



**SRI VENKATESWARA INTERNSHIP PROGRAM  
FOR RESEARCH IN ACADEMICS  
(SRI-VIPRA)**



**SRI-VIPRA**


**Project Report of 2024: SVP-2402**

**Artificial Intelligence and Machine learning- Driven Real Time  
Power Quality Fault Detection System**


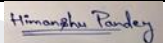









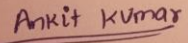

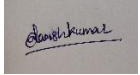
**IQAC  
Sri Venkateswara College  
University of Delhi  
Benito Juarez Road, Dhaula Kuan, New Delhi  
New Delhi -110021**

**SRIVIPRA PROJECT 2024**

**Title : Artificial Intelligence and Machine learning- Driven Real Time Power Quality Fault Detection System**

<p><b>Name of Mentor: Dr. Rahul</b>  <b>Name of Department: Electronics</b>  <b>Designation: Assistant Professor</b></p>	
--	--

*List of students under the SRIVIPRA Project:*

S. No.	Photo	Name of the students	Roll number	Course	Signature
1		Himanshu Pandey	1622047	BSc. (H) Electronics	
2		Devansh Sawhney	1623058	BSc. (H) Electronics	
3		Mohd. Hermain Irfan	1623044	B.Sc(Hons) Electronics	
4		Aarav Mazumdar	1623041	B.Sc(H) Electronics	
5		Bhanu Pratap Singh	1123014	B.Sc(P) Life Science	
6		Ankit Kumar	1523074	B. Sc(H) Chemistry	
7		Adarsh Kumar	0923040	B.com(P)	



**Signature of Mentor**

## Certificate of Originality

This is to certify that the aforementioned students from Sri Venkateswara College, University of Delhi have participated in the summer project SVP-2024 titled **“Artificial Intelligence and Machine learning- Driven Real Time Power Quality Fault Detection System ”**.The participants have carried out the research project work under my guidance and supervision from 1<sup>st</sup> July, 2024 to 30<sup>th</sup> September 2024. The work carried out is original and carried out in an online/offline/hybrid mode.



**Signature of Mentor**

## **Acknowledgements**

Firstly, we are grateful to God Almighty for providing us with the knowledge, perseverance, and strength essential to finish the job successfully.

We would like to express our sincere gratitude to Prof. Vajala Ravi, the esteemed Principal of Sri Venkateswara College, for organizing the internship program, providing us with this wonderful opportunity, and for her constant support and encouragement during the entire research process. Their dedication to creating a climate of academic achievement has been crucial to this project's success.

We also like to sincerely thank Prof Vartika Mathur, the coordinator, IQAC of SRIVIPRA, and the Convenors of SRIVIPRA, for their unwavering assistance during the project. Their commitment to advancing research efforts has been indispensable. Their involvement in the project made it possible for the authors to effectively carry out the goals of their research.

We appreciate the 'Department of Electronics' encouragement for broadening our academic boundaries. The quality of this research has been greatly improved by their mentorship, insightful challenges, and commitment to academic achievement.

We owe an abundance of thankfulness to our mentor, Dr. Rahul, Assistant Professor, Department of Electronics, for his unwavering support and direction; it was made possible by his aid and encouragement. Over the course of four weeks, we worked with and learned about various significant Python libraries, as well as gained insight into important algorithms and the concepts of machine learning and artificial intelligence. For all of us, it was a wonderful learning opportunity.

Finally, we would like to express our gratitude to everyone who has supported and been a part of our initiative. We have immense gratitude and appreciation for all of your assistance and support.

## **TABLE OF CONTENTS**

<b>S.No</b>	<b>Topic</b>	<b>Page No.</b>
<b>1.</b>	<b>Preface</b>	<b>5</b>
<b>2.</b>	<b>Introduction</b>	<b>6</b>
<b>3.</b>	<b>AI/ML Concepts</b>	<b>8</b>
<b>4.</b>	<b>Libraries and Modules</b>	<b>10</b>
<b>5.</b>	<b>Feature Extraction Algorithms</b>	<b>11</b>
<b>6.</b>	<b>Optimisation Techniques</b>	<b>22</b>
<b>7.</b>	<b>Conclusion</b>	<b>26</b>

## **Preface**

The reliability of the power provided to our homes, businesses, and industries has become essential in an era identified by a never-ending pursuit of technological innovation and an infinite demand for electricity. We need the power we get to be not just dependable but also uphold high standards of quality. In a world where electrically operated gadgets and systems are used more and more, it is crucial now more than ever to understand and deal with complex power quality challenges.

This research, titled ‘AI/ML based analysis of power signal disturbances,’ explores the convergence of two transformative domains: artificial intelligence/machine learning and power quality analysis. We start with a study of how artificial intelligence (AI) and machine learning (ML) can be used to enhance the way we evaluate, track, and enhance the quality of electric energy throughout these pages. The growing need for energy efficiency, the emergence of renewable energy sources, and the complexity of power systems are the primary factors behind this research. AI/ML provide a viable way to address the problems head-on.

We explore the fundamentals of the AI/ML ideas used to perform the analysis, the several metrics and parameters that affect power quality, and the consequences of low power quality on our power systems in this report. After that, we started investigating the field of AI/ML, looking at how these tools might be used to evaluate and predict power quality issues with previously unheard-of speed and precision. We emphasize the advantages and practical uses of AI/ML-based power quality analytics along the way.

We anticipate that this study will be a useful tool for scholars, engineers, decision-makers, and stakeholders as we share the conclusions and revelations from our in-depth investigation. We hope we can bring in a new era of energy management that is not just dependable and efficient but also environmentally sustainable by combining power quality analytics with AI/ML.

In conclusion, we would like to thank all the people and institutions that made contributions to this research. We believe that the information on these pages will

stimulate more creativity, teamwork, and advancement in the field of AI/ML power quality analysis.

## Introduction

In today's world, electricity plays a vital role in our everyday lives, powering everything from residential and commercial spaces to transportation networks and industrial operations. As our reliance on electricity grows, so does the need for maintaining high-quality power. Power quality refers to the consistent and reliable delivery of electricity, free from disruptions, harmonics, or other irregularities that may negatively affect the performance of electrical equipment and systems.

The significance of power quality is evident in its broad impacts. Poor power quality can result in equipment malfunctions, higher energy consumption, shorter lifespans of electrical devices, and potential safety hazards. In addition, as renewable energy sources, electric vehicles, and smart grids become more integrated into our energy systems, maintaining optimal power quality becomes even more crucial for the stability and sustainability of the overall energy infrastructure.

Historically, power quality analysis has depended on manual techniques, which are often slow, labor-intensive, and limited in their ability to address the complexities of modern power systems. However, with the rise of artificial intelligence (AI) and machine learning (ML), new possibilities have emerged to enhance how we assess and manage power quality.

This research report delves into the integration of AI/ML with power quality analytics, with the goal of developing and applying AI/ML techniques to improve power quality assessments. Ultimately, this research aims to introduce the Power Quality Index (PQI), a comprehensive metric that evaluates various dimensions of energy quality, providing a more precise and effective assessment than traditional approaches.

Our research objectives include the following key goals:

- **Enhanced Accuracy:** By utilizing AI/ML algorithms, we aim to increase the precision of power quality assessments, enabling better detection and characterization of electrical disturbances and anomalies.

- **Real-Time Monitoring:** We are investigating the potential of AI/ML to facilitate real-time monitoring of power quality, which is essential for timely interventions and maintenance.
- **Data-Driven Insights:** By analyzing large datasets, we aim to reveal hidden patterns and correlations within power quality data, offering valuable insights to grid operators, utility companies, and energy researchers.
- **Automation and Predictive Maintenance:** Through predictive analytics, we intend to develop models capable of forecasting power quality issues before they arise, supporting proactive maintenance strategies and minimizing system downtime.
- **Energy efficiency:** Enhancing power quality not only guarantees the dependability of power systems but also promotes greater energy efficiency and sustainability, aligning with global environmental objectives.

As we embark on the path of merging AI/ML with power quality analytics, we anticipate that the findings and advancements outlined in this report will help create a more adaptable, efficient, and sustainable energy infrastructure. By integrating AI/ML technologies, we can deepen our comprehension of energy quality while tackling the evolving challenges and opportunities in the swiftly changing energy sector.



## **Concepts of Artificial Intelligence and Machine Learning**

Artificial Intelligence (AI) and Machine Learning (ML) are two transformative technologies reshaping industries across the globe. While AI focuses on creating machines that can simulate human intelligence, ML is a subset of AI that enables systems to learn from data, improve their performance over time, and make decisions with minimal human intervention. Understanding the core principles and applications of these technologies is essential for leveraging their full potential in a variety of fields.

### **The Foundation of AI and ML:**

AI is a broad field that encompasses various subdomains like natural language processing, computer vision, and robotics. Its primary goal is to develop machines that can perform tasks typically requiring human intelligence, such as reasoning, problem-solving, and language comprehension. ML, on the other hand, focuses on developing algorithms that allow computers to learn patterns from data and make predictions or decisions without being explicitly programmed.

### **Supervised and Unsupervised Learning:**

In ML, there are two primary types of learning methods: supervised and unsupervised. Supervised learning involves training a model on a labeled dataset, where the input-output pairs are known. The model learns to make predictions by mapping input features to the correct outputs. Common applications include image classification, speech recognition, and fraud detection.

In unsupervised learning, the data is not labeled, and the algorithm must find hidden patterns or relationships in the dataset. This type of learning is often used for clustering, anomaly detection, and recommendation systems. As AI/ML technologies advance, more sophisticated models like reinforcement learning and deep learning are also gaining traction.

### **Applications of AI and ML:**

AI and ML have wide-ranging applications across sectors such as healthcare, finance, energy, and more. In healthcare, AI-powered tools can analyze medical images, assist in diagnosis, and predict patient outcomes. In finance, ML models are used for fraud detection, risk assessment, and algorithmic trading. Energy sectors use AI/ML to optimize power grids, forecast energy consumption, and improve energy efficiency.

### **The Future of AI and ML:**

As AI and ML continue to evolve, they hold the promise of driving unprecedented innovation. Autonomous vehicles, smart cities, and personalized healthcare solutions are just a few examples of future applications. However, the rapid growth of these technologies also raises ethical concerns, such as bias in algorithms, data privacy, and job displacement. Addressing these challenges will be crucial as AI/ML becomes increasingly integrated into everyday life.

In summary, AI and ML are powerful tools that are not only enhancing current technologies but are also creating new opportunities. With the right approach, these concepts can be harnessed to improve efficiency, solve complex problems, and build a more intelligent and connected world.

## Libraries, Modules and Functions

### ➤ Feature Extraction Algorithms in Machine Learning:

Feature extraction is a critical step in machine learning and data analysis algorithms. By converting raw data into a more interpretable and informative form, it significantly enhances the efficiency, precision, and usability of these algorithms.

Essentially, feature extraction acts as a bridge between raw data and effective machine learning techniques. It allows data to be transformed into a more valuable representation, improving the overall performance and applicability of these algorithms when addressing various real-world problems.

Some of the feature extraction techniques we utilize include:

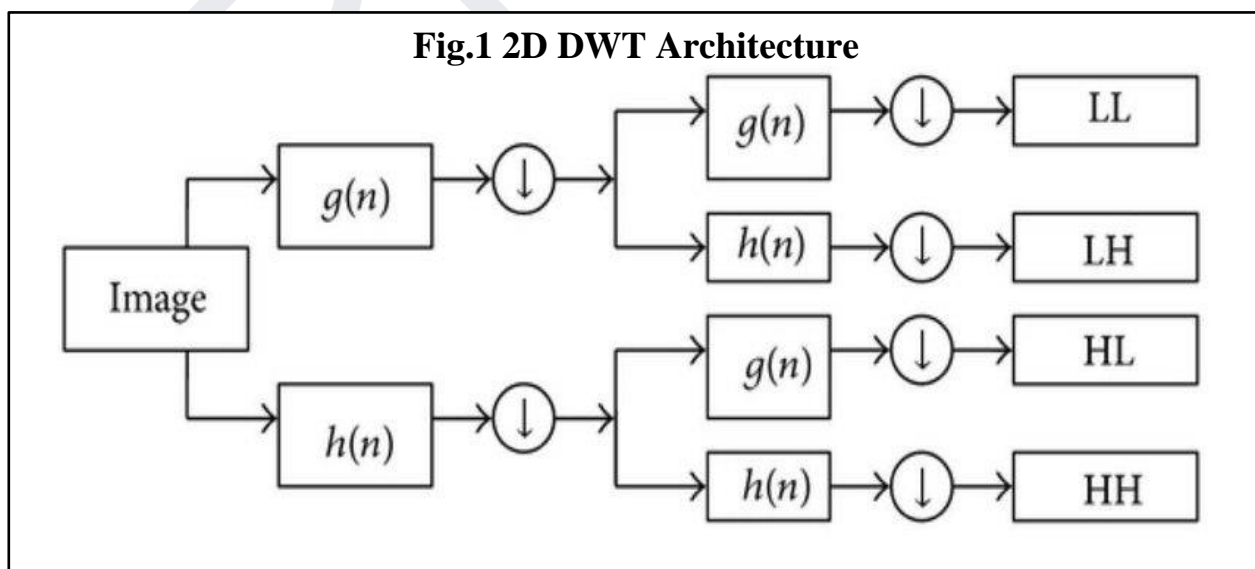
- a. **Discrete Wavelet Transform (DWT)** - It's a versatile method used to decompose data into different frequency bands, making it possible to analyze signals at multiple levels of detail. This allows us to capture both the timing and frequency characteristics, making it well-suited for applications like signal processing and image compression.
- b. **Principal Component Analysis (PCA)** - A technique used to reduce the complexity of a dataset by identifying the most influential features, allowing us to retain essential patterns and relationships while simplifying the overall structure. It's often used to make large datasets easier to understand and visualize.

## ❖ Discrete Wavelet Transform (DWT) :

The Discrete Wavelet Transform (DWT) is a widely-used signal processing technique that decomposes a signal into different frequency components while preserving temporal information. Unlike traditional Fourier transforms, which only provide frequency information, DWT allows for analysis across both time and frequency domains. This makes it particularly useful for signals with non-stationary characteristics, where frequencies change over time.

DWT operates by applying wavelet functions, small oscillatory waveforms, to the input signal at various scales and positions. The signal is decomposed into a set of approximations (low-frequency components) and details (high-frequency components). The approximation represents the overall trend of the signal, while the detail captures abrupt changes or high-frequency information. This multi-resolution analysis makes DWT effective in applications such as image compression, denoising, and feature extraction.

One of the main advantages of DWT is its ability to localize both time and frequency aspects of a signal, which is especially important in real-time systems and pattern recognition tasks. By selecting appropriate wavelet functions, DWT can be tailored to suit specific types of data, providing an efficient method for analyzing complex signals across a wide range of domains, including biomedical engineering, audio processing, and fault detection.



## Program 01 using PQDATASET:

Step 1: Import Required Libraries

+ Code + Markdown

```
# Cell 1: Importing Libraries
import numpy as np
import pandas as pd
import pywt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

[16]

Step 2: Load the Dataset

```
# Cell 2: Load the Dataset
dataset_path = r'D:\NUMPYPROJ\myenv\pqdataset.csv' # Adjust the path as necessary
data = pd.read_csv(dataset_path)

# Display the first few rows of the dataset to understand its structure
print(data.head())
```

✓ 0.4s

```
Unnamed: 0    Col1    Col2    Col3    Col4 \
0      0 -797.962914 -1320.199586 -1828.741445 -2319.935857
1      1  4387.885674  4567.747164  4693.741356  4776.824895
2      2 -4050.074299 -4340.408587 -4576.875995 -4769.518913
3      3  3458.450134  3686.700705  4000.773209  4392.451258
4      4  -910.262100 -312.245917  298.552078  896.568469

    Col5    Col6    Col7    Col8    Col9 ... \
0 -2789.218649 -3231.111864 -3641.962908 -4018.119193 -4355.929647 ...
1  4824.300958  4850.777519  4862.646175  4867.211521  4866.298624 ...
2 -4918.338175 -5025.159834 -5100.026323 -5141.111052 -5153.892968 ...
3  4834.343975  5269.845213  5636.871457  5876.078192  5950.944421 ...
4  1462.629392  1978.474763  2439.540162  2847.652177  3212.852886 ...

    Col120    Col121    Col122    Col123    Col124 \
0  3632.834002  3221.982426  2780.090228  2311.720161  1821.438861
1 -165.252287  524.062959  1199.684611  1842.437384  2437.714882
2  563.322919 -47.475988 -656.447463 -1252.637741 -1825.089177
3 -722.183210  67.562203  850.005032  1561.232918  2152.858410
```

4 -4506.574476 -4196.154858 -3881.168803 -3560.705563 -3225.633496

	Col125	Col126	Col127	Col128	output
0	1311.983509	790.659456	262.031836	-269.335205	1
1	2972.733259	3437.450839	3828.215741	4143.201102	2
2	-2364.673580	-2861.346591	-3309.630078	-3706.785897	2
3	2601.141643	2912.474970	3123.378574	3286.805684	3
4	-2860.433401	-2452.321914	-1991.255895	-1474.497200	3

[5 rows x 130 columns]

Step 3: Data Preprocessing

```
# Cell 3: Data Preprocessing
# Separate features and target variable
X = data.iloc[:, :-1].values # Features: all columns except the last
y = data.iloc[:, -1].values # Target variable: the last column

# Split the dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the feature data to have a mean of 0 and variance of 1
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

✓ 0.1s

step 4: Applying DWT

```
# Cell 4: Apply DWT
def apply_dwt(signal):
    """
    Apply Discrete Wavelet Transform (DWT) to a 1D signal.

    Parameters:
    | signal (array-like): Input 1D signal to transform.

    Returns:
    | tuple: Approximation coefficients and detail coefficients.
    """
    if len(signal.shape) == 1:
        coeffs = pywt.dwt(signal, 'haar') # Using Haar wavelet; can be changed if necessary
        return coeffs
    else:
        raise ValueError("Input signal must be 1D.")

# Apply DWT to each sample in the training and testing data
X_train_dwt = np.array([apply_dwt(sample)[0] for sample in X_train_scaled]) # Extract approximation coefficients
X_test_dwt = np.array([apply_dwt(sample)[0] for sample in X_test_scaled]) # Extract approximation coefficients
```

Confusion Matrix:

```
[[377  4  8  0 12]
 [ 3 399  0  0  3]
 [ 4  0 599  0  1]
 [ 0  0  0 412  0]
 [11  0  1  0 566]]
```

Classification Report:

	precision	recall	f1-score	support
1	0.95	0.94	0.95	401
2	0.99	0.99	0.99	405
3	0.99	0.99	0.99	604
4	1.00	1.00	1.00	412
5	0.97	0.98	0.98	578

Model Accuracy                    0.98    2400

macro avg    0.98    0.98    0.98    2400  
weighted avg    0.98    0.98    0.98    2400

#### Step 6: Plotting Results

```
# Cell 6: Plotting Results
# Plotting the confusion matrix using a heatmap
plt.figure(figsize=(10, 7))
sns.heatmap(confusion_mat, annot=True, fmt='d', cmap='Blues',
            xticklabels=np.unique(y), yticklabels=np.unique(y))
plt.ylabel('Actual Labels')
plt.xlabel('Predicted Labels')
plt.title('Confusion Matrix')
plt.show()

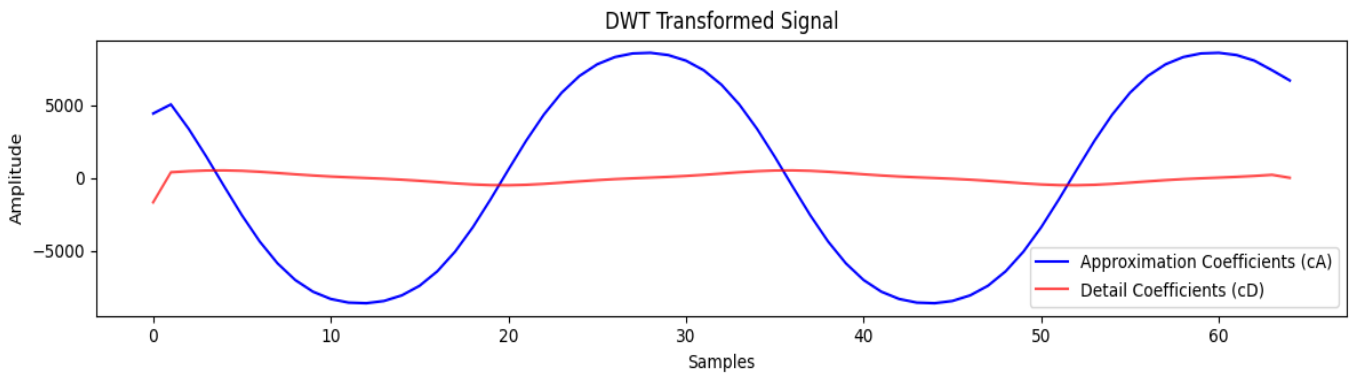
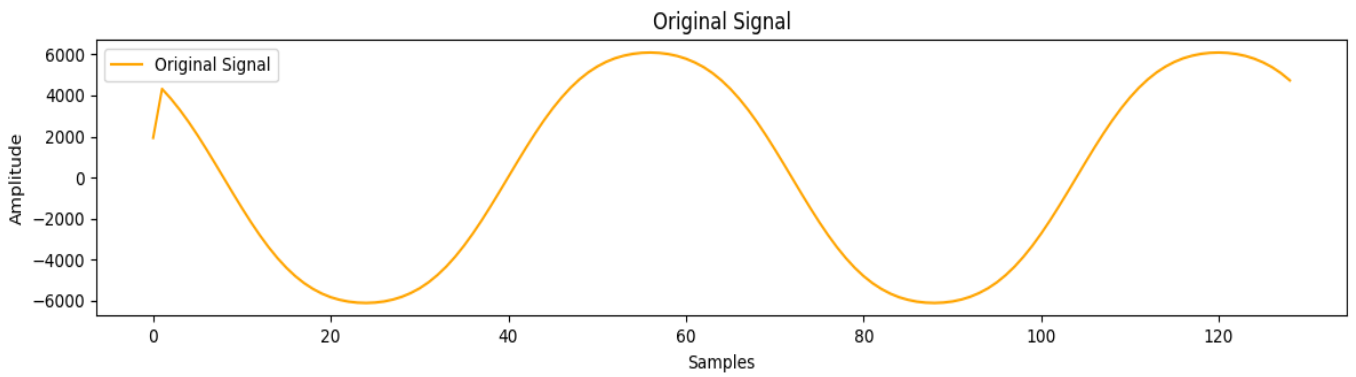
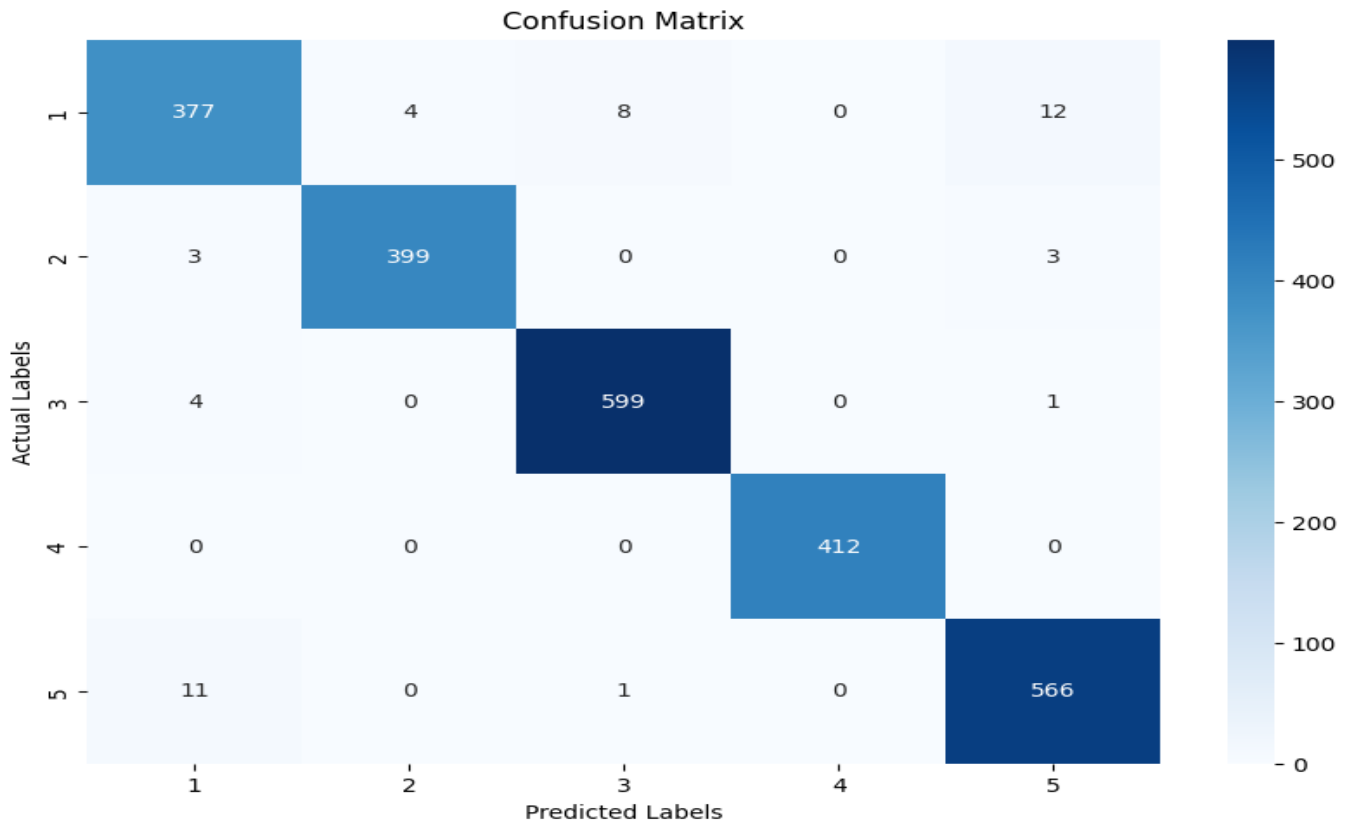
# Plotting the original signal and its DWT transformed representation
plt.figure(figsize=(12, 6))

# Original signal plot
plt.subplot(2, 1, 1)
plt.plot(X_test[0], label='Original Signal', color='orange')
plt.title('Original Signal')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()

# DWT transformed signal plot
plt.subplot(2, 1, 2)
approx_coefs, detail_coefs = apply_dwt(X_test[0]) # Obtain DWT coefficients
plt.plot(approx_coefs, label='Approximation Coefficients (cA)', color='blue')
plt.plot(detail_coefs, label='Detail Coefficients (cD)', alpha=0.7, color='red') # Include detail coefficients
plt.title('DWT Transformed Signal')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
plt.legend()

plt.tight_layout() # Adjust the layout for better visibility
plt.show()
```

✓ 0.9s





## Summary -

- **Multi-resolution Analysis:** DWT provides a multi-scale analysis of signals by decomposing them into varying frequencies.
- **Time-Frequency Localization:** It captures both time and frequency domain features of the input signal.
- **Signal Decomposition:** The signal is divided into approximation (low-frequency) and detail (high-frequency) components.
- **Efficient Feature Extraction:** Approximation coefficients represent essential features, while detail coefficients capture rapid changes.
- **Noise Reduction:** DWT helps remove noise by focusing on significant low-frequency components.
- **Dimensionality Reduction:** Transforms high-dimensional signals into compact, meaningful representations for easier processing.
- **Versatility:** DWT is adaptable to various data types, including time-series and image data.

## ❖ Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique that simplifies complex datasets by transforming them into a new set of variables called principal components. Unlike traditional methods that analyze data directly, PCA identifies the directions (components) in which the data varies the most and projects it onto these new axes, capturing the most significant patterns and trends. This helps in reducing the number of features while preserving the overall structure and variability of the data.

PCA operates by calculating the covariance matrix of the data and determining its eigenvectors and eigenvalues. The eigenvectors represent the principal components, and the corresponding eigenvalues indicate the amount of variance captured by each component. Typically, only a few principal components are sufficient to capture most of the variability, making PCA effective in reducing data dimensionality while retaining crucial information. This makes it ideal for visualization, noise reduction, and improving computational efficiency in tasks like machine learning and data analysis.

One of the key advantages of PCA is its ability to highlight underlying patterns and relationships in the data, making it easier to interpret and analyze. PCA is widely used in fields such as image processing, finance, bioinformatics, and more, where managing high-dimensional data is a challenge. By focusing on the most important features, PCA simplifies complex datasets, making it a fundamental tool in data-driven research and applications.

### Principal Component Analysis Steps

01

Perform standardization on the initial set of continuous variables

02

Find out the covariance matrix

03

Calculate the eigenvectors and eigenvalues of the covariance matrix to arrive at the PCs

04

Figure out which PCs to retain

05

Replot the data on your original axes

## Program 02 using PQDATASET:

Improved Power Quality Falut Detection using PCA

Importing necessary Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn import svm
```

[37] ✓ 0.0s

Python

loading and reading the real time dataset

```
real_time_dataset=pd.read_csv('pqdataset.csv')

# Printing few values to check the dataset
print("Dataset Head:\n", real_time_dataset.head())

# Checking the number of rows and columns in the Dataset
print("Dataset Shape:", real_time_dataset.shape)

# Checking for any missing values to improve accuracy
print("Number of Missing Values:\n", real_time_dataset.isnull().sum())
```

[2] ✓ 0.2s

Python

```
... Dataset Head:
   Unnamed: 0      Col1      Col2      Col3      Col4  \
0           0 -797.962914 -1320.199586 -1828.741445 -2319.935857
1           1  4387.885674  4567.747164  4693.741356  4776.824895
2           2 -4050.074299 -4340.408587 -4576.875995 -4769.518913
3           3  3458.450134  3686.700705  4000.773209  4392.451258
4           4  -910.262100  -312.245917   298.552078   896.568469

      Col15      Col16      Col17      Col18      Col19  ...  \
0 -2789.218649 -3231.111864 -3641.962908 -4018.119193 -4355.929647  ...
1  4824.300958  4850.777519  4862.646175  4867.211521  4866.298624  ...
2 -4918.338175 -5025.159834 -5100.026323 -5141.111052 -5153.892968  ...
3  4834.343975  5269.845213  5636.871457  5876.078192  5950.944421  ...
4  1462.629392  1978.474763  2439.540162  2847.652177  3212.852886  ...

      Col120      Col121      Col122      Col123      Col124  \
0  3632.834002  3221.982426  2780.090228  2311.720161  1821.438861
1  -165.252287   524.062959  1199.684611  1842.437384  2437.714882
2   563.322919  -47.475988   -656.447463 -1252.637741 -1825.089177
3  -722.183210   67.562203   850.005032  1561.232918  2152.858410
4 -4506.574476 -4196.154858 -3881.168803 -3560.705563 -3225.633496
```

	Col125	Col126	Col127	Col128	output
0	1311.983509	790.659456	262.031836	-269.335205	1
1	2972.733259	3437.450839	3828.215741	4143.201102	2
...					
Col127	0				
Col128	0				
output	0				

Length: 130, dtype: int64

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings...](#)

## Processing the Data

```
# Splitting the Dataset in features (X) and target (Y)
X = real_time_dataset.iloc[:, :-1] # All columns except the last one as features
y = real_time_dataset.iloc[:, -1] # Last column as target, that is output

# Splitting Data into training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
'''
test_size has been set to 0.2 which means 80% of Data goes for training and remaining for testing
this has been done to improve the accuracy
'''
```

```
# Standardizing features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Python

## Applying PCA for reduction

```
pca = PCA(n_components=60)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
explained_variance = np.sum(pca.explained_variance_ratio_)
print("Explained Variance by 30 Components: " + str(round(explained_variance, 2)))
```

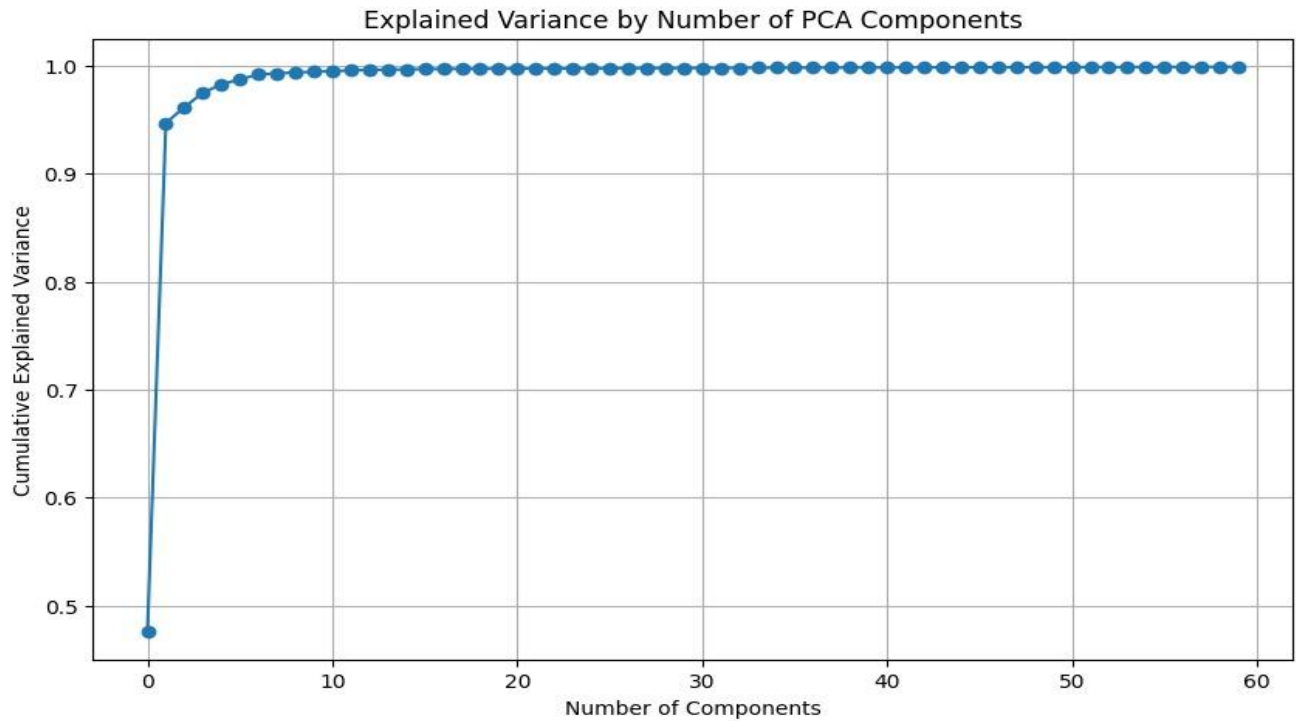
Python

... Explained Variance by 30 Components: 1.0

## Plotting the graph

```
explained_variance = pca.explained_variance_ratio_
plt.figure(figsize=(10, 6))
plt.plot(np.cumsum(explained_variance), marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Explained Variance by Number of PCA Components')
plt.grid()
plt.show()
```

Python



```

svm_model = svm.SVC()

# Hyperparameter tuning using GridSearchCV
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10, 50], 'gamma': ['scale', 'auto'], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(svm_model, param_grid, cv=5, scoring='accuracy', verbose=1)

# Fit the model
grid_search.fit(X_train_pca, y_train)

```

[ ] Python

... Fitting 5 folds for each of 16 candidates, totalling 80 fits

... **GridSearchCV**

- ▶ estimator: SVC
  - ▶ SVC

```

# Evaluate the Optimized Model
# Calculate the accuracy and print a detailed classification report.

# Predict on test data
y_pred = grid_search.best_estimator_.predict(X_test_pca)

# Model accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy with Optimized PCA: {:.2f}%".format(accuracy * 100))

# Detailed classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))

```

[ ] Python

... Model Accuracy with Optimized PCA: 99.96%

Classification Report:				
	precision	recall	f1-score	support
1	1.00	1.00	1.00	401
2	1.00	1.00	1.00	405
3	1.00	1.00	1.00	604
4	1.00	1.00	1.00	412
5	1.00	1.00	1.00	578
accuracy			1.00	2400
macro avg	1.00	1.00	1.00	2400
weighted avg	1.00	1.00	1.00	2400

## Summary :

- **Dimensionality Reduction:** PCA reduces the number of input variables while retaining essential data patterns.
- **Feature Extraction:** It transforms original variables into new uncorrelated variables called principal components.
- **Maximizes Variance:** Principal components capture the maximum variance in the data, prioritizing meaningful information.
- **Orthogonal Components:** Each principal component is orthogonal, ensuring they are independent of each other.
- **Eigenvectors and Eigenvalues:** PCA calculates eigenvectors and eigenvalues to determine directions and importance of components.
- **Variance Retention:** A few principal components often retain most of the original dataset's variance.
- **Useful for Noise Reduction:** By focusing on the most important components, PCA filters out less important data, reducing noise.

## ➤ Optimization Techniques Algorithms in Machine Learning:

Optimization Techniques in Machine Learning:

Optimization techniques play a vital role in enhancing the performance and efficiency of machine learning algorithms. They are employed to minimize or maximize an objective function, often related to the accuracy of a predictive model. By refining model parameters, optimization techniques help in achieving better generalization and convergence during the training process. These methods are essential for finding the optimal solution in complex datasets and improving the overall functionality of various algorithms.

Among the various optimization techniques utilized, one prominent method is Support Vector Machine (SVM). SVM is a powerful supervised learning algorithm used for classification and regression tasks. It works by identifying the hyperplane that best separates different classes in the feature space, maximizing the margin between the closest points of the classes (support vectors). This approach ensures that the model is robust and can generalize well to unseen data. Additionally, SVM can employ various kernel functions, such as linear, polynomial, or radial basis functions (RBF), to effectively handle non-linear relationships in the data. By optimizing the placement of the hyperplane, SVM not only enhances classification accuracy but also provides a clear geometric interpretation of the decision boundaries, making it a popular choice in various applications, including image recognition, text classification, and bioinformatics.

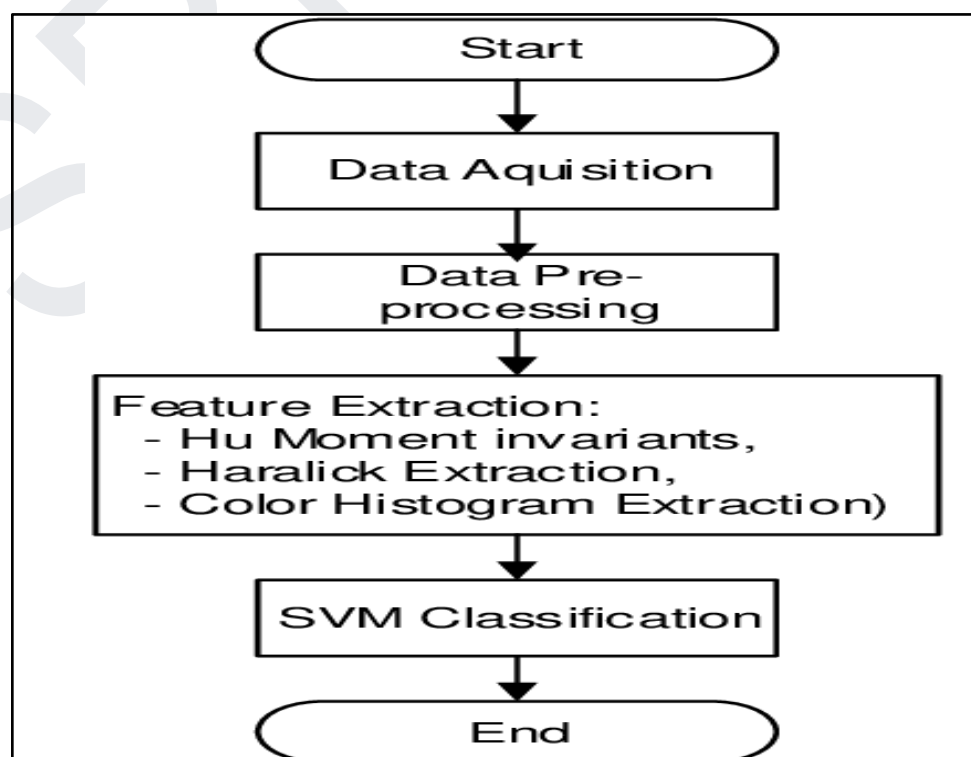


## Support Vector Machine(SVM):

Support Vector Machine (SVM) is a powerful supervised learning algorithm primarily used for classification tasks. It works by finding the optimal hyperplane that separates data points of different classes in a high-dimensional space, ensuring maximum separation between them. SVM is effective in handling complex datasets where the classes may not be linearly separable by using different kernel functions to transform the data into higher dimensions.

The core idea behind SVM is to create a decision boundary that maximizes the margin between the closest points (support vectors) of the classes, thus reducing the risk of misclassification. By selecting appropriate kernel functions (e.g., linear, polynomial, or radial basis function), SVM can model complex, non-linear relationships in the data. This flexibility allows SVM to perform well in a variety of scenarios, even with intricate datasets containing overlapping classes.

A key advantage of SVM is its robustness in handling high-dimensional data and its ability to generalize well to unseen samples. It is widely applied in image classification, text categorization, bioinformatics, and other areas where precise classification is crucial. Due to its mathematical rigor and effective handling of outliers, SVM remains a popular choice for both binary and multi-class classification problems.





## Program 03 using PQDATASET:

Importing necessary libraries

```
import pandas as pd
from sklearn import svm
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
```

[1]

Python

Loading and reading the dataset

```
real_time_data = pd.read_csv('pqdataset.csv')
```

[2]

Python

Splitting the Data into features and labels

```
X = real_time_data.iloc[:, :-1] # All columns except the last are features
y = real_time_data.iloc[:, -1] # The last column is the target label, that is output
```

[3]

Python

```
# Encode labels
from sklearn.preprocessing import LabelEncoder
if y.dtypes == 'object':
    le = LabelEncoder()
    y = le.fit_transform(y)
```

[4]

Python

Splitting the data into training and testing sets

```
# Split the data into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
'''
test_size has been set to 0.2 which means 80% of Data goes for training and remaining for testing
this has been done to improve the accuracy
'''
```

[10]

[10]

Python

```
... '\ntest_size has been set to 0.2 which means 80% of Data goes for training and remaining for testing\nthis has been done
```

Standardizing data to improve accuracy

```
# Standardize the feature values
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train) # Fit and transform training data
X_test = scaler.transform(X_test) # Transform test data
```

[6]

Python

using Hyperparameter to increase accuracy

```
# Hyperparameter tuning using GridSearchCV
param_grid = {'C': [0.1, 1, 10, 50, 100], 'gamma': [0.001, 0.01, 'scale'], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(svm.SVC(), param_grid, cv=5, scoring='accuracy', verbose=1)
grid_search.fit(X_train, y_train)
```

[7]

```
[7] # Using the best model found by GridSearchCV
    best_svm_model = grid_search.best_estimator_
Python

... Fitting 5 folds for each of 30 candidates, totalling 150 fits

Fitting and prediction on Dataset

# Fit the model and predict on test data
best_svm_model.fit(X_train, y_train)
y_pred = best_svm_model.predict(X_test)
[8] Python
```

```
Calculating the accuracy

accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy on Test Data: {:.2f}".format(accuracy * 100))
[9] Python

... Model Accuracy on Test Data: 99.21
```

## Summary :

- **Binary Classification Model:** SVM is primarily used for binary classification tasks by separating data into two classes.
- **Hyperplane Separation:** It identifies an optimal hyperplane that maximizes the margin between two classes.
- **Support Vectors:** Support vectors are the data points closest to the hyperplane, critical in defining it.
- **Maximizing Margin:** The primary goal of SVM is to maximize the margin between the support vectors.
- **Non-Linear Classification:** SVM handles non-linear data by using kernel functions to transform data into higher dimensions.
- **Kernel Functions:** Common kernels include linear, polynomial, and radial basis function (RBF), helping separate non-linear data.
- **Regularization Parameter (C):** This parameter controls the trade-off between maximizing the margin and minimizing classification errors.
- **Soft Margin SVM:** Allows misclassification in cases of non-separable data, improving generalization with noisy datasets.

## ❖ CONCLUSION:

The integration of artificial intelligence (AI) and machine learning (ML) into power quality analytics presents a transformative opportunity for the future of energy infrastructure. As global reliance on electricity continues to rise, maintaining a stable and consistent supply of high-quality electrical power becomes increasingly critical. AI/ML technologies offer the potential to revolutionize how we analyze, monitor, and manage power quality, introducing unprecedented levels of efficiency and accuracy.

This report has highlighted several key objectives in leveraging AI/ML for power quality analysis:

**Enhanced Accuracy:** AI and ML algorithms can significantly improve the precision of power quality assessments by detecting electrical irregularities and anomalies with unmatched accuracy. This ensures that even the most subtle issues are identified, leading to more reliable diagnostics and resolutions.

**Real-Time Monitoring:** The ability of AI/ML systems to enable real-time monitoring is crucial for timely interventions. By continuously tracking power quality in real-time, these systems allow operators to anticipate and address issues before they evolve into larger, more costly problems, thereby improving the overall reliability of the power grid.

**Data-Driven Insights:** Advanced AI/ML models can analyze vast amounts of power quality data, revealing hidden patterns, correlations, and trends. These insights provide invaluable guidance for utilities, operators, and researchers, enabling them to make more informed decisions regarding maintenance, upgrades, and operational strategies.

Looking ahead, we are optimistic about AI/ML's ability to enhance power quality management. As these technologies mature, they will help create a more resilient, efficient, and sustainable energy system, reducing downtime and improving the longevity of power infrastructure. Their impact extends beyond operational efficiency, contributing to a future where energy reliability meets the growing demands of modern society.