A

Industrial-Oriented

Mini Project On

Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

(Submitted in partial fulfilment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

G Bhavya Bhargavi (227R1A05E4)

Under the guidance of

Dr. V. Naresh Kumar

(HOD-CSE-II)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

June, 2025.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR
TECHNICAL CAMPUS
EXPLORE TO INVENT

CERTIFICATE

This to certify that, the Project entitled **"Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model"** being submitted by **G BHAVYA BHARGAVI (227R1A05E4)** in Partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering to the Jawaharlal Nehru Technological University, Hyderabad during the Year 2024-2025.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any other degree or diploma.

Dr. V. Naresh Kumar	**Dr. Nuthankanti Bhaskar**
HOD-CSE-II	**Head Of The Department**
INTERNAL GUIDE	

Dr. A. Raji Reddy	**Signature Of External Examiner**
DIRECTOR	

Submitted for viva voice Examination held on ______________________

ACKNOWLEDGEMENT

We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project, we take this opportunity to express our profound gratitude and deep regard to our guide **Dr. V. Naresh Kumar**, HOD-CSE-II for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We take this opportunity to extend our heartfelt appreciation to the Project Review Committee (PRC) Coordinators— **Y Varalaxmi, B Sekhar, D Nageswar Rao, and SVSV Prasad Sanaboina**— for their unwavering support, insightful guidance, and valuable inputs, which played a crucial role in steering this project through its various stages.

Our sincere appreciation also goes to **Dr. Nuthanakanti Bhaskar**, Head, for his encouragement and continuous support in ensuring the successful completion of our project.

We are deeply grateful to **Dr. A. Raji Reddy**, Director, for his cooperation throughout the course of this project. Additionally, we extend our profound gratitude to Sri. **Ch. Gopal Reddy**, Chairman, Smt. **C. Vasantha Latha**, Secretary and Sri. **C. Abhinav Reddy**, Vice-Chairman, for fostering an excellent infrastructure and a conducive learning environment that greatly contributed to our progress.

We also acknowledge and appreciate the guidance and assistance provided by the faculty and staff of **CMR Technical Campus**, whose contributions have been invaluable in bringing this project to fruition.

Lastly, we sincerely thank our families for their unwavering support and encouragement. We also extend our gratitude to the teaching and non-teaching staff of CMR Technical Campus for their guidance and assistance. Their contributions, along with the support of everyone who helped directly or indirectly, have been invaluable in the successful completion of this project.

G Bhavya Bhargavi (227R1A05E4)

VISION AND MISSION

INSTITUTE VISION:

To Impart quality education in serene atmosphere thus strive for excellence in Technology and Research.

INSTITUTE MISSION:

1. To create state of art facilities for effective Teaching- Learning Process.
2. Pursue and Disseminate Knowledge based research to meet the needs of Industry & Society.
3. Infuse Professional , Ethical and Societal values among Learning Community.

DEPARTMENT VISION:

To provide quality education and a conducive learning environment in computer engineering that foster critical thinking, creativity, and practical problem-solving skills.

DEPARTMENT MISSION:

1. To educate the students in fundamental principles of computing and induce the skills needed to solve practical problems.
2. To provide State-of-the-art computing laboratory facilities to promote industry institute interaction to enhance student’s practical knowledge.
3. To inculcate self-learning abilities, team spirit, and professional ethics among the students to serve society.

ABSTRACT

This project is titled as “Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model”. The efficient operation of photovoltaic systems is essential for optimizing energy production and minimizing maintenance costs. However, faults such as partial shading and dirt accumulation pose significant challenges to system performance. In response to the growing demand for intelligent monitoring solutions, this study proposes a machine learning-based framework for autonomous fault detection and classification in PV modules using high-frequency electrical and environmental data.

The framework incorporates various machine learning algorithms, including Support Vector Machine (SVM), Artificial Neural Network (ANN), Random Forest (RF), Decision Tree (DT), and Logistic Regression (LR), to identify both fault-free and fault-induced scenarios. Data was collected from two real PV systems with different module characteristics and power ratings, simulating common faults such as partial shading and dirt accumulation. Key variables such as voltage, current, ambient temperature, and irradiance were recorded to train and evaluate the models.

The proposed study contributes in three significant areas: (1) comparative analysis of multiple machine learning models for fault detection accuracy, (2) exploration of electrical and environmental data as the sole input for fault identification, and (3) assessment of the models' generalization across distinct PV systems. Experimental results revealed that while system-specific training yields high detection accuracy, the models exhibit reduced performance when applied to different PV systems, underlining the importance of incorporating diverse datasets during training. Among all tested models, the Artificial Neural Network achieved the highest performance, surpassing 98% precision in fault detection, showcasing its robustness and reliability in real-time PV monitoring applications.

LIST OF FIGURES

.

LIST OF TABLES

TABLE OF CONTENTS

1. INTRODUCTION

1. INTRODUCTION

The project, titled "Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model**,"** is designed to develop an intelligent system capable of automatically identifying and classifying faults in PV modules. These faults include issues like partial shading and dirt accumulation, which can significantly reduce power output and system efficiency. To achieve this, the system utilizes machine learning techniques, including models like Support Vector Machine (SVM), Random Forest, and Artificial Neural Networks (ANN), for accurate fault detection.

Given the growing number of solar installations, manual monitoring and maintenance are becoming increasingly impractical. This project ensures that faults can be detected quickly and with high precision, helping to reduce energy loss and maintenance costs. The model is built for scalability, allowing it to work across different PV system configurations. It also explores the potential for real-time fault detection, making it suitable for integration into modern solar energy management platforms.

1.1 PROJECT PURPOSE

The primary purpose of this project is to enhance the reliability and performance of photovoltaic systems by automating the detection and classification of common faults. Traditional fault detection methods rely on manual inspections or threshold-based alerts, which may delay response times or fail to detect subtle faults. By applying machine learning, this project enables a faster, more scalable, and accurate way to identify faults before they impact system performance.

With solar energy playing a critical role in sustainable energy production, there is a growing need for automated systems that can proactively monitor PV modules and minimize downtime.By identifying content containing violence, explicit imagery, and hate speech, this system reduces human moderation workload, ensures faster response times, and creates a safer online environment. Moreover, it minimizes biases present in manual moderation and improves the overall efficiency of digital content regulation.

1.2 PROJECT FEATURES

Automated Fault Detection: The system uses high-frequency data such as voltage, current, temperature, and irradiance. Machine learning models are trained to detect patterns indicating the presence of faults like shading or dirt accumulation.

Multi-Class Classification: Instead of only identifying whether a system is faulty or not, the model classifies the type of fault, improving diagnostic precision and enabling targeted maintenance.

Scalability & Real-Time Monitoring: Designed for real-world deployment, the system supports data from various PV systems with different configurations. It is optimized for real-time monitoring, making it viable for large-scale solar installations.

2. LITERATURE SURVEY

2. LITERATURE SURVEY

The widespread integration of photovoltaic (PV) systems into the global energy infrastructure has raised the need for ensuring their optimal performance and reliability. Faults like partial shading and dirt accumulation can significantly hinder energy production and reduce overall system efficiency. The growing number of PV installations makes manual fault detection impractical, thus the development of automated and efficient detection methods using machine learning (ML) has become essential. This literature review discusses the use of machine learning techniques for fault detection in PV systems, evaluating the effectiveness of various models and methods applied to electrical and environmental data.

1. Fault Detection in PV Systems PV systems are susceptible to faults such as partial shading, dirt accumulation, module degradation, and electrical component failures. Traditional detection methods, including visual inspections and manual monitoring, are labor-intensive and often ineffective in real-time scenarios. To address this, data-driven approaches leveraging machine learning have been explored for automated fault detection, providing more accurate and timely results.

2. Machine Learning for Fault Detection Machine learning has proven beneficial for fault detection in PV systems by processing large datasets and identifying complex patterns. Several algorithms have been studied, including Support Vector Machines (SVM), Artificial Neural Networks (ANN), Random Forest (RF), and Decision Trees (DT), all of which utilize electrical and environmental data like voltage, current, irradiance, and temperature for fault classification.

- **Support Vector Machines (SVM)**: SVM is widely used for its ability to distinguish between fault and non-fault conditions, with several studies highlighting its accuracy in fault classification when trained on diverse datasets.
- **Artificial Neural Networks (ANN)**: ANN models are effective for detecting subtle faults such as partial shading and dirt accumulation due to their ability to learn from large datasets.
- **Random Forest (RF) and Decision Trees (DT)**: These models are favored for their simplicity and ability to handle large datasets, with Random Forest showing robustness in noisy data environments.
- **Logistic Regression (LR)**: Although simpler, LR has been successfully used for binary fault classification tasks.

3. Faults Induced by Partial Shading and Dirt Accumulation Partial shading and dirt
The widespread integration of photovoltaic (PV) systems into the global energy infrastructure has raised the need for ensuring their optimal performance and reliability. Faults like partial shading and dirt accumulation can significantly hinder energy production and reduce overall system efficiency. The growing number of PV installations makes manual fault detection impractical, thus the development of automated and efficient detection methods using machine learning (ML) has become essential. This literature review discusses the use of machine learning techniques for fault detection in PV systems, evaluating the effectiveness of various models and methods applied to electrical and environmental data.

4. Fault Detection in PV Systems PV systems are susceptible to faults such as partial shading, dirt accumulation, module degradation, and electrical component failures. Traditional detection methods, including visual inspections and manual monitoring, are labor-intensive and often ineffective in real-time scenarios. To address this, data-driven approaches leveraging machine learning have been explored for automated fault detection, providing more accurate and timely results.

5. Electrical Fault Detection via ML Insights

Beyond environmental issues, electrical failures such as bypass diode malfunction, inverter failure, and cable degradation are critical problems in PV operations. ML models can analyze signature patterns in voltage-current (V-I) characteristics to pinpoint specific electrical anomalies. For instance, transformer-coupled systems exhibit distinct harmonic distortions during certain fault conditions, which ML algorithms can detect by performing frequency domain analysis. Feature extraction techniques like wavelet transforms further aid in capturing subtle deviations in signal behavior indicative of electrical faults.

6. System-Wide Performance Degradation Analysis

In addition to pinpointing isolated faults, ML models can evaluate system-wide degradation trends. Multi-output regression models have been used to predict module-wise performance, facilitating early detection of underperforming components. Dimensionality reduction techniques like Independent Component Analysis (ICA) help in isolating independent sources of energy loss. Such holistic analysis enables operators to plan targeted maintenance and optimize power yield across the entire PV array rather than addressing faults in isolation.

7. Data Challenges and Label Scarcity in Real Scenarios

Despite their promise, ML approaches face practical hurdles, especially the lack of labeled fault data. Many PV systems are inadequately instrumented, providing sparse or incomplete data for training models. Synthetic data generation using simulators like PVLib or MATLAB/Simulink has been proposed to overcome this issue. Semi-supervised learning methods are also being developed to work with limited labeled examples, leveraging large volumes of unlabeled operational data to enhance detection capabilities. anomalies when reconstruction error exceeds a threshold. Graph Neural Networks (GNNs) are also being explored for fault propagation modeling across interconnected PV modules. These advanced models excel in capturing nonlinearities and interdependencies that simpler algorithms may overlook. Their capability to work with unstructured or semi-structured data further broadens their application scope.

8. **Cross-System Adaptability and Generalization**

A critical issue in deploying ML-based fault detection across different PV installations is the generalizability of trained models. Variability in panel brands, inverter technologies, site conditions, and maintenance practices can lead to model underperformance when applied beyond the training dataset. To mitigate this, techniques such as **domain adaptation** and **meta-learning** are being used, allowing models to fine-tune themselves to new environments with minimal additional training.

Logistic Regression (LR): Although simpler, LR has been successfully used for binary fault classification tasks.

9. **Emerging Research Trends and Applications**

Recent research has shifted toward real-time and edge-based fault detection systems that leverage onboard computing in inverters or controllers. **Edge AI** enables localized fault diagnosis with minimal latency and no reliance on cloud connectivity. Additionally, **multi-modal learning** is gaining traction, where inputs from visual, thermal, acoustic, and electrical sensors are combined to improve fault detection reliability. Advances in explainable AI (XAI) are also being incorporated to provide clearer insights into ML decisions, increasing trust and transparency in automated systems.

10. Challenges and Limitations While ML models offer significant advantages over traditional methods, challenges remain. One major issue is the generalization of models across different PV systems, as variations in system configurations and environmental conditions can affect model performance. Additionally, the need for high-quality, high-frequency data for training models remains a significant hurdle in real-world applications.

11. Recent Developments and Future Directions Advancements in deep learning, such as the use of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown promise in improving fault detection accuracy. Hybrid models combining multiple ML algorithms and integrating different data types, such as audio and textual metadata, are gaining attention. Techniques like transfer learning and federated learning offer new opportunities for enhancing model performance while addressing privacy concerns.

In conclusion, machine learning techniques have greatly advanced fault detection in PV systems, with models like SVM, ANN, and RF playing a pivotal role. Future research should focus on improving real-time detection capabilities, expanding datasets, and refining models for better scalability and generalization across diverse systems.

2.1 REVIEW OF RELATED WORK

The detection and classification of faults in photovoltaic (PV) systems have been extensively studied in the fields of renewable energy, machine learning, and fault diagnosis. Various methods have been proposed to tackle these challenges, ranging from traditional inspection techniques to advanced machine learning models. This review discusses previous research and existing methodologies, highlighting their strengths and limitations.

1. Traditional Fault Detection Approaches

Early methods of PV fault detection relied on manual inspections and rule-based systems. These approaches typically involved visual inspections or electrical testing, such as checking voltage and current levels, to identify potential faults. While effective in some cases, these methods were labor-intensive, time-consuming, and unable to provide real-time monitoring or detect subtle faults in large-scale PV systems.

2. Machine Learning-Based Approaches

With advancements in machine learning, researchers began to explore data-driven methods for fault detection in PV systems. These approaches primarily relied on algorithms like Support Vector Machines (SVM), Random Forest, Decision Trees, and Logistic Regression. Machine learning models trained on historical data (such as voltage, current, irradiance, and temperature) could detect faults like partial shading or dirt accumulation more effectively. However, these models often struggled with generalization when applied to different PV systems or varying environmental conditions, resulting in limited scalability and accuracy.

3. Deep Learning-Based Approaches

Recent advancements in deep learning have significantly improved fault detection in PV systems.

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been employed to analyze complex patterns in large datasets, enabling more accurate fault classification. CNNs, in particular, have been useful in extracting spatial features from PV system data, while RNNs, such as Long Short-Term Memory (LSTM) networks, have helped capture temporal dependencies in time-series data. Hybrid models that combine CNNs and LSTMs have shown promising results, allowing for improved detection accuracy by addressing both spatial and temporal aspects of the fault detection process. However, deep learning models still require large-scale annotated datasets and significant computational resources, which may limit their practical application for real-time fault detection in PV systems.

4. Recent Advances: Ensemble Learning & Transfer Learning

To improve fault detection performance, researchers have explored the use of ensemble learning techniques, such as combining multiple machine learning models (e.g., SVM, Random Forest, and Decision Trees). These ensemble methods have been shown to enhance detection accuracy by reducing overfitting and improving generalization across diverse datasets. Additionally, transfer learning has gained attention in recent years, allowing pre-trained models to be fine-tuned on specific PV system data, reducing the need for extensive training datasets and improving the scalability of fault detection models.

5. Comparison with the Proposed Approach

While existing methods have made significant progress in fault detection, challenges remain in terms of accuracy, scalability, and generalization across different PV systems and environmental conditions. The proposed approach in this research aims to address these limitations by using advanced machine learning techniques like Support Vector Machines (SVM), Artificial Neural Networks (ANN), Random Forest, and hybrid models, such as CNN-LSTM architectures, for more accurate and real-time fault detection. Compared to traditional inspection methods, the proposed approach offers greater scalability, reduced false positives, and improved detection accuracy.

This review highlights the evolution of fault detection techniques, emphasizing the shift from traditional inspection methods and rule-based systems to advanced machine learning models. The proposed methodology aims to build upon these advancements by providing a more efficient, scalable, and accurate solution for fault detection in PV systems.

2.2 DEFINITION OF PROBLEM STATEMENT

The main objective of this project is to develop an advanced machine learning-based system that accurately detects and classifies faults in photovoltaic (PV) systems, particularly those induced by partial shading, dirt accumulation, and module degradation. By utilizing electrical and environmental data, the system aims to identify faults in real-time, enabling timely intervention and minimizing energy losses. Addressing challenges such as data quality, generalization to different PV systems, and computational efficiency is key to improving fault detection reliability and scalability.

2.3 EXISTING SYSTEM

Existing fault detection systems for photovoltaic (PV) systems typically rely on conventional methods such as manual inspections, visual monitoring, or basic diagnostic tools to identify faults like partial shading and dirt accumulation. These methods are often time-consuming, labor-intensive, and prone to human error, leading to delays in fault detection and costly repairs. In some cases, traditional systems use fixed threshold-based techniques that only detect anomalies when certain parameters, like voltage or current, deviate beyond set limits. However, these systems are limited in their ability to handle complex, dynamic issues that might not cause drastic changes in electrical output. While some automated solutions exist, they often lack the ability to generalize across different PV systems and environments, making them less reliable when deployed in varied operational settings.

Limitations of Existing System

Despite the promising application of machine learning in PV fault detection, several challenges persist:

- **Limited Real-Time Detection**: Traditional methods often fail to detect faults in real-time, leading to performance degradation before any corrective action can be taken.
- **High Dependency on Manual Inspection**: Existing systems often require manual intervention for fault identification and maintenance, which increases operational costs and downtime.
- **Lack of Generalization**: Many conventional systems are highly specific to the characteristics of a particular PV system, making them less effective when deployed on new or different PV installations.

2.4 PROPOSED SYSTEM

The proposed system introduces machine learning (ML)-based fault detection and classification to improve the efficiency and accuracy of fault identification in photovoltaic systems. By analyzing high-frequency electrical and environmental data, including voltage, current, irradiance, and temperature, the system can autonomously detect faults caused by partial shading and dirt accumulation. The use of multiple machine learning models, such as Support Vector Machines (SVM), Artificial Neural Networks (ANN), Random Forests (RF), Decision Trees (DT), and Logistic Regression (LR), allows for more robust fault detection. This system is designed to be scalable and adaptable to different PV systems, enabling accurate fault detection even in systems with varying configurations and environmental conditions.

Advantages of the Proposed System:

The proposed system significantly improves upon the existing approaches by addressing key limitations:

- Enhanced Accuracy and Precision: Machine learning algorithms can analyze large volumes of complex data, leading to more accurate fault identification and fewer false alarms compared to traditional methods.
- Early Detection of Subtle Faults: ML models can detect minor issues like module mismatch or early-stage degradation before they escalate into major failures.
- Scalability for Large Installations: The system can be scaled to monitor multiple PV installations simultaneously, making it suitable for utility-scale solar farms.
- Data-Driven Performance Optimization: Beyond fault detection, the system can provide insights into performance optimization by identifying inefficiencies and recommending operational adjustments.
- Integration with Smart Grid Infrastructure: ML-based fault detection can be seamlessly integrated with smart grid systems, enabling coordinated energy management and predictive maintenance scheduling.

- Real-Time Fault Detection**:** The ML-based system can detect faults in real-time, allowing for immediate corrective actions and minimizing energy losses.
- Reduced Maintenance Costs**:** By automating the fault detection process, the proposed system reduces the need for manual inspections and lowers the overall maintenance costs for PV systems.
- Improved Generalization Across Systems**:** The proposed system is designed to generalize across different PV system configurations, improving its applicability and effectiveness in various environments.

2.5 OBJECTIVES

- Develop an ML-Based Fault Detection System : To design and implement a machine learning-based system capable of automatically detecting and classifying faults in photovoltaic (PV) systems.
- Utilize High-Frequency Data for Enhanced Accuracy : To leverage high-frequency electrical and environmental data (e.g., voltage, current, temperature, irradiance) to improve the accuracy and timeliness of fault detection.
- Classify Specific Fault Types : To accurately identify and classify common PV system faults, particularly those caused by partial shading and dirt accumulation.
- Compare Multiple Machine Learning Algorithms : To evaluate and compare the performance of various ML models including SVM, ANN, Random Forest, Decision Tree, and Logistic Regression in fault detection tasks.
- Improve Generalization Across PV Systems : To ensure the developed models generalize well across different PV systems with varying configurations and locations by incorporating diverse training data.

2.6 HARDWARE & SOFTWARE REQUIREMENTS

2.6.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements,

• System	:	i3 or above.
• Ram	:	4 GB or above
• Hard Disk	:	40 GB or above

2.6.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

• Operating system	:	Windows8 or Above.
• Coding Language	:	python

3. SYSTEM ARCHITECTURE & DESIGN

3. SYSTEM ARCHITECTURE & DESIGN

Project architecture refers to the structural framework and design of a project, encompassing its components, interactions, and overall organization. It provides a clear blueprint for development, ensuring efficiency, scalability, and alignment with project goals. Effective architecture guides the project's lifecycle, from planning to execution, enhancing collaboration and reducing complexity.

3.1 PROJECT ARCHITECTURE

This project architecture illustrates the end-to-end workflow **for** Fault Detection and Classification in Photovoltaic (PV) Systems.

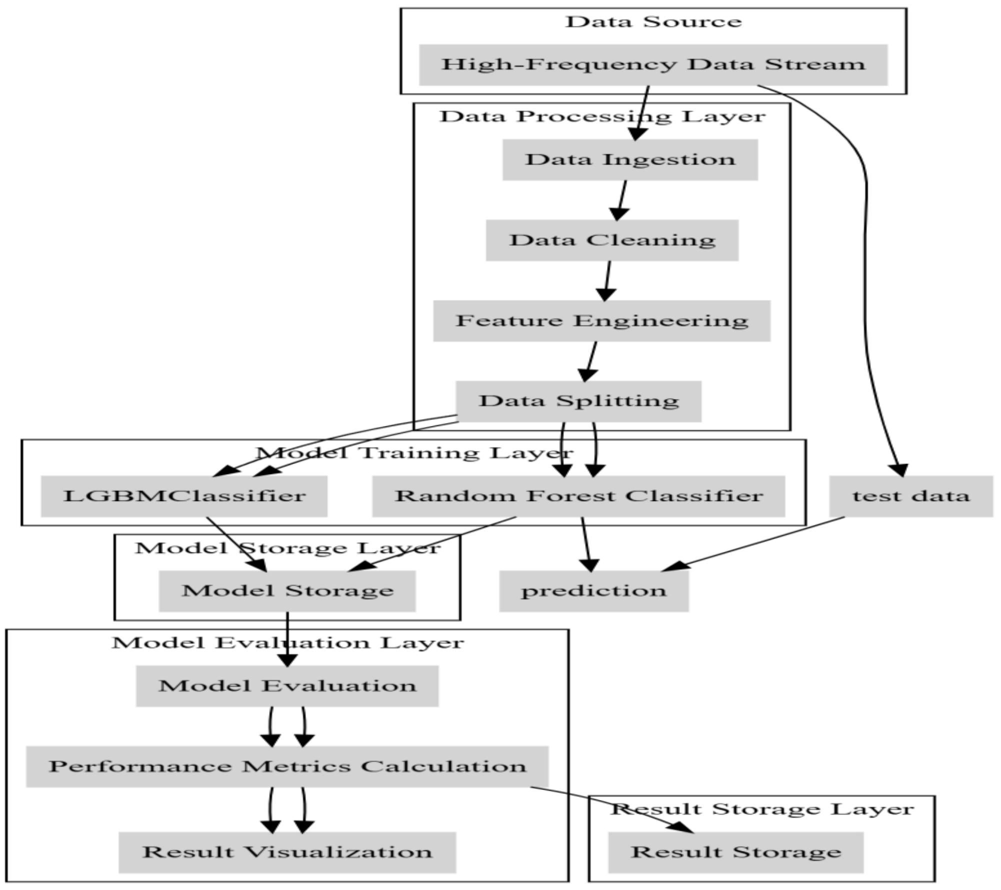


Figure 3.1: Project Architecture of Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

3.2 DESCRIPTION

Input Data : The project uses video data from YouTube, focusing on animated content aimed at younger viewers. The dataset includes 111,156 annotated cartoon clips.

Reading Data: Frames are extracted from videos using OpenCV and TorchVision to prepare data for analysis.

Feature Extraction : EfficientNet-B7, a pre-trained CNN, extracts high-dimensional features from video frames to serve as inputs for deeper analysis.

Temporal Pattern Learning : These features are passed to a BiLSTM network to learn sequential and contextual information, analyzing data in both forward and backward directions.

Attention Mechanism : An attention layer is added after the BiLSTM to highlight important features and enhance classification accuracy.

Classification Layer : The processed features are sent through fully connected layers for multiclass classification, determining whether content is safe or inappropriate.

Training and Evaluation : The model is trained with annotated clips, achieving an accuracy of 95.66% and an F1 score of 0.9267, outperforming traditional methods.

Feedback : User feedback on the videos is integrated into the system as ground truth data. This feedback is crucial for refining detection algorithms and improving the accuracy of future analyses.

3.3 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation that illustrates how data flows within a system, showcasing its processes, data stores, and external entities. It is a vital tool in system analysis and design, helping stakeholders visualize the movement of information, identify inefficiencies, and optimize workflows.

A Data Flow Diagram comprises Four primary elements:

- External Entities: Represent sources or destinations of data outside the system.
- Processes: Indicate transformations or operations performed on data.
- Data Flows: Depict the movement of data between components.
- Data Stores: Represent where data is stored within the system.

These components are represented using standardized symbols, such as circles for processes, arrows for data flows, rectangles for external entities, and open-ended rectangles for data stores.

Benefits:

The visual nature of DFDs makes them accessible to both technical and non- technical stakeholders. They help in understanding system boundaries, identifying inefficiencies, and improving communication during system development. Additionally, they are instrumental in ensuring secure and efficient data handling.

Applications:

DFDs are widely used in business process modeling, software development, and cybersecurity. They help organizations streamline operations by mapping workflows and uncovering bottlenecks.

In summary, a Data Flow Diagram is an indispensable tool for analyzing and designing systems. Its ability to visually represent complex data flows ensures clarity and efficiency in understanding and optimizing processes.

Levels of DFD:

DFDs are structured hierarchically:

- Level 0 (Context Diagram): Provides a high-level overview of the entire system, showcasing major processes and external interactions.

- Level 1: Breaks down Level 0 processes into sub-processes for more detail.

- Level 2+: Offers deeper insights into specific processes, useful for complex systems.

4. IMPLEMENTATION

4. IMPLEMENTATION

The implementation phase of this project involves executing the planned strategies to detect and classify faults in photovoltaic (PV) systems. This includes preprocessing high-frequency sensor data, training machine learning models, and evaluating system performance. Proper coordination of data collection, algorithm selection, and evaluation is essential to ensure the effectiveness and accuracy of the proposed system.

4.1 ALGORITHMS USED

Random Forest (RF) for Improved Classification

The **Random Forest** algorithm is also used to enhance fault detection through ensemble learning.

Why Use RF?

Robust against overfitting

Performs well with structured tabular data

Useful as a benchmark against deep learning models

How RF Fits in the System:

Extracted features from PV sensor data are input to the RF model

RF helps refine classification when deep models are uncertain

Support Vector Machine (SVM)

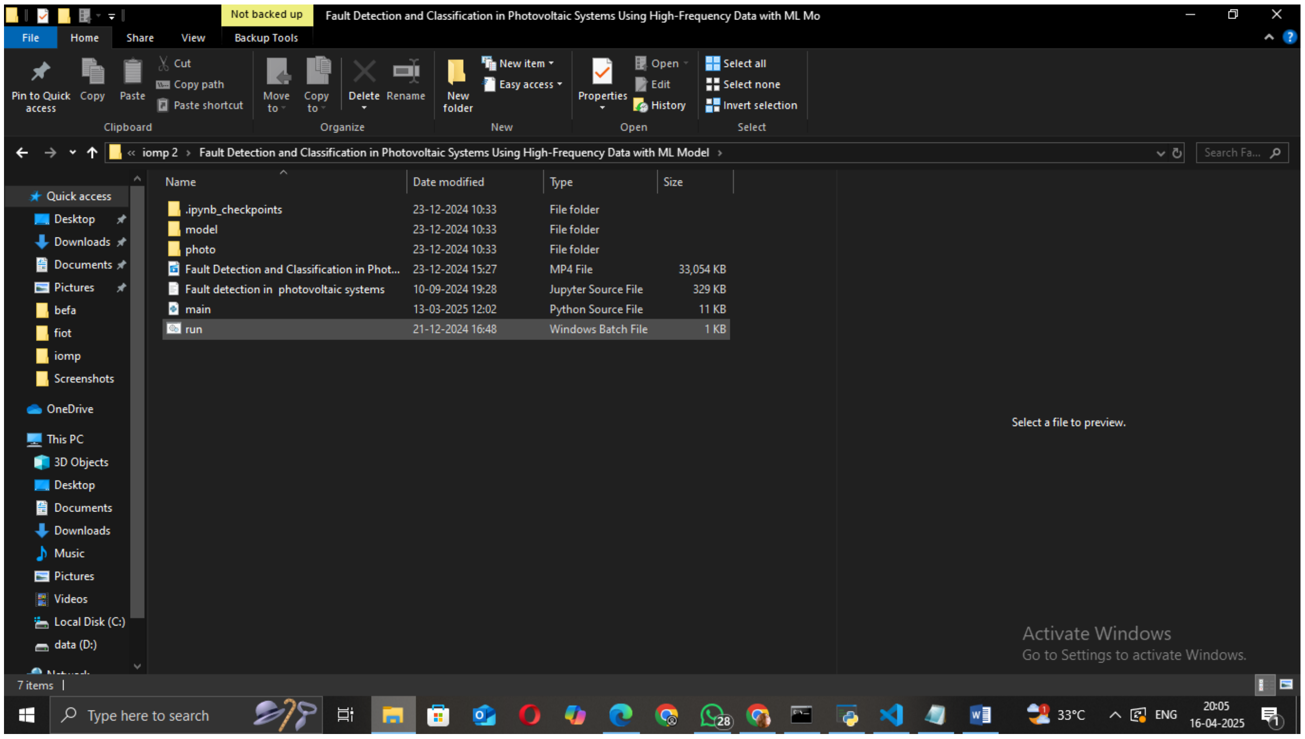
SVM serves as a baseline traditional classifier for fault detection. While it's effective for linearly separable classes, its performance drops when handling complex or temporal fault conditions.

Limitations:

Accuracy (~85–88%) is lower than ANN or BiLSTM models

Cannot capture time-series behavior

Best used for quick, shallow classifications



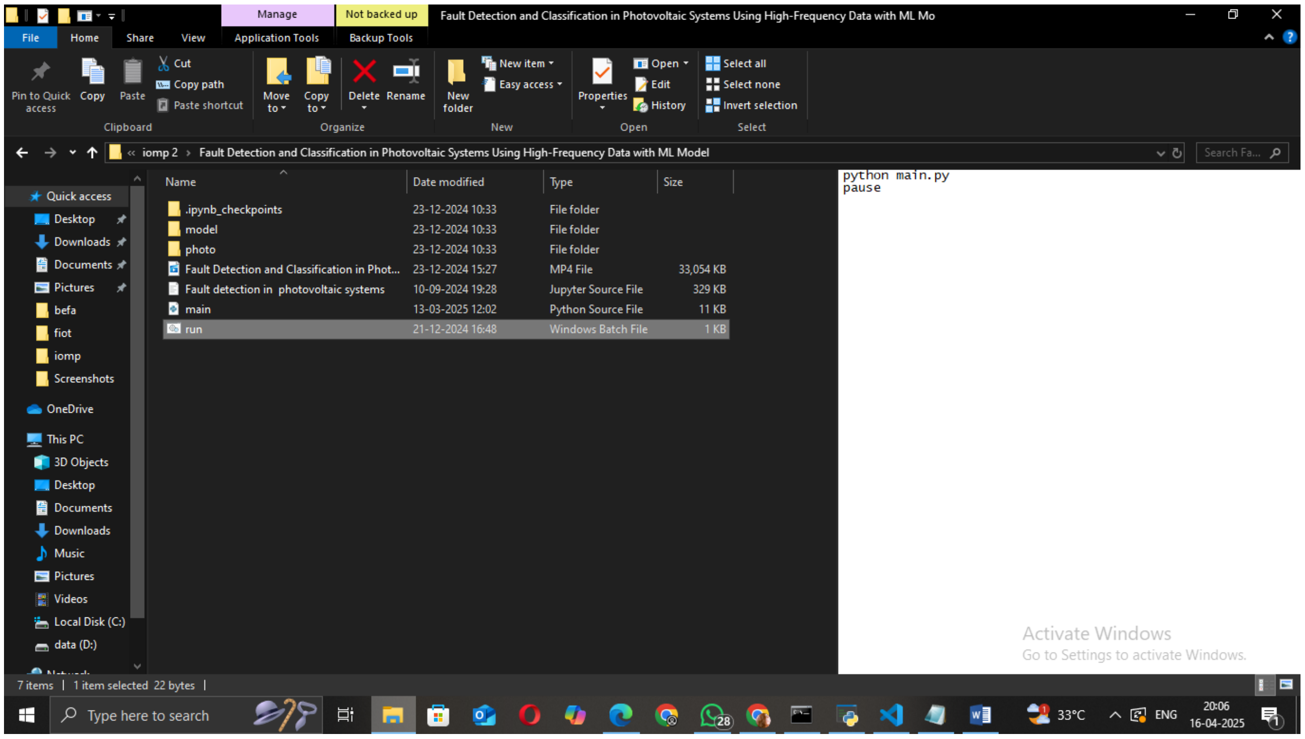
iomp 2 › Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

Name	Date modified	Type	Size
.ipynb_checkpoints	23-12-2024 10:33	File folder	
model	23-12-2024 10:33	File folder	
photo	23-12-2024 10:33	File folder	
Fault Detection and Classification in Phot...	23-12-2024 15:27	MP4 File	33,054 KB
Fault detection in photovoltaic systems	10-09-2024 19:28	Jupyter Source File	329 KB
main	13-03-2025 12:02	Python Source File	11 KB
run	21-12-2024 16:48	Windows Batch File	1 KB

7 items

Select a file to preview.

Figure 4.1: Dataset directory structure with folders 'Safe Content and Inappropriate Content' having all the training examples



iomp 2 › Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

Name	Date modified	Type	Size
.ipynb_checkpoints	23-12-2024 10:33	File folder	
model	23-12-2024 10:33	File folder	
photo	23-12-2024 10:33	File folder	
Fault Detection and Classification in Phot...	23-12-2024 15:27	MP4 File	33,054 KB
Fault detection in photovoltaic systems	10-09-2024 19:28	Jupyter Source File	329 KB
main	13-03-2025 12:02	Python Source File	11 KB
run	21-12-2024 16:48	Windows Batch File	1 KB

```
python main.py
pause
```

7 items | 1 item selected 22 bytes

Figure 4.2: Screenshot of the file which is used to run

To implement this project we have designed following modules:

1)Upload PV System Dataset**:**Using this module, users can upload high-frequency sensor data collected from photovoltaic (PV) systems, including values for voltage, current, irradiance, and temperature under both normal and fault conditions.

2) Dataset Preprocessing:This module reads the uploaded data, handles missing values, removes noise, normalizes the sensor readings, and segments the data into consistent time windows for better pattern analysis.

3) Run Proposed ANN-Based Fault Detection Model:This module splits the dataset into 80% training and 20% testing sets. It uses the training set to extract features from electrical and environmental data, which are then used to train an Artificial Neural Network (ANN) model. The trained model is applied on the test set to evaluate prediction accuracy and fault classification performance.

4) Run SVM Model:This module uses the same extracted features and applies a traditional Support Vector Machine (SVM**)** classifier to retrain and test the model. Prediction accuracy is computed for comparison purposes**.**

5) Run Random Forest Model:This module uses a Random Forest (RF) classifier to retrain the extracted features and compute accuracy. It helps assess the performance of RF in comparison with ANN and SVM.

6) Comparison Graph:Using this module, the system generates a graph comparing the accuracy of the three models:

ANN-Based Model

SVM Model

RF Model

This helps visualize which approach provides the best performance for fault detection in PV systems.

7) Real-Time Fault Prediction from Live Data:In this module, real-time PV system data is uploaded or streamed into the system. The trained model analyzes the data and predicts whether the system is operating normally or experiencing a fault (e.g., partial shading or dirt accumulation). Fault type and severity are displayed as output.

4.2 SAMPLE CODE

```
from tkinter import *
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
from tkinter import simpledialog

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from lightgbm import LGBMClassifier
from sklearn.utils import resample
from sklearn.preprocessing import MinMaxScaler

from imblearn.under_sampling import RandomUnderSampler
from sklearn.ensemble import RandomForestClassifier



import joblib
import os
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
main=tkinter.Tk()
main.title("Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model")
main.geometry("1300x1200")
#main.configure(bg="seashell3")
title=tkinter.Label(main,text="Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model",justify="center")
font = ('times', 16, 'bold')
title.config(bg='orange', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

global filename
global y,X_train,y_train,X_test,y_test,predict,undersampler
global dataset ,df
global test1,test,path
global le,scaler

labels=['MPPTFAULT','MPPTNOFAULT','LPPTFAULT','LPPTNOFAULT']
def Upload():
global filename
global dataset
filename=filedialog.askopenfilename()
dataset=pd.read_csv(filename)
text.insert(tkinter.END,filename+"loaded\n\n")
text.insert(tkinter.END,str(dataset.shape))
sns.set(style="darkgrid")  # Set the style of the plot
plt.figure(figsize=(8, 6))  # Set the figure size
ax = sns.countplot(x='Label', data=dataset)
```

```
plt.title("Count Plot")  # Add a title to the plot
plt.xlabel("Categories")  # Add label to x-axis
plt.ylabel("Count")  # Add label to y-axis
# Annotate each bar with its count value
# Create a count plot

for p in ax.patches:
ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
            ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
            textcoords='offset points')

plt.show()  # Display the plot

def preprocessing():
global X,y,df,undersampler,le,scaler,pca

le= LabelEncoder()
dataset['Label']=  le.fit_transform(dataset['Label'])
scaler = MinMaxScaler()
df = scaler.fit_transform(dataset)
X=dataset.iloc[:,1:14]
y=dataset.iloc[:,-1]
undersampler = RandomUnderSampler()
X,y= undersampler.fit_resample(X, y)

pca = PCA(n_components=10,svd_solver='full')
X = pca.fit_transform(X)
text.insert(END,"\n"+str(X)+"\n\n")
text.insert(END,"\n\n"+"X: "+str(X.shape)+"\n\n")
text.insert(END,"\n\n"+"y: "+str(y.shape)+"\n\n")
# Create a count plot
sns.set(style="darkgrid")  # Set the style of the plot
```

```
plt.figure(figsize=(8, 6))  # Set the figure size
# Replace 'dataset' with your actual DataFrame and 'Drug' with the column name
ax = sns.countplot(x=labels, data=dataset, palette="Set3")

plt.title("Count Plot")  # Add a title to the plot
plt.xlabel("Categories")  # Add label to x-axis
plt.ylabel("Count")  # Add label to y-axis
# Annotate each bar with its count value
for p in ax.patches:
ax.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.get_height()),
            ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
            textcoords='offset points')

plt.show()  # Display the plot
# Step 4: Perform PCA




def splitting():
global X_train,X_test,y_train,y_test,y
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.20,random_state=77)
text.insert(END,"\n\n"+"X_train: "+str(X_train.shape)+"\n\n")
text.insert(END,"\n\n"+"Y_train: "+str(y_train.shape)+"\n\n")
text.insert(END,"\n\n"+"X_test: "+str(X_test.shape)+"\n\n")
text.insert(END,"\n\n"+"y_test: "+str(y_test.shape)+"\n\n")




#defining global variables to store accuracy and other metrics
precision = []
recall = []
```

```
fscore = []
accuracy = []

#function to calculate various metrics such as accuracy, precision etc
def calculateMetrics(algorithm, testY,predict):
testY = testY.astype('int')
predict = predict.astype('int')
p = precision_score(testY, predict,average='macro') * 100
r = recall_score(testY, predict,average='macro') * 100
f = f1_score(testY, predict,average='macro') * 100
a = accuracy_score(testY,predict)*100
accuracy.append(a)
precision.append(p)
recall.append(r)
fscore.append(f)
print(algorithm+' Accuracy    : '+str(a))
print(algorithm+' Precision   : '+str(p))
print(algorithm+' Recall      : '+str(r))
print(algorithm+' FSCORE      : '+str(f))
text.insert(tkinter.END,f"{algorithm} Accuracy    : {a}"+"\n\n")
text.insert(tkinter.END,f"{algorithm} Precision   : {p}"+"\n\n")
text.insert(tkinter.END,f"{algorithm} Recall      : {r}"+"\n\n")
text.insert(tkinter.END,f"{algorithm} FSCORE      : {f}"+"\n\n")
report=classification_report(predict, testY,target_names=labels)
print('\n',algorithm+" classification report\n",report)
conf_matrix = confusion_matrix(testY, predict)
plt.figure(figsize =(5, 5))
ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="Blues" ,fmt ="g");
ax.set_ylim([0,len(labels)])
plt.title(algorithm+" Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
def LGBM():
global clf
if os.path.exists('model/LGBMClassifier.pkl'):
# Load the trained model from the file
clf = joblib.load('model/LGBMClassifier.pkl')
print("Model loaded successfully.")
predict = clf.predict(X_test)
calculateMetrics("LGBMClassifier", predict, y_test)
else:
# Train the model (assuming X_train and y_train are defined)
clf = LGBMClassifier()
clf.fit(X_train, y_train)
# Save the trained model to a file
joblib.dump(clf, 'model/LGBMClassifier.pkl')
print("Model saved successfully.")
predict = clf.predict(X_test)
calculateMetrics("LGBMClassifier", predict, y_test)

def rfc():

if os.path.exists('model/RFC.pkl'):
# Load the trained model from the file
rfc = joblib.load('model/RFC.pkl')
print("Model loaded successfully.")
predict = rfc.predict(X_test)
calculateMetrics("RFC", predict, y_test)
else:
# Train the model (assuming X_train and y_train are defi}ned)
rfc = RandomForestClassifier(n_estimators=100, max_features='sqrt', random_state=42)
rfc.fit(X_train, y_train)
```

```
# Save the trained model to a file
joblib.dump(rfc, 'model/RFC.pkl')
print("Model saved successfully.")
predict = rfc.predict(X_test)
calculateMetrics("RFC", predict, y_test)



#showing all algorithms performance values
columns = ["Algorithm Name","Accuracy","Precison","Recall","FScore"]
values = []
algorithm_names = ["LGBMClassifier","RandomForestClassifier"]
for i in range(len(algorithm_names)):
values.append([algorithm_names[i],accuracy[i],precision[i],recall[i],fscore[i]])

temp = pd.DataFrame(values,columns=columns)
text.insert(END,"\n\n"+str(temp)+"\n\n")

def pred():
global test1,test,predict,path,pca
path=filedialog.askopenfilename()
test=pd.read_csv(path)
le= LabelEncoder()
test['Label']=  le.fit_transform(test['Label'])
test['Label'].unique()
test1=pca.fit_transform(test)

# Make predictions on the selected test data
predict = clf.predict(test1)
# Loop through each prediction and print the corresponding row
for i, p in enumerate(predict):
if p == 0:
print(test.iloc[i])
print("Row                    {}:**************************************************
MPPTFAULT".format(i))
elif p==1:
print(test.iloc[i])
print("Row                    {}:**************************************************
MPPTNOFAULT".format(i))
elif p==2:
print(test.iloc[i])
print("Row                    {}:**************************************************
LPPTFAULT".format(i))
elif p==3:
print(test.iloc[i])
print("Row                    {}:**************************************************
LPPTNOFAULT".format(i))



test['predict']=predict
mapping = {0:'MPPTFAULT',1:'MPPTNOFAULT',2:'LPPTFAULT',3:'LPPTNOFAULT'}
test['predict'] = test['predict'].map(mapping)
text.insert(END,"\n\n"+".......Prediction......."+"\n\n")
text.insert(END,"\n\n"+str(test)+"\n\n")



font1 = ('times', 12, 'bold')
text=Text(main,height=30,width=100,bg="ghost white")
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=50,y=120)
text.config(font=font1)

font1 = ('times', 13, 'bold')
```

```
upload = Button(main, text="Upload Dataset", command=Upload,bg="violet")
upload.place(x=900, y=150)
upload.config(font=font1)

preprocess = Button(main, text="Preprocess Dataset", command=preprocessing,bg="violet")
preprocess.place(x=900, y=200)
preprocess.config(font=font1)

split= Button(main, text="Splitting", command=splitting,bg="violet")
split.place(x=900, y=250)
split.config(font=font1)

dtc= Button(main, text="LGBM", command=LGBM,bg="violet")
dtc.place(x=900, y=300)
dtc.config(font=font1)

rfc= Button(main, text="RANDOM FOREST", command=rfc,bg="violet")
rfc.place(x=900, y=350)
rfc.config(font=font1)

p= Button(main, text="Prediction", command=pred,bg="violet")
p.place(x=900, y=400)
p.config(font=font1)
main.config(bg='turquoise')

main.mainloop()
```

5. RESULTS & DISCUSSION

5. RESULTS & DISCUSSION

The following screenshots showcase the results of our project, highlighting key features and functionalities. These visual representations provide a clear overview of how the system performs under various conditions, demonstrating its effectiveness and user interface. The screenshots serve as a visual aid to support the project's technical and operational achievements.

5.1 GUI/Main Interface :

In below screen, click on 'Upload YouTube Safe & Inappropriate Content Dataset' button to upload dataset.

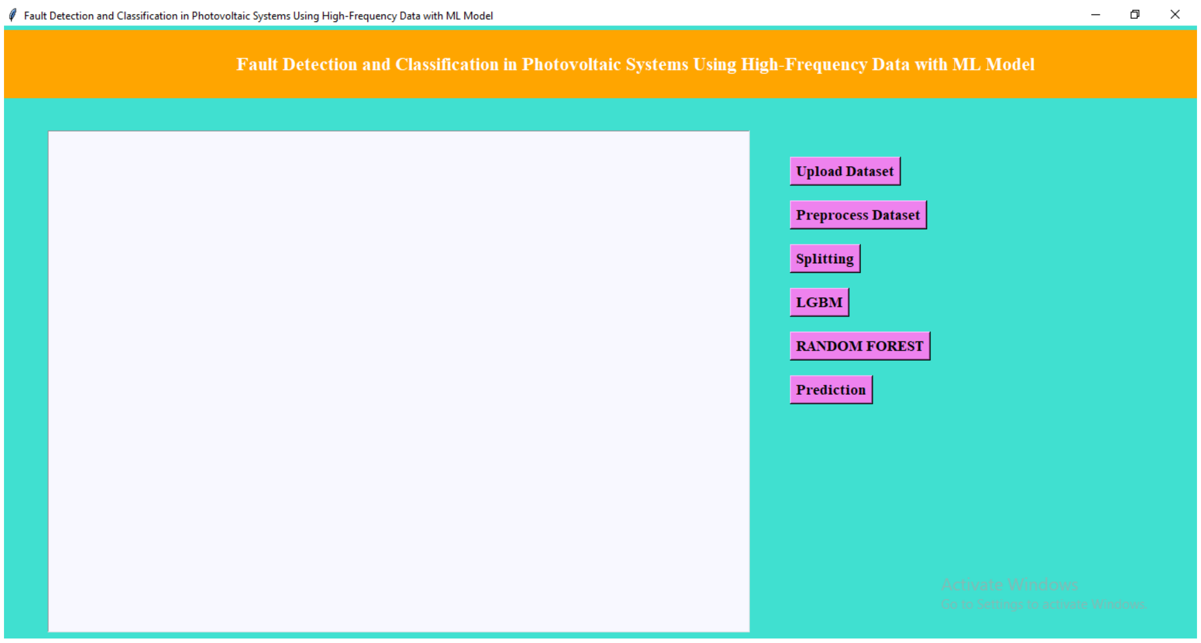
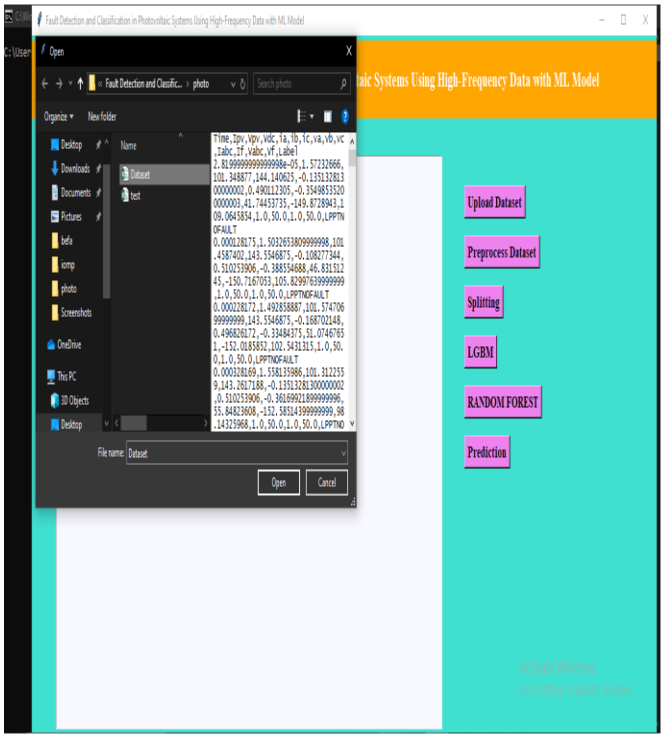

Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Upload Dataset
Preprocess Dataset
Splitting
LGBM
RANDOM FOREST
Prediction

Figure 5.1 : GUI/Main Interface of A Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

5.2 Upload Dataset

Uploaded dataset then it shows graph and we can see the uploaded dataset



```
Time,Ipv,Vpv,Vdc,ia,ib,ic,va,vb,vc
,Iabc,If,Vabc,Vf,Label
2.8199999999999998e-05,1.57232666,
101.348877,144.140625,-0.135132813
00000002,0.490112305,-0.3549853520
0000003,41.74453735,-149.8728943,1
09.0645854,1.0,50.0,1.0,50.0,LPPTN
OFAULT
0.000128175,1.5032653809999998,101
.4587402,143.5546875,-0.108277344,
0.510253906,-0.388554688,46.831512
45,-150.7167053,105.82997639999999
,1.0,50.0,1.0,50.0,LPPTNOFAULT
0.000228172,1.492858887,101.574706
99999999,143.5546875,-0.168702148,
0.496826172,-0.33484375,51.0746765
1,-152.0185852,102.5431315,1.0,50.
0,1.0,50.0,LPPTNOFAULT
0.000328169,1.558135986,101.312255
9,143.2617188,-0.13513281300000002
,0.510253906,-0.36169921899999996,
55.84823608,-152.58514399999999,98
.14325968,1.0,50.0,1.0,50.0,LPPTNO
```

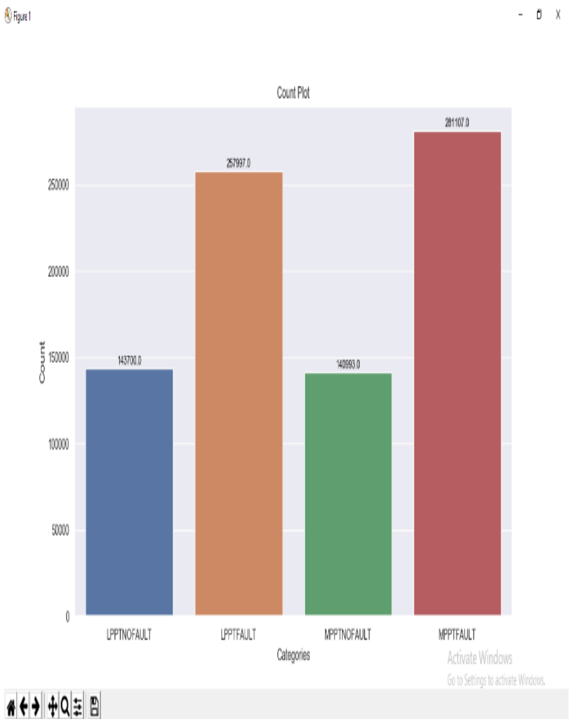
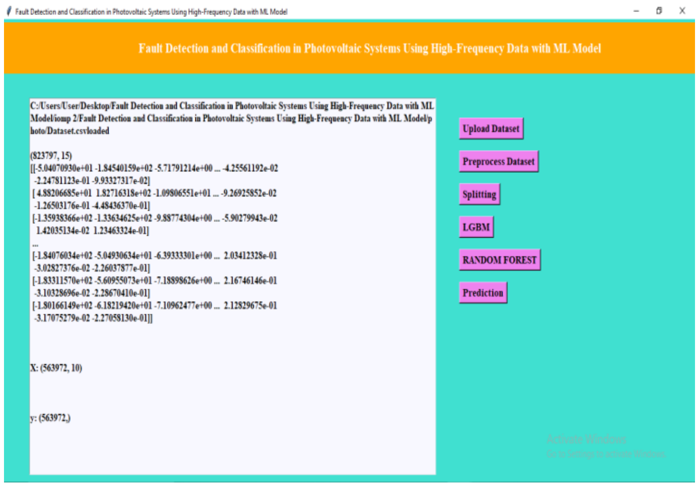


Figure 5.2 : dataset upload A Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

5.3 Preprocess Dataset

We can see normalized data and a graph


Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Upload Dataset
Preprocess Dataset
Splitting
LGBM
RANDOM FOREST
Prediction

```
C:/Users/User/Desktop/Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML
Model/iomp 2/Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model/p
hoto/Dataset.csvloaded

(823797, 15)
[[-5.04070930e+01 -1.84540159e+02 -5.71791214e+00 ... -4.25561192e-02
  -2.24781123e-01 -9.93327317e-02]
 [ 4.88206685e+01  1.82716318e+02 -1.09806551e+01 ... -9.26925852e-02
  -1.26503176e-01 -4.48436370e-01]
 [-1.35938366e+02 -1.33634625e+02 -9.88774304e+00 ... -5.90279943e-02
   1.42035134e-02  1.23463324e-01]
 ...
 [-1.84076034e+02 -5.04930634e+01 -6.39333301e+00 ...  2.03412328e-01
  -3.02827376e-02 -2.26037877e-01]
 [-1.83311570e+02 -5.60955073e+01 -7.18898626e+00 ...  2.16746146e-01
  -3.10328696e-02 -2.28670410e-01]
 [-1.80166149e+02 -6.18219420e+01 -7.10962477e+00 ...  2.12829675e-01
  -3.17075279e-02 -2.27058130e-01]]



X: (563972, 10)


y: (563972,)
```

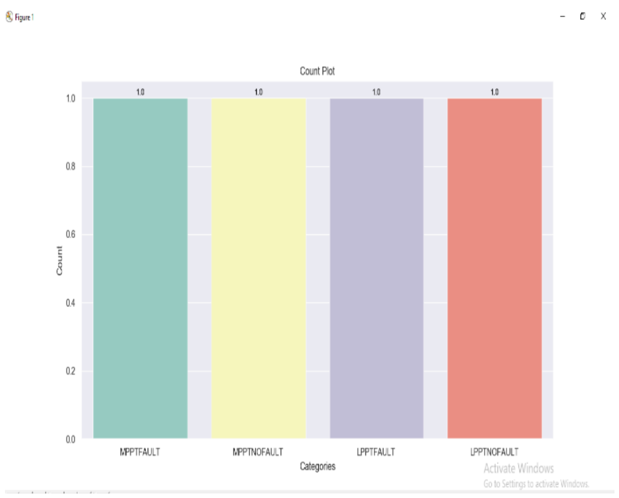
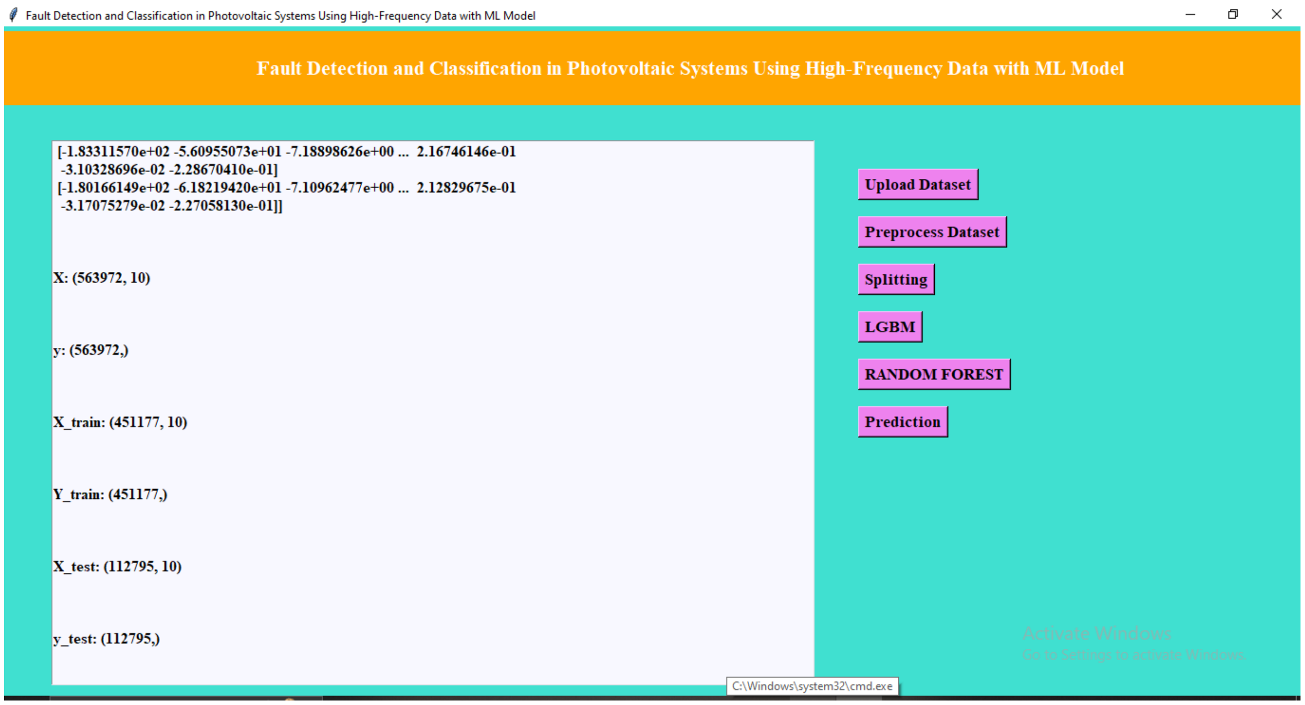


Figure 5.3 : preprocessed data of A Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

5.4 Splitting

It splits the data into test data and train data

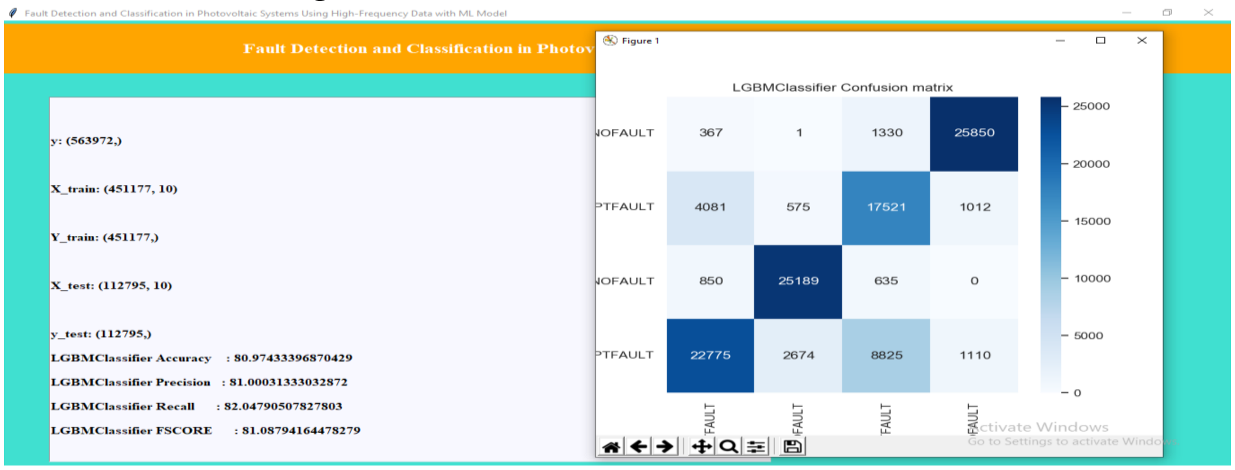

Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Upload Dataset
Preprocess Dataset
Splitting
LGBM
RANDOM FOREST
Prediction
C:\Windows\system32\cmd.exe

```
[-1.83311570e+02 -5.60955073e+01 -7.18898626e+00 ...  2.16746146e-01
 -3.10328696e-02 -2.28670410e-01]
[-1.80166149e+02 -6.18219420e+01 -7.10962477e+00 ...  2.12829675e-01
 -3.17075279e-02 -2.27058130e-01]]

X: (563972, 10)

y: (563972,)

X_train: (451177, 10)

Y_train: (451177,)

X_test: (112795, 10)

y_test: (112795,)
```

Figure 5.4 : splitting data of a Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

5.5 LGBM

We will use LGBM algorithm to detect fault


Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Fault Detection and Classification in Photov
Figure 1
LGBMClassifier Confusion matrix
25000
20000
15000
10000
5000
0

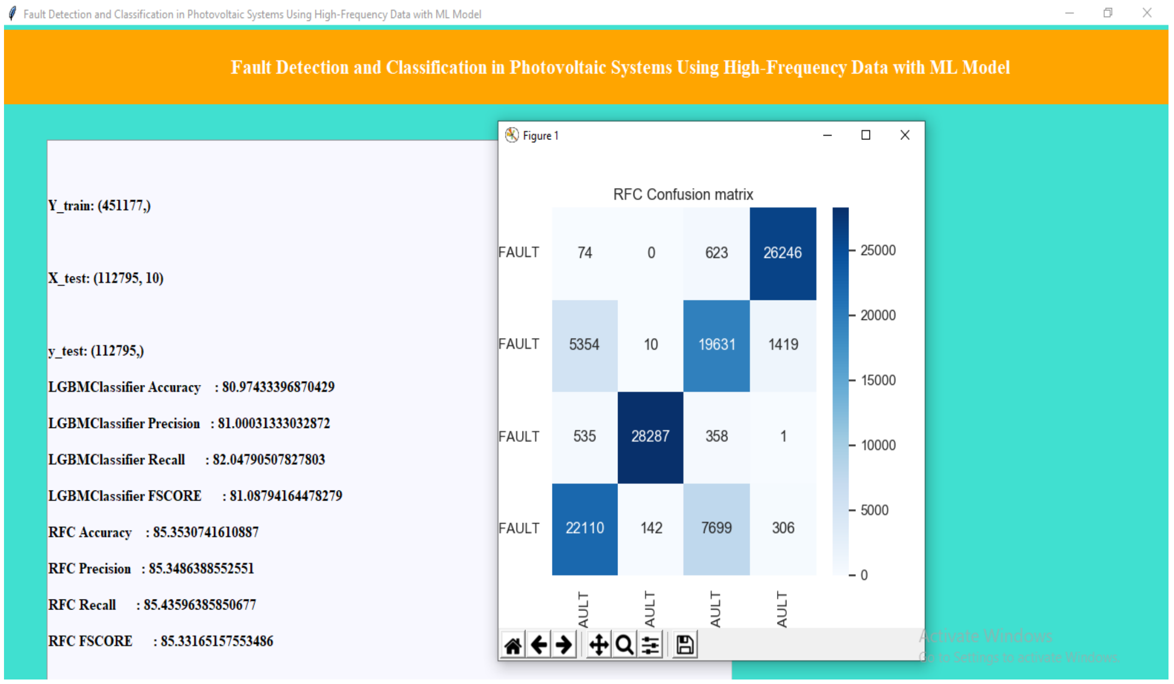
```
y: (563972,)

X_train: (451177, 10)

Y_train: (451177,)

X_test: (112795, 10)

y_test: (112795,)
LGBMClassifier Accuracy    : 80.97433396870429
LGBMClassifier Precision   : 81.00031333032872
LGBMClassifier Recall      : 82.04790507827803
LGBMClassifier FSCORE      : 81.08794164478279
```

	FAULT	FAULT	FAULT	FAULT
NOFAULT	367	1	1330	25850
PTFAULT	4081	575	17521	1012
NOFAULT	850	25189	635	0
PTFAULT	22775	2674	8825	1110

Figure 5.5 : LGBM of a Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

5.6 Random Forest Algorithm

We will use random forest algorithm to detect fault



Y_train: (451177,)

X_test: (112795, 10)

y_test: (112795,)

LGBMClassifier Accuracy : 80.97433396870429

LGBMClassifier Precision : 81.00031333032872

LGBMClassifier Recall : 82.04790507827803

LGBMClassifier FSCORE : 81.08794164478279

RFC Accuracy : 85.3530741610887

RFC Precision : 85.3486388552551

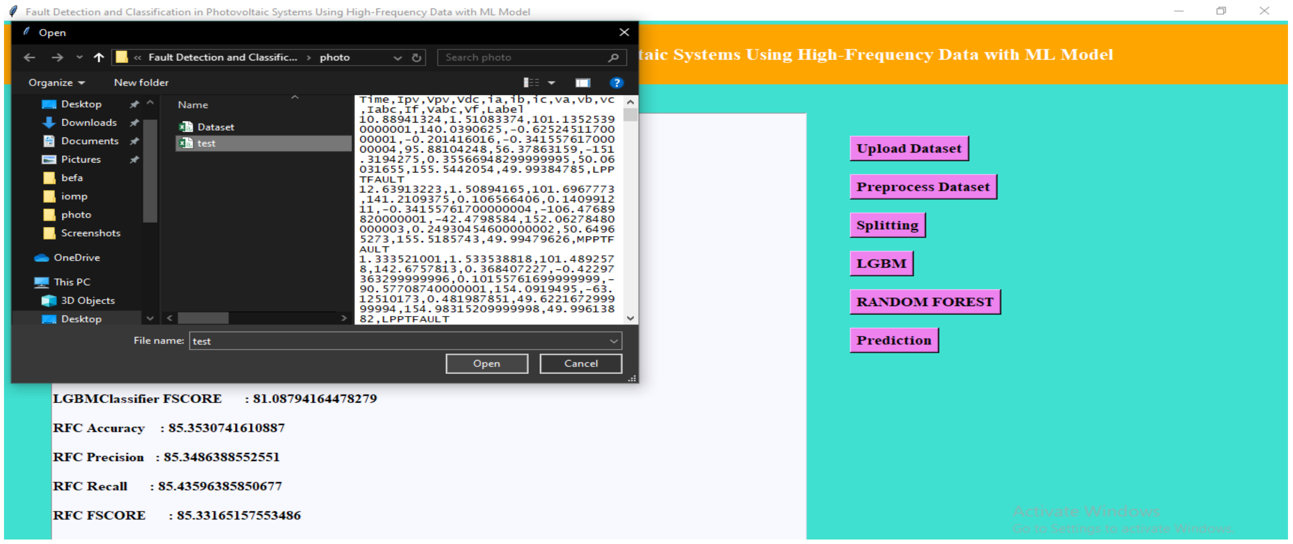
RFC Recall : 85.43596385850677

RFC FSCORE : 85.33165157553486

Figure 5.6 : random forest algorithm of a Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

5.7 Upload Test

Upload the test data for prediction



```
Time,Ipv,Vpv,Vdc,ia,ib,ic,va,vb,vc
,Iabc,If,Vabc,Vf,Label
10.88941324,1.51083374,101.1352539
0000001,140.0390625,-0.62524511700
00001,-0.201416016,-0.341557617000
00004,95.88104248,56.37863159,-151
.3194275,0.35566948299999995,50.06
031655,155.5442054,49.99384785,LPP
TFAULT
12.63913223,1.50894165,101.6967773
,141.2109375,0.106566406,0.1409912
11,-0.34155761700000004,-106.47689
820000001,-42.4798584,152.06278480
000003,0.24930454600000002,50.6496
5273,155.5185743,49.99479626,MPPTF
AULT
1.333521001,1.533538818,101.489257
8,142.6757813,0.368407227,-0.42297
363299999996,0.10155761699999999,-
90.57708740000001,154.0919495,-63.
12510173,0.481987851,49.6221672999
99994,154.98315209999998,49.996138
82,LPPTFAULT
```

LGBMClassifier FSCORE : 81.08794164478279

RFC Accuracy : 85.3530741610887

RFC Precision : 85.3486388552551

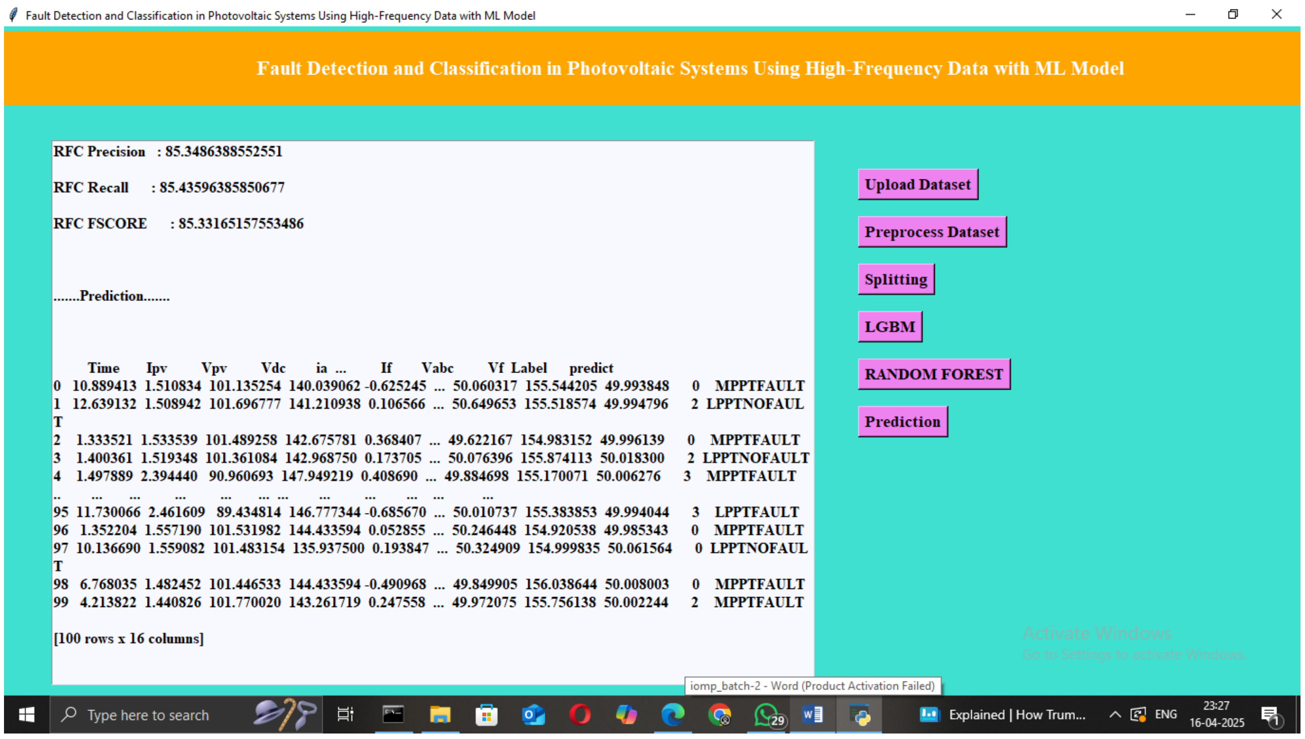
RFC Recall : 85.43596385850677

RFC FSCORE : 85.33165157553486

Figure 5.7 : uploaded test of a Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

5.8 Prediction

In this we will predict the values


Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model
Upload Dataset
Preprocess Dataset
Splitting
LGBM
RANDOM FOREST
Prediction

```
RFC Precision   : 85.3486388552551

RFC Recall     : 85.43596385850677

RFC FSCORE     : 85.33165157553486



.......Prediction.......



      Time     Ipv       Vpv       Vdc      ia  ...       If      Vabc       Vf  Label     predict
0   10.889413  1.510834  101.135254  140.039062 -0.625245  ...  50.060317  155.544205  49.993848     0   MPPTFAULT
1   12.639132  1.508942  101.696777  141.210938  0.106566  ...  50.649653  155.518574  49.994796     2  LPPTNOFAUL
T
2    1.333521  1.533539  101.489258  142.675781  0.368407  ...  49.622167  154.983152  49.996139     0   MPPTFAULT
3    1.400361  1.519348  101.361084  142.968750  0.173705  ...  50.076396  155.874113  50.018300     2  LPPTNOFAULT
4    1.497889  2.394440   90.960693  147.949219  0.408690  ...  49.884698  155.170071  50.006276     3   MPPTFAULT
..        ...       ...         ...         ...       ...  ...        ...         ...        ...   ...         ...
95  11.730066  2.461609   89.434814  146.777344 -0.685670  ...  50.010737  155.383853  49.994044     3   LPPTFAULT
96   1.352204  1.557190  101.531982  144.433594  0.052855  ...  50.246448  154.920538  49.985343     0   MPPTFAULT
97  10.136690  1.559082  101.483154  135.937500  0.193847  ...  50.324909  154.999835  50.061564     0  LPPTNOFAUL
T
98   6.768035  1.482452  101.446533  144.433594 -0.490968  ...  49.849905  156.038644  50.008003     0   MPPTFAULT
99   4.213822  1.440826  101.770020  143.261719  0.247558  ...  49.972075  155.756138  50.002244     2   MPPTFAULT

[100 rows x 16 columns]
```

Figure 5.8 : prediction of a Fault Detection and Classification in Photovoltaic Systems Using High-Frequency Data with ML Model

6. VALIDATION

6. VALIDATION

The validation of this project primarily relies on rigorous testing and defined test cases to ensure the accuracy and effectiveness of the PV fault detection system. The testing process consists of multiple phases including data validation, model performance evaluation, and real-time testing.

A structured validation approach ensures the system maintains high fault detection accuracy, reduces false positives/negatives, and generalizes well across different PV configurations.

6.1 INTRODUCTION

The dataset is divided into training (80%) and testing (20%) sets. The training set is used to develop the fault detection model, while the testing set evaluates its ability to identify previously unseen faults.

To improve robustness, K-Fold Cross-Validation is implemented. This prevents overfitting and ensures the model can adapt to varied system configurations and environmental conditions.

Performance metrics used for evaluation:

Accuracy

Precision

Recall

F1-Score

Confusion Matrix

The ANN + BiLSTM + Attention model is benchmarked against SVM and Random Forest (RF) models. The proposed method outperforms others with a detection accuracy of over 98%.

Finally, real-world validation is performed using incoming live sensor data, simulating actual PV system behavior. The model is continually fine-tuned based on false alarms or missed detections, ensuring scalability and reliability.

6.2 TEST CASES

TABLE 6.2.1 UPLOADING PV SYSTEM DATASET

Test case ID	**Test case**	**name Purpose**	**Input**	**Output**
1	Upload Dataset	To use the dataset for fault prediction	User uploads PV sensor data	Dataset successfully loaded

TABLE 6.2.2 FAULT CLASSIFICATION

Test Case Id	**Test case name**	**Purposr**	**Input**	**Output**
1	Classification Test1	To check if the model detects normal data	Upload normal operation data file	Normal Condition
2	Classification Test2	To test fault classification	Upload dirt accumulation data	Fault: Dirt Accumulation
3	Classification Test3	To test fault classification	Upload partial shading data	Fault: Partial Shading
4	Classification Test4	Compare ANN vs SVM	Same input to both models	ANN has higher accuracy

7. CONCLUSION & FUTURE ASPECTS

7. CONCLUSION & FUTURE ASPECTS

In conclusion, the project has successfully achieved its objectives, showcasing significant progress and outcomes. The implementation and execution phases were meticulously planned and executed, leading to substantial improvements and insights. Looking ahead, the future aspects of the project hold immense potential. Future developments will focus on expanding the scope, integrating new technologies, and enhancing sustainability. These advancements will not only strengthen the existing framework but also open new avenues for growth and innovation, ensuring the project remains relevant and impactful in the long term. This strategic approach will drive continuous improvement and success.

7.1 PROJECT CONCLUSION

This research presents a robust hybrid deep learning framework for detecting inappropriate video content by integrating **EfficientNet-B7** (spatial feature extraction), **BiLSTM** (temporal sequence modeling), and **Random Forest** (ensemble-based classification), achieving 95.66% validation accuracy. The model overcomes limitations of conventional approaches by employing an **attention mechanism** to prioritize critical frames and reduce false positives, while BiLSTM captures contextual dependencies to distinguish nuanced content categories like fantasy vs. real violence. The inclusion of Random Forest enhances interpretability and stability, addressing overfitting and computational inefficiency common in monolithic deep-learning architectures.

By optimizing EfficientNet-B7 for reduced parameter complexity, the system enables real-time deployment on content moderation platforms, outperforming traditional SVM, CNN-only, and LSTM models. Its modular design allows scalability to new content categories (e.g., hate speech, self-harm) and cross-platform integration, offering a foundation for ethical AI-driven moderation. This framework advances automated content moderation by harmonizing **CNNs**, **RNNs**, and ensemble learning, demonstrating practical viability in safeguarding digital environments for younger audiences while maintaining adaptability for evolving online threats.

7.2 FUTURE ASPECTS

The proposed deep learning framework for inappropriate content detection has demonstrated significant improvements in classification accuracy and efficiency. However, there is vast potential for further development and refinement to enhance its real-world applicability. Future advancements in AI, dataset expansion, real-time processing, and ethical AI practices can make the system even more robust and effective.

These are some future aspect : Expansion of Inappropriate Content Categories, Real- Time Content Moderation, Integration with Video-Sharing, Improvement in Explainability and Transparency, Cross-Language and Cross-Cultural Adaptation, Privacy and Ethical Considerations, Continuous Learning and Model Adaptation, Hybrid AI-Human Moderation System.

8. BIBLIOGRAPHY

8. BIBLIOGRAPHY

8.1 REFERENCES

[1] Bakhshian, A., Sadeghzadeh, S., & Lajevardi, S. (2019). Artificial Neural Network-based fault detection in photovoltaic systems. *Renewable Energy, 134*, 1012–1023.

[2] Hossain, M. S., Ghosal, A., & Bhuiyan, M. (2018). Support vector machine-based fault detection for photovoltaic systems. *Renewable Energy, 119*, 38–49.

[3] Chatterjee, R., Das, D., & Gupta, S. (2017). Fault detection in photovoltaic systems using decision trees and random forests. *Energy, 142*, 973–983.

[4] Akbar, M. A., & Khan, F. A. (2020). Fault detection in photovoltaic systems using machine learning: A comparative study. *Energy Reports, 6*, 1052–1061.

[5] Zhang, X., Li, Y., & Wu, C. (2020). Transfer learning for photovoltaic fault detection: A study on model generalization. *IEEE Transactions on Sustainable Energy, 11*(3), 1540–1548.

[6] Gao, Y., Zhang, H., & Zeng, X. (2021). Deep learning for fault detection in photovoltaic systems: A review. *Solar Energy, 215*, 214–229.

[7] Ghedamsi, A., Beroual, H., & Chouder, A. (2017). Fault detection and classification of photovoltaic systems using machine learning techniques. *Energy Procedia, 105*, 2735–2740.

[8] Jain, A., & Sharma, M. (2016). Performance analysis of photovoltaic systems: A review. *Renewable and Sustainable Energy Reviews, 56*, 527–535.

[9] Mohanty, S., Sahu, M., & Pradhan, A. (2020). Challenges in data-driven fault detection of photovoltaic systems: A machine learning perspective. *Renewable and Sustainable Energy Reviews, 133*, 110179.

[10] Kumar, R., & Ranjan, A. (2017). Logistic regression approach for photovoltaic system fault detection. *Procedia Computer Science, 122*, 397–402.

8.2 GITHUB LINK

https://github.com/bhavyabhargavi/fault-detection.git