, different machine learning algorithms have been proposed. These algorithms can broadly be classified into shallow and deep algorithms. The shallow algorithms work well with tabulated data, but to get tabulated data, a process called feature extraction must be done on the PD data first. Then, the PD data can be used as input to the algorithms. But deep algorithms have the advantage of automatically extracting features, which makes the whole process automatic.

A large number of machine learning algorithms have been implemented in recent history, and their number is continuously increasing. Different algorithms show varying performances when dealing with datasets from different fields. In this paper, we'll look at five different kinds of algorithms: Linear Discriminant Analysis, K-Nearest Neighbour, Decision Tree Classifier, Random Forest Classifier, and Gradient Boost Algorithm. We'll compare how well they work based on how accurate they are and how long it takes to run them. Then, we will see the effect of feature engineering on classification performance.

2. PD Dataset:

In this paper, the data sets were taken from an online source and analysed through a software tool called PDflex, coded in Matlab R2016b [3]. The measuring set-up described in [4] based upon conventional electrical methods as defined by standard IEC270 [1] is used in obtaining the database. The database is stored in waveform (.wfm) format. Using this, the software tool computes the values of charge and energy by applying the methods described in [5][6]. Apart from charge and energy, eight other parameters, namely time stamp, rise time, fall time, peak current, Energy/charge ratio, Energy/peak current ratio, peak current/charge ratio, and phase (°), are used in the classification process. All these parameters are obtained for five different types of fault, namely Floating Electrode Discharge, Free Moving Particle Discharge, Internal Discharge, Corona Discharge, and Surface Discharge. Table 2.1 gives the number of readings across different samples for each type of fault.

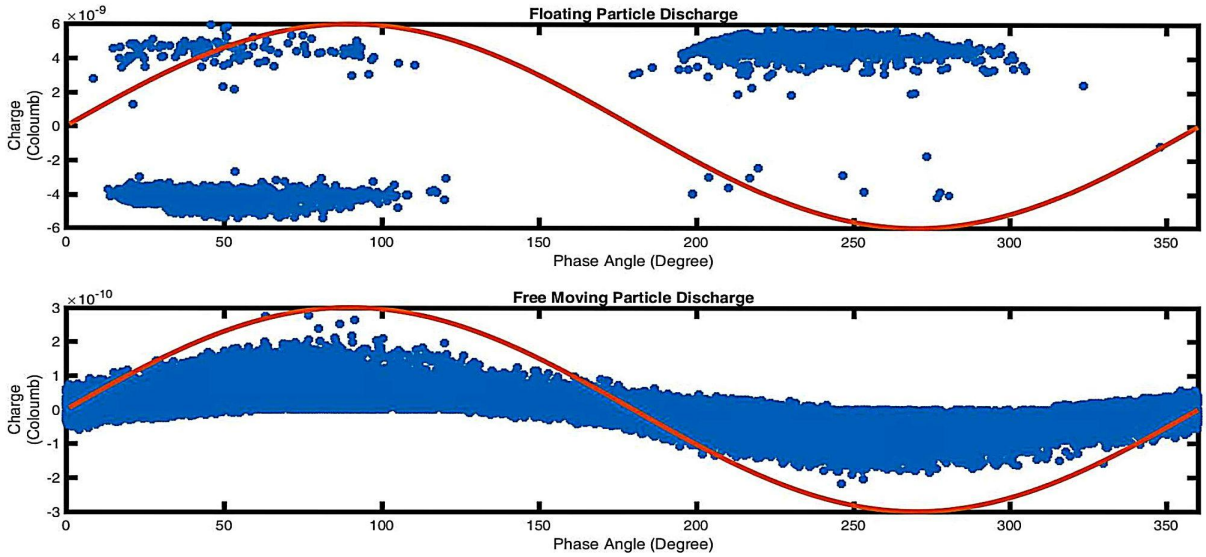
S.R. No.	Name of Defect	Size of Dataset
1.	Floating Electrode Discharges	14099x10
2.	Free Moving Particle Discharge	20000x10

3.	Internal Discharges	10000x10
4.	Surface Discharge	14927x10
5.	Corona Discharge	10000x10

Table 2.1

A floating electrode is a metallic object in the vicinity of a high voltage electric field that acquires a stray voltage depending on the level of capacitive coupling. If the voltage acquired by the metallic body is sufficient enough to cause a partial flashover to the main electrode or ground, or induce corona around the body, then the PD from the metallic object appears, which is termed floating electrode discharge.[7]. Gas-insulated switchgear contains free conductive particles which may emerge during production, transportation, assembly, and operation that can cause breakdown or flashover. This is termed "free metal particle discharge." [8], Internal discharges occur within the insulation groundwall, inside small voids [9], whereas surface discharges occur across the surface of the insulation. The term "corona" applies to a special category of PD that occurs in cases where solid insulating surfaces are absent or are far removed from the discharge zone.[10]

The data sets corresponding to different types of defects when plotted exhibit certain unique characteristics, based on which they can be classified or segregated. This is evident when Phase Resolved Data is plotted for different sets. For creating a Phase-resolved PD pattern, each PD pulse is recorded along with the phase angle related to the test voltage at which it occurred. Figure 2.1 shows phase-resolved plots between charge and phase corresponding to each type of defect. Since the source of the data sets was known, it was possible beforehand to identify the clusters that exist in each of the data sets.



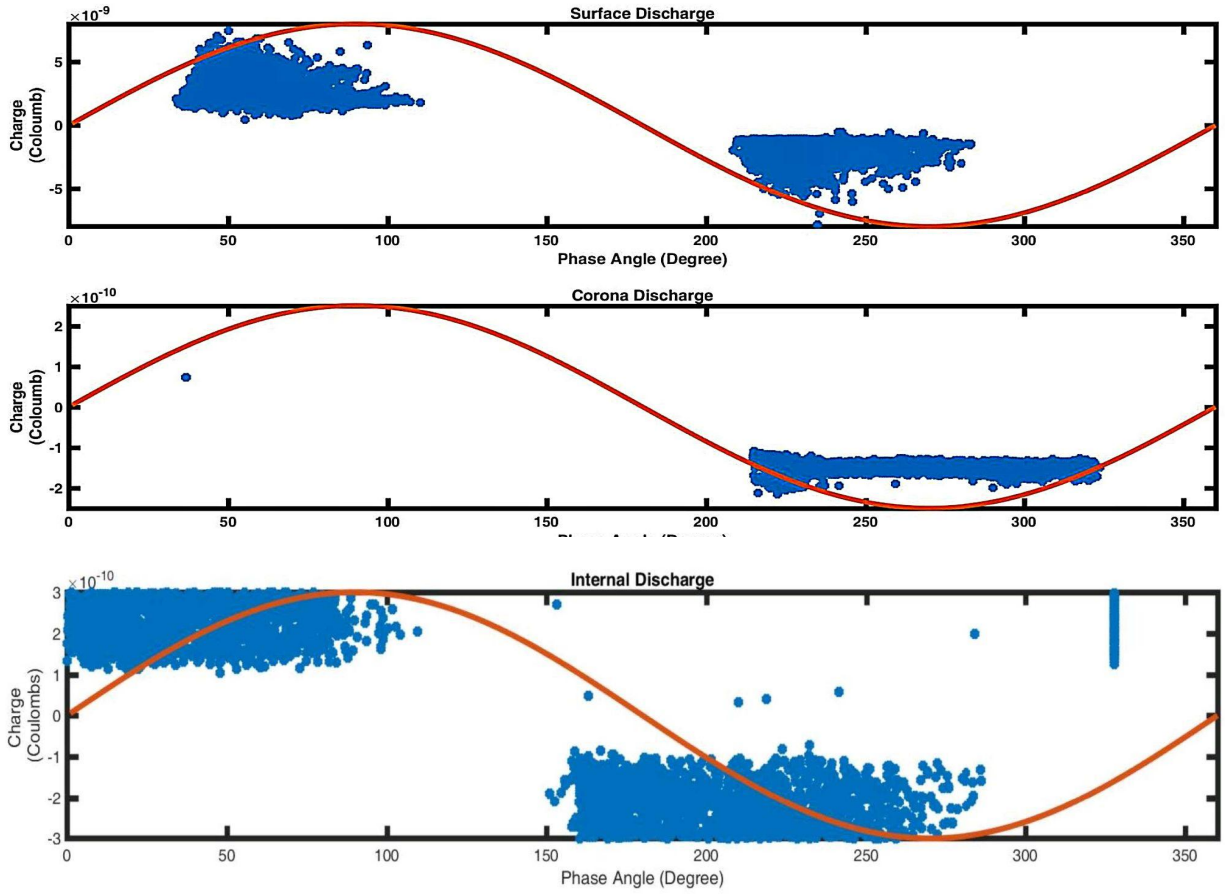


Figure 2.1

3. PD data preprocessing and pattern recognition tools:

A total of five machine learning algorithms have been used in this paper for PD defect classification. A brief description of all the algorithms is below.

Linear Discriminant Analysis (LDA):

The linear discriminant analysis estimates the probability that a new set of inputs will belong to every class. The output class is the one that has the highest probability. LDA uses Bayes' Theorem to estimate the probabilities.

If $X=(x_1 \ldots x_p)$ is drawn from a multivariate Gaussian distribution. k is the number of classes, and Y is the response variable.

Pik is the prior probability: the base probability of each class as observed in the training data.

$Pr(X = x \mid Y = k)$ is the posterior probability: the probability that a given observation is associated with the Kth class.

Let $f_k(X) = Pr(X = x \mid Y = k)$ is probability density function of X for an observation x that belongs to Kth class.

Then, as per Bayes' theorem,

$$Pr(Y = k \mid X = x) = \frac{pi_k f_k(x)}{\sum_{i=1}^{K} pi_k f_k(x)}$$

To calculate the posterior probability, prior pik and density function $f_k(X)$ values are required.
Prior probability (pik) can be calculated by taking a random sample of Ys and computing the fraction of the training observations that belong to the Kth class.
The probability density function of x is assumed to be multivariate Gaussian with a class means m_k and a common covariance matrix sigma.
As a formula, multi-variate Gaussian density is given by:

$$f(x|pi_k) = \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x - m_k)^T|\Sigma|^{-1}(x - m_k))$$

$|\Sigma|$ = determinant of the covariance matrix (same for all classes)
m_k = class means
On the basis of this density function, the linear score function can be derived as

$$\delta_k(x) = x^T\Sigma^{-1}m_k - \frac{1}{2}m_k{}^T\Sigma^{-1}m_k + logpi_k$$

The sample unit is classified into the class that has the highest linear score function for it.

K-Nearest Neighbour (KNN):

In this algorithm, a class label is assigned based on a majority vote. The label that is most frequently represented around a given data point is used. The majority vote technically requires a majority of greater than 50%, which primarily works when there are only two categories. But when there are multiple classes, like four categories, there is not necessarily a need for 50% of the vote to make a conclusion about a class. Here, a class label could be assigned with a vote of greater than 25%.

The goal of the k-nearest neighbour algorithm is to identify the nearest neighbours of a given query point so that a class label can be assigned to that point. To do this, KNN requires determining the distance metrics. To determine which data points are closest to a given query point, the distance between the query point and the other data points will need to be calculated. These distance metrics help to form decision boundaries, which partition query points into different regions.
The following distance measures can be used for this purpose [11]:
Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Manhattan distance :

$$d(x, y) = \sqrt{\sum_{i=1}^{m} (y_i - x_i)}$$

Minkowski distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} |y_i - x_i|^{\frac{1}{p}}}$$

Hamming distance

$$D_H = \sqrt{\sum_{i=1}^{k} |y_i - x_i|}$$

x=y D=0
x≠y D≠1

The k value in the k-NN algorithm defines how many neighbours will be checked to determine the classification of a specific query point. For example, if k = 1, the instance will be assigned to the same class as its single nearest neighbour. An appropriate value of k will help in the prevention of overfitting or underfitting. Lower values of k can have high variance but low bias, and larger values of k may lead to high bias and lower variance. The choice of k will largely depend on the input data, as data with more outliers or noise will likely perform better with higher values of k.

Decision Tree Classifier:

The intuition behind decision trees is to use dataset features to create yes/no questions and continually split the dataset until all data points belonging to each class are isolated. Therefore, in this process, data is organized in the form of a tree structure. Every time a question is asked, the dataset is split based on the value of a feature, and a node is added to the tree. The first node is called the root node, and the last node created when the split process is stopped is known as the leaf node. Every time a question is answered, new branches are created, segmenting the feature space into disjointed regions. One branch of the tree has all the data points corresponding to answering "yes" to the question. The other branch has the node with the remaining data points. This is because the feature space is narrowed down with each split or branch in the tree, and each data point will only belong to one region. The goal is to continue splitting the feature space and applying rules until there are no more rules to apply or no data points left.

The algorithm tries to completely separate the dataset such that all leaf nodes are assigned to a single class and leads are known as pure leaf nodes. But there are mixed lead nodes where not all data points have to be of the same class. In the case of mixed leaf nodes, the algorithms assign the most common class to all the data points in that node. A greedy approach is used to select the best tree, which picks the feature used in each split instead of trying to make the best overall decision. It focuses on the node at hand and what's best for the node in particular. On every split, the algorithm tries to divide the dataset into the smallest subset possible[13]. So, the final goal is to minimise the loss function as much as possible.

The loss function should evaluate a split based on the proportion of data points belonging to each class before and after the split. The Decision Tree uses loss functions like Gini impurity and entropy to evaluate the split based on the purity of the resulting nodes.

Gini Impurity is the measure of variance across the different classes. [12]

$$G(node) = \sum_{k=1}^{c} p_k(1 - p_k)$$

$$p_k = \ Probability\ of\ picking\ a\ data\ point\ from\ class\ k$$

$$1 - p_k = \ Probability\ of\ not\ picking\ a\ data\ point\ from\ class\ k$$

$$p_k = \frac{number\ of\ observations\ with\ class\ k}{all\ observations\ in\ node}$$

Entropy is a measure of chaos within the node. Chaos in the context of decision trees is having a node where all classes are equally present in the data.

$$Entropy(node) = -\sum_{k=1}^{c} p_k log(p_k)$$

$$p_k = \ Probability\ of\ picking\ a\ data\ point\ from\ class\ k$$

$$p_k = \frac{number\ of\ observations\ with\ class\ k}{all\ observations\ in\ node}$$

Using entropy as a loss function, a split is only performed if the entropy of each of the resulting nodes is lower than the entropy of the parent node. Otherwise, the split is not locally optimal.

Random forest Classifier:

The core unit of random forest classifiers is the decision tree, and a random forest is said to be an ensemble of many decision trees [14]. As already explained, the decision tree is a hierarchical structure that is built using the features (or the independent variables) of a data set. Each node of the decision tree is split according to a measure associated with a subset of the features. The random forest is a collection of decision trees that are associated with a set of bootstrap samples that are generated from the original data set. The nodes are split based on the entropy (or Gini index) of a selected subset of the features. The subsets that are created from the original data set, using bootstrapping, are of the same size as the original data set. In the standard random forest approach, bootstrapping is randomly selecting samples from training data with replacement. They are called "bootstrap samples."

The bootstrapping technique helps the development of a random forest with a set of the required number of decision trees in order to improve classification accuracy through the concept of overlap thinning [15]. Then an approach called the bagging (bootstrap aggregate) technique, in which decision trees are used as parallel estimators, is used to select the best trees with a voting scheme. The result is based on the majority vote of the results received from each decision tree. Random forests reduce the risk of overfitting, and accuracy is much higher than with a single decision tree. Furthermore, decision trees in a random forest run in parallel so that time does not become a bottleneck.

The success of a random forest highly depends on using uncorrelated decision trees. If the trees are very similar, the overall result will not be much different than the result of a single decision tree. Random forests achieve uncorrelated decision trees by bootstrapping and feature randomness. Feature randomness is achieved by selecting features randomly for each decision tree in a random forest. The number of features used for each tree in a random forest can be controlled with the max_features parameter.

Gradient boost algorithm:

The gradient boosting algorithm is an ensemble model. An ensemble model is based upon the concept of ensemble learning, in which a strong model is built by using a collection of weaker models. AdaBoost was the first boosting algorithm. AdaBoost and related algorithms were the first cast in a statistical framework by [16]. The term "gradient" in gradient boosting refers to the fact that you have two or more derivatives of the same function. Gradient Boosting is an iterative functional gradient algorithm that minimizes a loss function by iteratively choosing a function that points toward the negative gradient.

Given a training dataset, $D = \{x_i, y_i\}_1^N$, the goal of gradient boosting is to find an approximation, F(x), of the function F*(x), which maps instances x to their output values y, by minimizing the expected value of given loss function L(y, F(x)). Gradient boosting builds an additive approximation of F*(x) as a weighted sum of functions.

$F_m(x) = F_{m-1}(x) + \rho_m h_m(x)$

Where ρ_m is the weight of the m^{th} function, $h_m(x)$. These functions are the models of the ensemble. The approximation is constructed iteratively. First, a constant approximation of F*(x) is obtained as

$$F_0(x) = arg\,min_a \sum_{i=1}^{N} L(y_i, \alpha)$$

Subsequent models are expected to minimize

$$(\rho_m, h_m(x)) = arg\,min_{\rho,h} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \rho h(x_i))$$

Instead of solving optimization problems directly, each h_m can be seen as a greedy step in a gradient descent optimization for F*. For that, each model, h_m, is trained on a new dataset , $D = \{x_i, r_{mi}\}_{i=1}^{N}$ where pseudo-residuals, r_{mi}, are calculated by

$$r_{mi} = \left[\frac{\partial L(y_i, F(x))}{\partial F(x)}\right]_{F(x)=F_{m-1}(x)}$$

The value of ρ_m is subsequently computed by solving a line search optimization problem.

Feature Engineering:

It involves exploring the options to improve the model's performance using a feature extraction approach to data preparation. This involves applying suitable data preparation techniques to raw data and then aggregating all features together to create one large dataset on which the model is fitted and evaluated.

Since the data is numeric in nature, the data preparation technique involves transforms such as MinMaxScalar, StandardScalar, and Robust Scalar, which are used to scale the input variable. In addition to this, the Qantile transformer and KBinsDiscretizer are used for chaining the distribution of input variables. Finally, the PCA and TruncatedSVD transform are used to remove the linear dependencies between the input variables. A further recursive feature elimination technique is used to select the most relevant features out of all the features extracted.

4. Result and Analysis

Firstly, the performance of all the algorithms defined here is obtained on the PD data corresponding to five numbers of defects. The computation is performed on the Google Colab platform on an inter-core i-5 2.71 GHz processor. Two parameters, namely percentage accuracy in classifying the data pattern and execution time, are taken into consideration for performance analysis. The results corresponding to all the algorithms are provided in table 4.1.

After this process, data preparation is performed that treats data transforms as an approach to extract salient features from raw data. After applying the respective transforms to the 10 input variables' data, a total of 64 features were extracted.

A Further Recursive Feature Eland execution time. It was obtained that random forest algorithms have the highest recognition rate with 99.974%, but the execution time is comparatively large. Whereas Linear Discriminate Analysis has the lowest computation time of 3.83 seconds but a comparatively low pattern recognition accuracy percentage. On average, the decision tree classifier shows better performance in both these aspects.

Sr. No.	Name of Algorithm	Accuracy (percentage)	Execution time (seconds)	Accuracy after Feature engineering (percentage)	Execution time (seconds)
1.	Random Forrest Classifier	99.9744	166.99	99.9986	301.85
2	Gradient Boost Algorithm	99.9710	2042.67	99.9865	13463.79
3	Decision Tree Classifier	99.9565	7.11	99.9686	51.92
4	Linear Discriminant Analysis	92.0899	3.82	96.0547	41.79
5	K-nearest neighbour	79.5082	9.88	79.7820	213.64

Table 4.1

5. Conclusion

Five algorithms were evaluated on the PD data corresponding to five types of defects on the basis of Patter recognition accuracy percentage and execution time. It was obtained that random forest algorithms have the highest recognition rate with 99.974%, but the execution time is comparatively large. Whereas Linear Discriminate Analysis has the lowest computation time of 3.83 seconds but a comparatively low pattern recognition accuracy percentage. On average, the decision tree classifier shows better performance in both these aspects.

Feature engineering was used in order to improve performance. In this, 64 features were extracted using the most suitable transforms, out of which the 15 most relevant ones were selected using a recursive feature elimination technique. It resulted in an improvement in the pattern recognition accuracy percentage of all the respective algorithms with a maximum of 99.9986 for the Random Forest classifier, but simultaneously the computation time also increased for all the algorithms. Therefore, feature engineering is recommended where accuracy is of prime importance over execution time.

6. References:

6.1. IEC60270. High-Voltage test techniques-PD measurements;2000.

6.2. Wani, Shufali Ashraf, et al. “Advances in DGA Based Condition Monitoring of Transformers: A Review.” *Renewable and Sustainable Energy Reviews*, vol. 149, 2021, p. 111347., https://doi.org/10.1016/j.rser.2021.111347.

6.3. “Raw Data Examples to Analyse with PDFlex.” *Signal Processing Tool*, 19 Jan. 2022, http://pdflex.ewi.tudelft.nl/examples/.

6.4. Rodrigo Mor, A., et al. “A New Design of a Test Platform for Testing Multiple Partial Discharge Sources.” *International Journal of Electrical Power & Energy Systems*, vol. 94, 2018, pp. 374–384., https://doi.org/10.1016/j.ijepes.2017.07.013.

6.5. Mor, A. Rodrigo, et al. “New Clustering Techniques Based on Current Peak Value, Charge and Energy Calculations for Separation of Partial Discharge Sources. "*IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 24, no. 1, 2017, pp. 340–348., https://doi.org/10.1109/tdei.2016.006352.

6.6. Mor, A. Rodrigo, et al. “New Clustering Techniques Based on Current Peak Value, Charge, and Energy Calculations for Separation of Partial Discharge Sources.” *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 24, no. 1, 2017, pp. 340–348., https://doi.org/10.1109/tdei.2016.006352.

6.7. Abdul Madhar, Saliha, et al. “Physical Interpretation of the Floating Electrode Defect Patterns under AC and DC Stress Conditions.” *International Journal of Electrical Power & Energy Systems*, vol. 118, 2020, p. 105733., https://doi.org/10.1016/j.ijepes.2019.105733.

6.8. Sun, Jixing, et al. “Partial Discharge Characteristics of Free Moving Metal Particles in Gas Insulated Systems under DC and AC Voltages.” *IET Generation, Transmission & Distribution*, 2022, https://doi.org/10.1049/gtd2.12501.

6.9. “Degradation of Solid Dielectrics Due to Internal Partial Discharge: Some Thoughts on Progress Made and Where to Go Now.” *IEEE Transactions on Dielectrics and Electrical Insulation*, 2005, pp. 1275–1275., https://doi.org/10.1109/tdei.2005.1561811.

6.10. Hudon, C., and M. Belec. “Partial Discharge Signal Interpretation for Generator Diagnostics.” *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 12, no. 2, 2005, pp. 297–319., https://doi.org/10.1109/tdei.2005.1430399.

6.11. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. (2013). An introduction to statistical learning : with applications in R. New York : Springer

6.12. P. Tan, M. Steinbach, and V. Kumar. (2005) Introduction to Data Mining. Addison Wesley.

6.13. Suthaharan, S. “A Cognitive Random Forest.” *Handbook of Statistics*, 2016, pp. 207–227., https://doi.org/10.1016/bs.host.2016.07.006.

6.14. Suthaharan, Shan. “Machine Learning Models and Algorithms for Big Data Classification.” *Integrated Series in Information Systems*, 2016, https://doi.org/10.1007/978-1-4899-7641-3.

6.15. Authors Leo Breiman. “Arcing the Edge.” *UCB Statistics*, 1 June 1997, https://statistics.berkeley.edu/tech-reports/486.

6.16. Singh Tomar, Geetam; Chaudhari, Narendra S.; Barbosa, Jorge Luis V.; Aghwariya, Mahesh Kumar (2020). [algorithms for intelligent systems] International Conference on Intelligent Computing and Smart Communication 2019 (Proceedings of ICSC 2019) doi:10.1007/978-981-15-0633-8.