Archives available at journals.mriindia.com

International Journal of Advanced Electrical and Electronics Engineering

ISSN: 2278-8948

Volume 12 Issue 01, 2023

A Novel Machine Learning Technique for PV Panel Series DC Arc Fault Detection

[1]Ezhil Vignesh K, [2]Anusha V, [3]Santheesh C, [4]Ajin S

[1,2,3,4]Department of Electrical and Electronics Engineering, Stella Mary's College of Engineering, Aruthenganvillai, Kanyakumari District, Tamilnadu 629202, India.

Peer Review Information

Submission: 06 Feb 2023
Revision: 10 March 2023
Acceptance: 08 April 2023

Keywords

Fast Fourier Transform, Short Fourier Transform, Wavelet Transform, Random Forest Classifier, Sail Fish Optimization.

Abstract

In our method, data is gathered from a variety of sources, such as smart meters, sensors, and meteorological information, and is then used to train a machine learning model. On the basis of the recent and previous data, the model is then used to forecast the probability of a problem happening. We run several experiments on a sizable dataset of power faults to show the efficacy of our methodology. Our findings demonstrate that our model has a high degree of precision and recall when it comes to fault detection. Overall, our research has the potential to increase the dependability and efficiency of electrical systems and marks a substantial advancement in the automation of electricity issue detection. The suggested strategy was put into action in three stages. Applying the discrete wavelet transform after reading the signals within a specific window size is necessary. With a five-level wavelet, we have used Daubechies wavelet 3 or 9 (db3, db9). Next, calculate each level's attributes, including energy, variance, wavelength, standard deviation, and entropy. Additionally, the sailfish optimization approach selects the top qualities before applying a random forest classifier to determine whether or not a fault is present. The fact that our suggested method provides for precision and early identification due to the right window size and characteristics is one of its main benefits. Our proposed method beats existing approaches like independent RF and ANN.

Introduction

One of the most important renewable energy technologies today, solar energy produces electricity that is both clean and effective. Normally, a long wire connects PV panels to inverters or DC micro grids. Wire connections, high voltage transitions, thermal expansion due to external factors, and unit ageing all lead to various failures.

An arc is a bright electrical discharge between two insulating materials that is typically followed by a partial volatilization of the electrodes. A harmful unintended parallel or series arc between conductors is referred to as an arc fault. This error affects the overall dependability and effectiveness of the system while also cutting maintenance expenses. Early detection is one of the key ideas in our research. Fault types include open faults, arc faults, line-to-line faults, and ground faults. Arc faults have a stochastic nature. As a result, grid systems have higher maintenance expenses and less dependability. It is therefore one of the most important issues with power systems. The three different types of arc faults are ground-arc faults, parallel arc faults, and finally series DC-arc faults.

A parallel arc occurs when insulation between two or more parallel lines breaks down due to an external heat force, whereas a series arc fault

is caused by the disconnection of a conductor in transmission power lines. Arc fault detection techniques are crucial because it is crucial to quickly identify arc faults for the safety of DC systems. In most cases, parallel arc faults with differing current flows and arc fault currents that rapidly increase act as extra impedances in the system and reduce arc fault current. As a result, standard protective mechanisms are disabled. Series arc faults have the potential to harm the power supply sources, the system controller, and potentially cause explosions if they are not quickly identified and corrected.

Arc fault detection has been the subject of various kinds of research. The conventional threshold approach is used to locate AC arc faults, however it is unsuitable for locating DC arc faults, especially DC series faults. Because noisy data can make a conventional system detect a problem. Similar to the previous example, the need for a rapid and effective solution is driven by the growing time complexity of resolving all of these problems as a result of the segment length of feature extraction and unwanted feature detection failures.

Using data from experiments, mathematical models of arc faults have been created. The models are appropriate for theoretical research and study but do not completely capture the exterior properties of the arc. Additionally, arc fault detection techniques have been developed based on the traits of arc faults, such as strong light, high temperature, significant distortion noise, and high electromagnetic radiation.

Proposed System Description

The suggested strategy was put into action in three stages. A five-level wavelet daubechies wavelet (db3, db9). For each level, figure out properties like energy, variance, wavelength, standard deviation, and entropy. Prior to employing a random forest classifier to determine whether or not a defect is present, the sailfish optimization approach selects the optimal attributes.

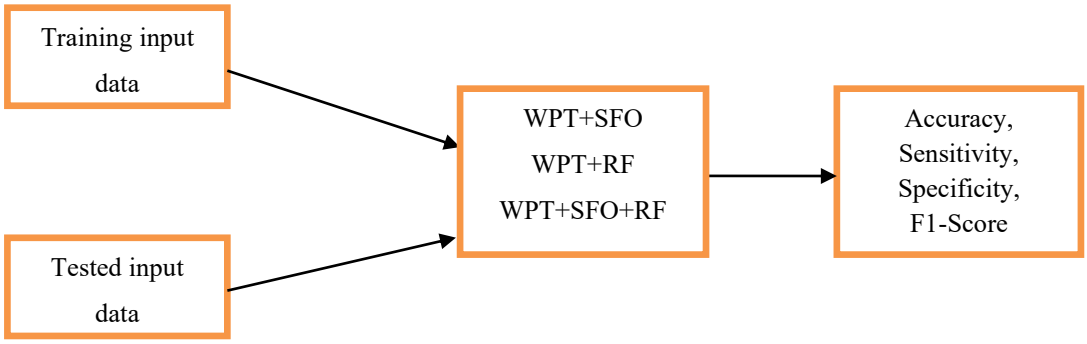

Figure 1: Block Diagram

The objective of this research is to rapidly and precisely identify DC-arc faults. In our proposed method, we first extracted input from single wavelet characteristics such as energy, variance, waveform length, and entropy, and then we applied selected features using the SFO algorithm. The data is then classified using the RF classifier.

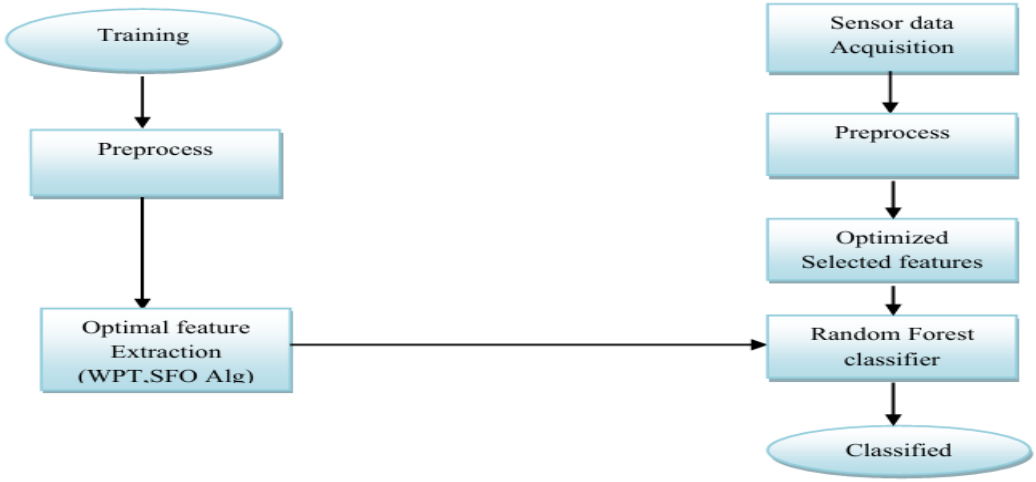

Figure 2: Flow Chart

Proposed System Modelling

1. Discrete Wavelet Transform

The DWT analyses the data at numerous frequency bands with different resolutions by splitting it into a rough approximation and comprehensive information. DWT uses two sets of functions: scaling functions and wavelet functions, which are coupled with low-pass and high-pass filters, respectively. Simply by applying consecutive high-pass and low-pass filters to the time domain data, the signal is divided into separate frequency bands. The original signal x [n] is subjected to half band high pass filter g [n] and low pass filter h [n]. After filtering, the signal's highest frequency is now p/2 radians rather than p, therefore half of the samples can be eliminated using Nyquist's criterion. The signal can be subsampled by two by removing all other samples. Where yhigh [k] and ylow [k] are the outputs of the high-pass and low-pass filters, respectively, after two subsampling, this is one level of decomposition. This breakdown reduces the temporal resolution in half because only half the samples are now needed to define the entire signal. The approach doubles the frequency resolution while halving the frequency uncertainty because the signal's frequency spectrum now only covers half of the previous frequency band. For more breakdown, the aforementioned technique also referred to as subband coding can be used repeatedly. With filtering and subsampling, the frequency resolution will be doubled while the number of samples used will be cut in half, giving the temporal resolution a 50% reduction. Figure 3 illustrates this technique with the original signal to be deconstructed (x [n]), a low-pass filter (h [n]), and a high-pass filter (g [n]), respectively. The letter "f" on the graphic stands for the signal's bandwidth at each level.

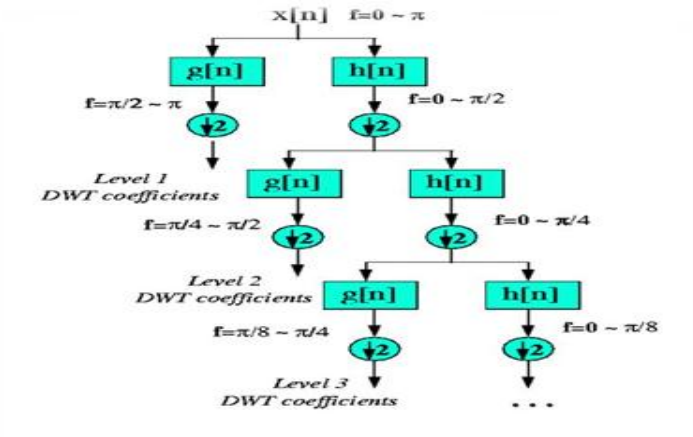

Figure 3: Subband Coding Algorithm

An one-dimensional real-valued vector x is transformed into a transformed vector w of the same length using the (one-dimensional) DWT. Figures 6(a) and (b) depict the DWT's first two stages for a 16-dimensional vector. First, a discrete-time low-pass filter (LPF) h of specified length (in the Diagrams, we use length four for demonstration purposes) filters the vector x at two-second intervals.

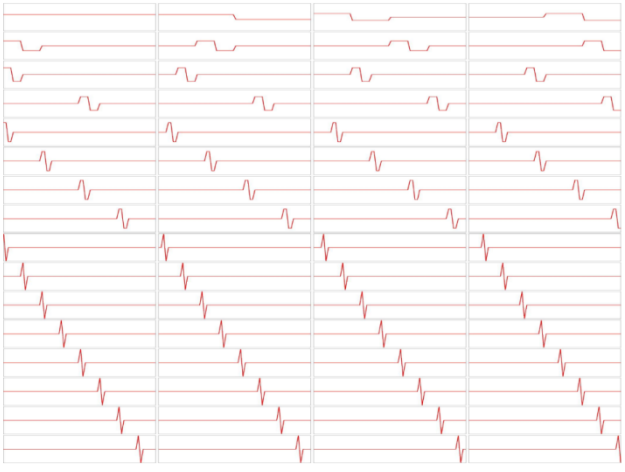

Figure 4: Haar wavelet basis functions (length 64)

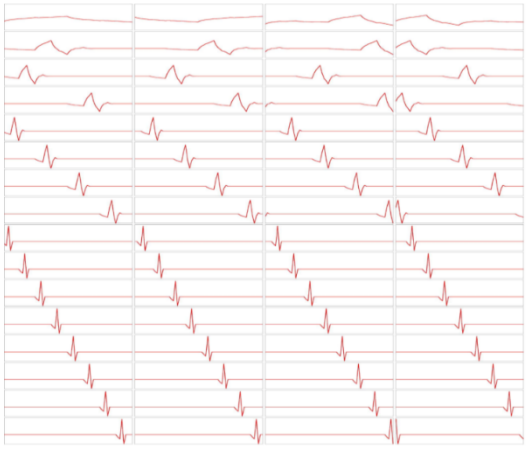

Figure 5: Haar Wavelet Basis Functions (length 64)

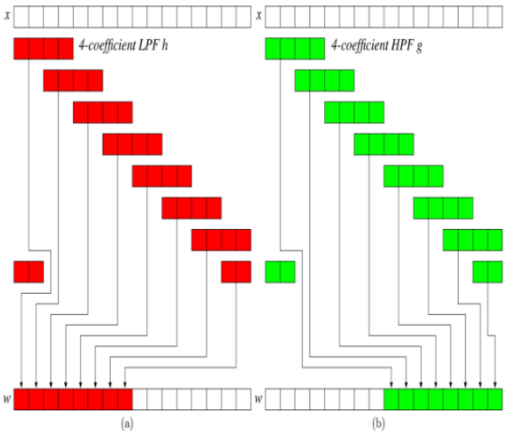

Figure 6: For a signal of length 16, the first step of the DWT

After being low-pass filtered in two-step increments, the coefficients of the original signal are clustered as the first eight elements of the vector. In the second stage of the DWT, the original signal is high-pass filtered in steps of two, and the resulting coefficients are aggregated to form the last eight elements of the vector.

To hold values, the first eight elements of w are used. This process is shown in Figure 6 (a). Second, a discrete-time high-pass filter (HPF) of specified length (again, we assume a filter of length four for example reasons) is applied to the vector x at intervals of two, and the resulting high-pass values are stored in the final eight elements of w. This process (b) is shown in Figure 6.

Note how the vector x is qualitatively altered by this method. The low-pass part of the vector w merely represents a two-fold down sampled version of the original signal x, but the high-pass part of the vector w locates and detects high frequencies in x. If we stopped here, the vector w would represent a one-level wavelet transform of x. We can continue; using the same method described above and shown in Figure 6, the low-pass filtered portion of w (in this case, the first eight components) may be further adjusted. Figure 7 depicts a three-level, one-dimensional DWT as an illustration.

In the final transform w3, values L3 are the outcome of three consecutive low-pass filters, values H3 are the outcome of two consecutive low-pass filtering operations followed by a high-pass filter, values H2 are the outcome of a low-pass filter followed by a high-pass filter, and values H1 are the outcome of one high-pass filter. The three values H1, H2, and H3, respectively, will be formed by separating and concentrating the highest frequencies. It's important to keep in mind that at lower frequencies, the resolution for each level of the wavelet transform is cut in half.

Since lower frequencies cannot be localized with the same precision as higher frequencies, the DWT operation is aware of this (see Heisenberg Uncertainty Principle). In conclusion, a multi-resolution frequency decomposition is used to breakdown and localize a one-dimensional discrete-time signal.

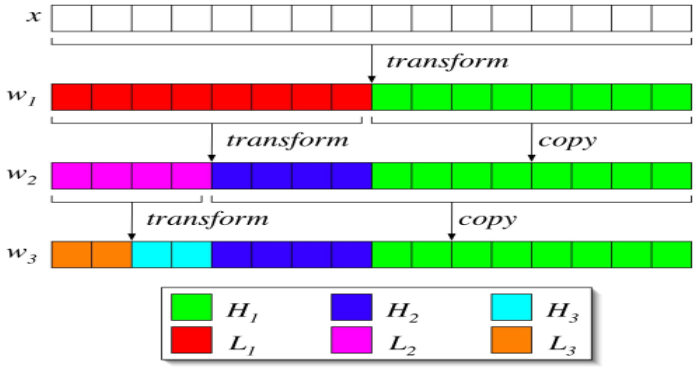

Figure 7: Signal x has a length of 16 and a three-level wavelet transform

H1 coefficients stay unchanged from w1 to w2, whereas H1 and H2 coefficients remain unchanged from w2 to w3.

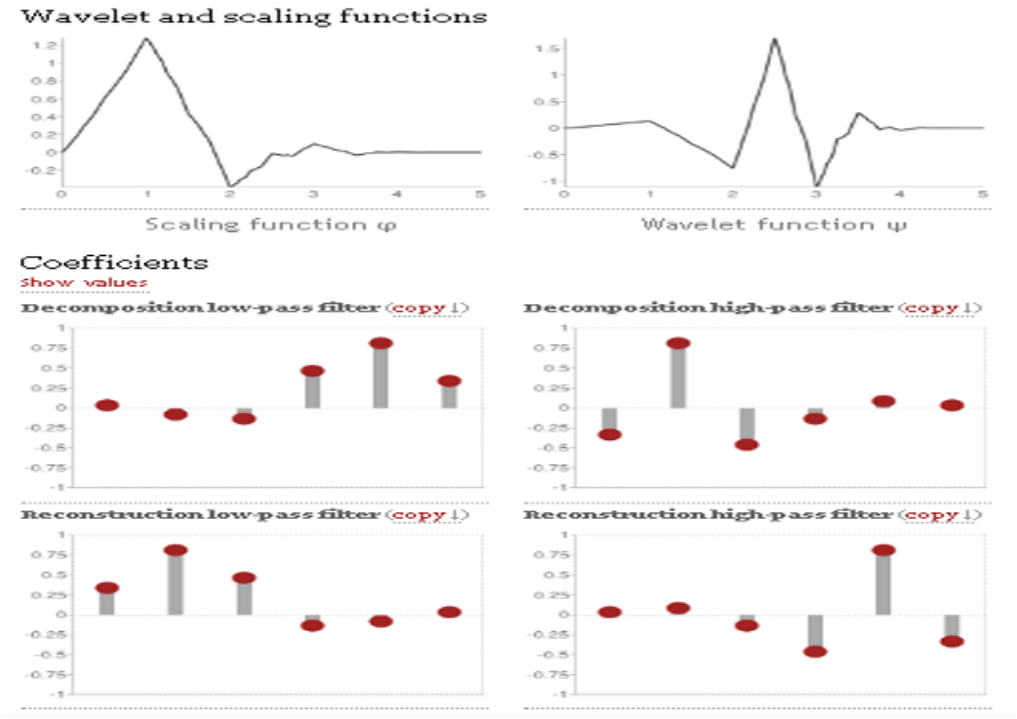

Figure 8: Db3 coefficient

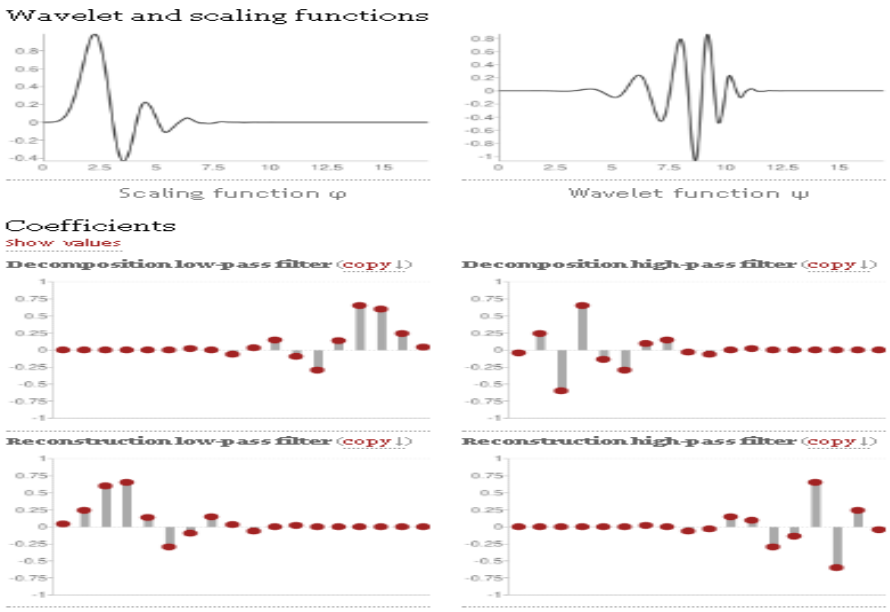

Figure 9: Db9 Coefficient

2. Feature Extraction

With the type of db3, features are retrieved from wavelet trees; here, five layers of decomposition were applied. There are a total of 6 features computed for each window channel, including Energy, Variance, Standard Deviation, Waveform Length, and Entropy. We therefore have about 6 X 5 = 30 characteristics for each segment. Each equation listed below.

Wavelet decomposition values for each level are X0, X1,..., X5. Wavelet decomposition values are arrays that make up each X value. Each equation in the feature calculation is presented below.

K= 0, 1, 2 ….. M level

Energy

$$E^{(i)}(t) = \sum_{\tau=t_0}^{i} \left(f^{(i)}(\tau)\right)^2 \quad \text{……(1)}$$

Where E(i)(t) represents the energy of level i. After completing the calculation of energy of decomposition levels for all waveforms, a depiction of the energy of each level for each waveform follows.

Waveform Length

$f(t) = \sum_{i=1}^{i=j} D_i(t) + A_j(t)$ …… (2)

Where D (t) $_i$ denotes the wavelet detail and A (t) $_j$ stands for the wavelet approximation at the j th level, respectively.

Variance

$V(\alpha) = \frac{1}{n}\sum_{j=1}^{n} W^2(a, x_j)$ …… (3)

Where W^2 (a, x_j) is the squared wavelet coefficient associated with scale a at data point x_j , and n is the number of data points.

Standard Deviation

$Std_k = \sqrt{\sigma_k^2}$ …… (4)

Entropy

$H_i = -\sum_{i=1}^{n} p(i)\, e^{1-p(i)} * \sum_{i=1}^{n} p(i)\, log_2 p(i)$ …… (5)

Where P(i) probability of total energy.

3. SFO Algorithm

A population-based metaheuristic algorithm is the SFO. The sailfish are assumed to be potential solutions in this technique, and the variables in the problem represent the sailfish's locations inside the search space. As a result, the population in the solution space is produced at random. With their changeable location vectors, sailfish can hunt in one, two, three, or hyper dimensional space. The *i*th member at the *k*th searching bout has a current location *SFi, R (i = 1, 2,..., m)* in a d-dimensional search space. The idea of saving the location of every sailfish has been considered by Matrix SF. As a result, during optimization, these spots display all solutions' variables.

$$SF_{position} = \begin{bmatrix} SF_{1,1} & SF_{1,2} & \cdots & SF_{1,d} \\ SF_{2,1} & SF_{2,2} & \cdots & SF_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ SF_{m,1} & SF_{m,2} & \cdots & SF_{m,d} \end{bmatrix} \quad (1)$$

where *f* calculates the fitness function and SF fitness saves the fitness value, which gives the value of fitness or objective function for each sailfish, and m is the number of sailfish, *SFi,j* is the value of the *j*th dimension of the *i*th sailfish, and *f* calculates the fitness function. The first row of the SF position matrix is supplied to the fitness function, and the output shows the sailfish's fitness value in the SF fitness matrix for that row. Another important factor included in the SFO algorithm is the school of sardines. It is assumed that the sardine school is also present in the search area. Therefore, the following uses the position of sardines and their fitness values:

$$S_{position} = \begin{bmatrix} S_{1,1} & S_{1,2} & \cdots & S_{1,d} \\ S_{2,1} & S_{2,2} & \cdots & S_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ S_{n,1} & S_{n,2} & \cdots & S_{n,d} \end{bmatrix} \quad (4)$$

where *n* is the number of sardines and *SF i,j* is the value of the *j*th dimension *of i*th sardine, *Sposition* matrix indicates the position of all sardines.

$$S_{Fitness} = \begin{bmatrix} f(S_{1,1}\, S_{1,2} \cdots S_{1,d}) \\ f(S_{2,1}\, S_{2,2} \cdots S_{2,d}) \\ \vdots \\ f(S_{n,1}\, S_{n,2} \cdots S_{n,d}) \end{bmatrix} = \begin{bmatrix} F_{S_1} \\ F_{S_2} \\ \vdots \\ F_{S_n} \end{bmatrix} \quad (5)$$

where *n* is the total number of sardines, *SFi,j* displays the value of the *i*th sardine's *j*th dimension, *f* is the objective function, and SF itness stores the fitness value of each sardine. It is noteworthy that sardines and sailfish are equivalent factors in determining the answers. Sailfish are the primary factor that is dispersed around the search space in this algorithm, and sardines can collaborate to discover the best position in this region 22 S. Shadravan, H.R. Naji, and V.K. Bardsiri Engineering Applications of Artificial Intelligence 80 (2019) 20–34. Sardine can actually be defeated by sailfish when searching the search space, and sailfish changes its position if a better solution is discovered than the one already found.

4. RF Classifier

Basic RF concepts and steps involved in using random forests for classification or regression:

- Collect and prepare the training data: For the task at hand, this involves collecting a sizable number of data points and labels, and then dividing the data into a training set and a validation set.
- Determine the appropriate number of trees: The random forest's tree count is a hyper parameter that can be adjusted to enhance the model's performance. In general, more trees will result in a better model, but at the expense of longer computation times.
- Pick the features you'll use: Random forests frequently perform best when a lot of features are used. However, caution must be exercised when employing highly linked or duplicated features since this might result in over fitting.
- Create the trees: To create each tree in the random forest, a random sample of

the training data is chosen and replaced (sampling with replacement, sometimes referred to as bootstrapping). Next, the tree is built by selecting the best split based on a random subset of the features at each node.

- Make predictions: The model averages the forecasts of all the trees in the forest to produce a prediction on a new data point. The final prediction for classification is the class that received the most votes. The final prediction in regression is determined using the mean of all the tree predictions.
- Evaluate the model: The performance using a range of measures, such as accuracy (for classification) or mean squared error (for regression). The model can then be improved by modifying the hyper parameters or adding more data.

Working of Random Forest Algorithm:

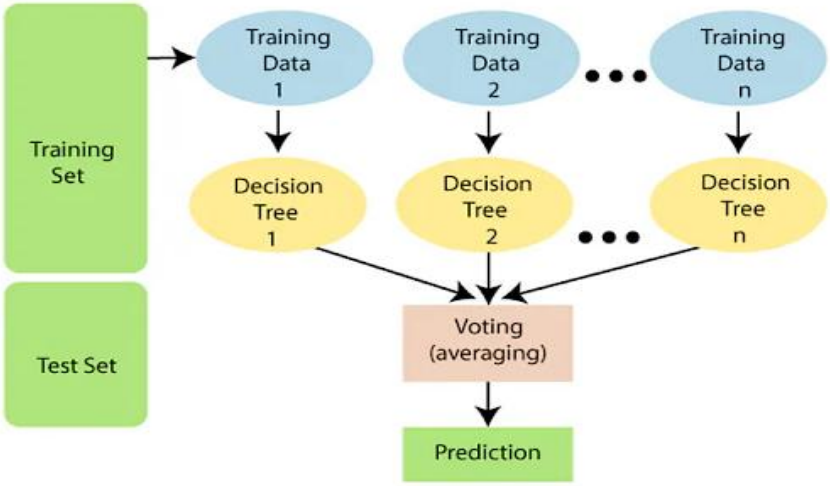

Figure 10: Random Forest Algorithm

Steps Explain the Working of Random Forest Algorithm

Step 1: Select random samples from a given data or training set.

Step 2: This algorithm will construct a decision tree for every training data.

Step 3: Voting will take place by averaging the decision tree.

Step 4: Finally, select the most voted prediction result as the final prediction result.

This combination of multiple models is called Ensemble. Ensemble uses two methods:

Bagging:

- Bagging is the process of generating an alternative training subset via replacement from a sample training dataset. The outcome is decided by a majority vote.
- The bootstrap aggregating, or bagging, generalization technique is applied to tree learners via the random forest training algorithm. Bagging repeatedly (B times) chooses a random sample with replacement of the training set and fits trees to these samples given a training set X = x1,..., xn and responses Y = y1,..., yn:

For $b = 1, ..., B$:

- Sample n training examples from X and Y with replacement; label these Xb and Yb.
- Train the fb classifier or regression tree using the Xb and Yb data.
- Following training, predictions for x' can be made by averaging predictions from all the distinct regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

• The standard deviation of the predictions from each separate regression tree on x' can also be used to measure the prediction's level of uncertainty:

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

Boosting:

Boosting is the process of turning weak learners into strong learners by developing successive models with the goal of achieving the maximum accuracy possible. Like ADA BOOST and XG BOOST.

Features of Random Forest

- Miscellaneous: Every tree has a different kind, trait, and feature

compared to other trees. Different types of trees exist.

- Immune to the dimensionality curse: A tree is a conceptual idea, hence it doesn't need to take features into account. As a result, the feature space is smaller.
- Parallelization: Since each tree is independently constructed from various data and features, we may fully utilize the CPU when constructing random forests.
- Train-Test split: Because the decision tree never sees 30% of the data in a Random Forest, we don't need to separate the data for train and test.
- Stability: The outcome is based on bagging, which means it is based on the average or majority vote.
- Random Forest Algorithm

1. For b = 1 to B:

(a) Draw a bootstrap sample Z∗ of size N from the training data.

(b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

i. Select m variables at random from the p variables.

ii. Pick the best variable/split-point among the m.

iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree.

Then $\hat{C}_{rf}^B$ (x) = majority vote $\{\hat{C}_b(x)\}_1^B$.

When grown deeply enough, data structures have a relatively low bias. Trees gain a lot from the averaging because they are known to be loud. The expectation of an average of B such trees is also the same as the expectation of any one of them because each tree produced by bagging is identically distributed (i.d.). Accordingly, the bias of bagged trees is identical to that of the individual trees, and the only way to improve is by reducing the variation. Contrastingly, in boosting, the trees are grown in an adaptive manner to eliminate bias and are not hence i.d.

An average of B i.d. random variables, each with variance σ^2, has variance $\frac{1}{b}\sigma^2$. If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pair wise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

The benefits of averaging are constrained by the size of the correlation between pairs of bagged trees since when B increases, the second term vanishes but the first one stays. With random forests, the goal is to increase variance reduction while keeping the correlation between the trees as low as possible. This is accomplished during the tree-growing process by selecting the input variables at random. In particular, choose m ≤ p input variables at random as candidates for splitting before each split while constructing a tree on a bootstrapped dataset.

Result

In this paper, a machine learning method for detecting DC arc faults in PV panel series was developed. Wavelet parameters of the input signal, such as energy, variance, waveform length, and entropy, have been retrieved. Selected features have then been applied utilising the optimization procedure in our suggested technique. The data is then classified using an RF classifier.

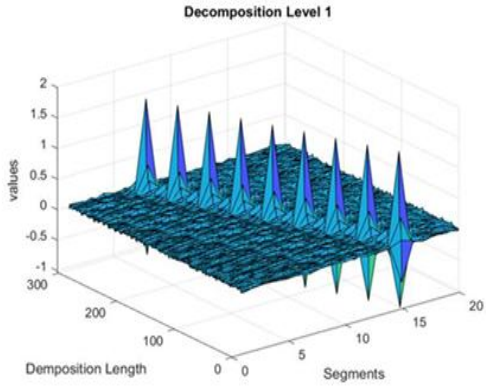

Figure 11: Decomposition Level 1

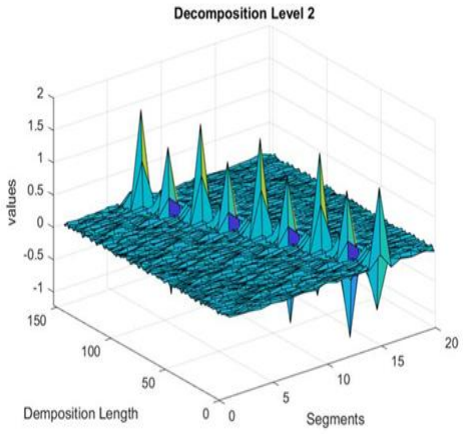

Figure 12: Decomposition Level 2

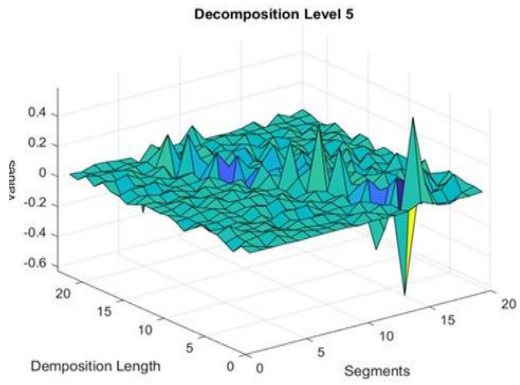

Figure 13: Decomposition level 4

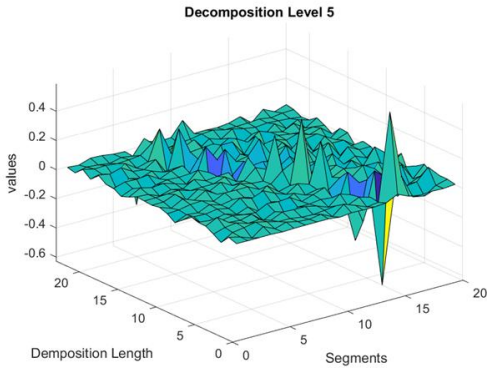

Figure 14: Decomposition level 5

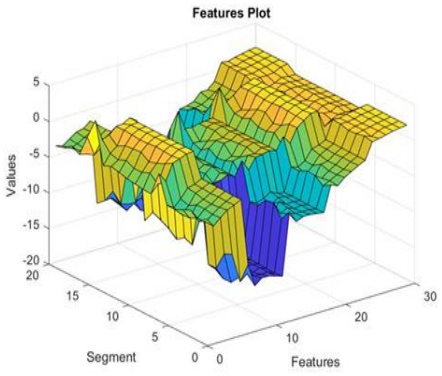

Figure 15: Selected Features

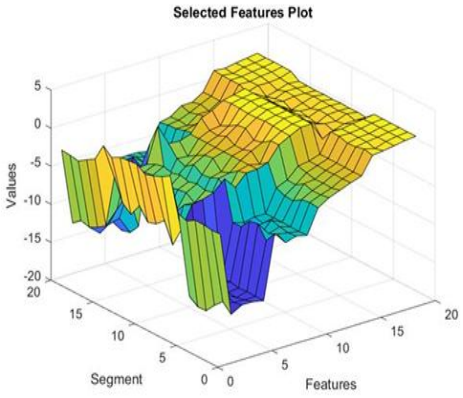

Figure 16: Arrangement of Selected Features

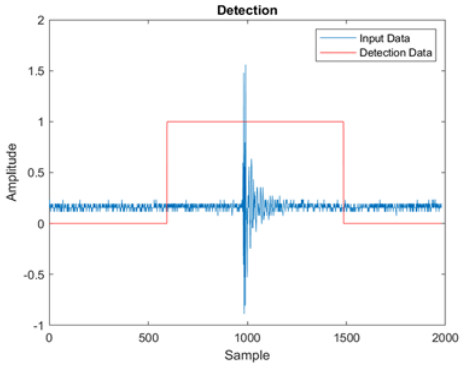

Figure 17: Classified output

A binary classifier's performance can be seen using a confusion matrix, as shown in Table 4. The diagonal shows how many cases the classifier correctly predicted. True negatives (TN) correspond to accurately discovered flaws, whereas true positives (TP) refer to successfully recognized normal. The classifier misclassified two different sorts of instances: false positives (FP), which represent faults mistakenly classified as normal, and false negatives (FN), which represent normal mistakenly classified as faults.

A confusion matrix (S1 and S2) is used to help a binary classifier distinguish between two classes.

Table 1. Confusion matrix

True class	Predicted class	
	S1 (Normal)	S2 (Fault)
S1 (Normal)	TP	FN
S2 (Fault)	FP	TN

The proportion of instances correctly classified by a classifier is used to assess accuracy:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (13)$$

This may not be the greatest performance metric if the class distribution of the dataset is unbalanced. A classifier with a high accuracy rate might identify all cases as belonging to class C1 if class C1 is much larger than class C2. Sensitivity is defined as the proportion of true positives (TP), and specificity is defined as the proportion of true negatives (TN). The definitions of "sensitivity" and "specificity" are as follows:

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (14)$$

$$\text{Sensitivity} = \frac{TN}{TN+FP}$$

Error = 1-accuracy

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{Specificity} = \frac{TN}{TN+FP}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

False positive rate = $\frac{FP}{FP+TN}$ F_1 score = $\frac{2TP}{2TP+FP+FN}$

Matthews correlation coefficient MCC = $\frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

Results

Table 2. Fault Detection Result of SFORF

SFO+RF	Predict_normal	Predict_fault
Actual_normal	1264	20
Actual_fault	76	590

Table 3. Simulation Output Values of SFORF

Metrics	**Values**
Accuracy	0.9508
Error	0.0492
Sensitivity	0.9844
Specificity	0.8859
Precision	0.9433
False Positive Rate	0.1141
F1_score	0.9634
Matthews Correlation Coefficient	0.8902
Kappa	0.8883

Table 4. Fault Detection Result of NN

NN	**Predict_normal**	**Predict_fault**
Actual_normal	1715	1
Actual_Fault	598	301

Table 5. Simulation Output Values of NN

Metrics	**Values**
Accuracy	0.7709
Error	0.2291
Sensitivity	0.9994
Specificity	0.3348
Precision	0.7415
False Positive Rate	0.6652
F1_score	0.8513
Matthews Correlation Coefficient	0.4967
Kappa	0.3970

Table 6. Fault Detection Result of SVM

SVM	**Predict_normal**	**Predict_fault**
Actual_normal	1708	8
Actual_fault	266	633

Table 7. Simulation Output Values of SVM

Metrics	**Values**
Accuracy	0.8952
Error	0.1048
Sensitivity	0.9953
Specificity	0.7041
Precision	0.8652
False Positive Rate	0.2959
F1_score	0.9257
Matthews Correlation Coefficient	0.7723
Kappa	0.7507

Table 8. Simulation Output Values of SFORF, NN & SVM

Metrics	SFO+RF	NN	SVM
Accuracy	0.9508	0.7709	0.8952
Error	0.0492	0.2291	0.1048
Sensitivity	0.9844	0.9994	0.9953
Specificity	0.8859	0.3348	0.7041
Precision	0.9433	0.7415	0.8652
False Positive Rate	0.1141	0.6652	0.2959
F1_score	0.9634	0.8513	0.9257
Matthews Correlation Coefficient	0.8902	0.4967	0.7723
Kappa	0.8883	0.3970	0.7507

Conclusion

The above project model is used to predict the likelihood of a fault occurring, based on the current and historical data. We run several experiments on a sizable dataset of power faults to show the efficacy of our methodology. Our findings demonstrate that our model has a high degree of precision and recall when it comes to fault detection. Our work has the potential to increase the dependability and effectiveness of electrical systems and marks a substantial advancement in the automation of electricity fault detection. Using Matlab and Arduino systems, we designed and put into action a DC arc defect detection system. Comparing our SFO-based RF Classifier to other machine learning techniques, it performed better.

References

Grid Kostyantyn Koziy, Bei Gou, Member, IEEE, and Joel Aslakson, "A Low-Cost Power-Quality Meter With Series Arc-Fault Detection Capability for Smart Grid", IEEE TRANSACTIONS ON POWER DELIVERY, VOL. 28, NO. 3.

W. J. Lee, M. Sahni, C. Kwan, Z. Ren, J. Sheeley "A Novel Approach for Arcing Fault Detection for Medium/Low-Voltage Switchgear"

Carlos E. Restrepo "Arc Fault Detection and Discrimination Methods" Siemens Energy and Automation, Residential Product Division, 5400 Triangle Parkway,Norcross, GA 30092

Hoang-Long Dang, Jaechang Kim, Sangshin Kwak, Member IEEE, and Seungdeog Choi, Senior Member IEEE "Series DC Arc Fault Detection Using Machine learning Algorithms"

Andoni Lazkano, Jesus Ruiz, Elisabete Aramendi and Luis A. Leturiondo "Evaluation of a new proposal for an arcing fault detection method based on wavelet packet analysis", EUROPEAN TRANSACTIONS ON ELECTRICAL POWER Euro. Trans. Electr. Power ; 14:161–174 (DOI: 10.1002/etep.13)

S. A. Saleh, A. S. Aljankawey, R. Errouissi, and E. Castillo-Guerra "Extracting the Phase of Fault Currents: A New Approach for Identifying Arc Flash Faults", Department of Electrical and Computer Engineering, University of New Brunswick Fredericton, NB, Canada, asaleh@unb.ca.

John A. Kay, G. Amjad Hussain, Matti Lehtonen, Lauri Kumpulainen"New Pre-Emptive Arc Fault Detection Techniques In Medium Voltage Switchgear And Motor Controls".

S. A. Saleh, Senior, Member, IEEE, A. S. Aljankawey, Member IEEE, R. Errouissi, Member, IEEE,and M. A. Rahman, Life Fellow, IEEE, "Phase-Based Digital Protection for Arc Flash Faults" DOI 10.1109/TIA.2016.2515991, IEEE Transactions on Industry Applications.

Yang Yi, Yu Qiongfang, Dong Aihua, Yu Qiongfang, Zheng Dezhong, "Study on the Fault Arc On-line Detecting Based on Wavelet Singularity Detection Principle" 2009 WASE International Conference on Information Engineering, yuqf@hpu.edu.cn.

Nikola L. Georgijevic, Marko V.Jankovic Senior Member, IEEE, Srdjan Srdic, Member, IEEE and Zoran Radakovic, "The Detection of series Arc-Fault in Photovoltaic Systems Based on the Arc Current Entropy", DOI 10.1109/TPEL.2015.2489759, IEEE Transactions on Power Electronics.