POWER TRANSFORMER DIAGNOSTIC MODEL USING MACHINE LEARNING ALGORITHMS

BY

EGBON OSAYOMORE DESIRE
PSC1808806

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF PHYSICAL SCIENCES
UNIVERSITY OF BENIN

SEPTEMBER 2023

POWER TRANSFORMER DIAGNOSTIC MODEL USING MACHINE LEARNING ALGORITHMS

BY

EGBON OSAYOMORE DESIRE
PSC1808806

A PTOJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY IN PARTIAL FULFILLMENT OF THE REQUIREMNTS FOR AWARD OF BACHELOR OF SCIENCE DESGREE (B.Sc.) IN COMPUTER SCIENCE.

SEPTEMBER 2023

ATTESTATION

We hereby declare that this project **POWER TRANSFORMER FAULT DIAGNOSIS USING MACHINE LEARNING TECHNIQUES** was carried out by **EGBON OSAYOMORE DESIRE with Mat No. PSC1808806** and it is a record of our project work in the Department of Computer Sciences, Benin City, In partial fulfillment of a Bachelor of Science in Computer Science degree. It has not been presented before in any previous application for a bachelor's degree. References made to published literature have been duly acknowledged

APPROVAL

This project work carried out by **EGBON OSAYOMORE DESIRE** with matriculation number **PSC1808806** in partial fulfillment of the requirement for the award of the University of Benin Bachelor of Science (B. Sc.) degree in Computer Science, is adequate both in scope and content and it is hereby approved for presentation.

_____ _____

PROF. (MRS) A. O. EGWALI **DATE**
HEAD OF DEPARTMENT (HOD)

CERTIFICATION

This is to certify that **EGBON OSAYOMORE DESIRE** carried out this project work **POWER TRANSFORMER FAULT DIAGNOSIS USING MACHINE LEARNING TECNIQUES** for the award of B.Sc. with matriculation number **MAT NO: PSC1808806** Faculty of Physical Sciences, Department of Computer Science, University of Benin, has been accepted for defense.

_____ _____

E.C. IGODAN (PhD)
PROJECT SUPERVISOR **DATE**

DEDICATION

This project work is dedicated to God Almighty, for providence, guidance and grace in seeing me through this study; I give Him all the glory.

ACKNOWLEDGEMENT

Firstly I will like to thank God for his grace and mercies upon my life that endures forever more and also appreciate this great department for the privilege to study in this prestigious university and department of Computer Science.

I would like to thank and appreciate my esteemed supervisor assigned to me during the course of this project **MR. E.C. IGODAN (PhD).** My deepest appreciation for his mentorship, teachings and guidance during the course of this project. I am grateful sir and I pray God bless and prosper you. Thank you sir.

I also wish to appreciate the Head of Department of Computer Science, Prof. (Mrs.) A. O. Egwali for all her support and efforts in upholding the department to a reputable standard, I would like to extend my appreciation as well to other lecturers in the department, Prof. Godspower O. Ekhuobase, PHD, Dr. (Mrs.) G. O. Aziken, Mr. D. N Idehen, Mr. K. O. Otokiti, Dr. F. O. Oliha, Mr. S. O. P. Oliomogbe. For their selfless service to myself and other students of the department.

I would like to deeply appreciate my Family whom sacrificed a lot to take me through the university, my honest gratitude to my parents Mr. Teddy Victor Egbon and Mrs. Evelyn Egbon and my siblings Nature Egbon and Legacy Egbon. I would also like to extend my appreciation to Mrs. Doreen Imasuen, Mr. Etinosa Omoregie, Mrs. Itohan John, Mr. Ikponwonsa Omoregie, Mr. Rapheal Omoregie, Mrs. Felicia Omoregie, Mrs. Omoregie Mabel, Mr. Nosakhare Imasuen, Ms. Omatsuli Oduwa for their love and support.

Lastly I would like to appreciate my fellow colleagues of whom I started this journey with thank you all for your support over the course of study. I wish you all the best in all endeavors of life.

ABSTRACT

The use of machine learning techniques in the process of power transformer fault diagnosis represents a significant advancement in technology. This abstract of this project provides an overview of the key features of this work, displaying the profound benefits of machine learning in power transformer fault detection and prediction.

In recent years experiment have been conducted in this area using traditional methods like the Key Gas Method, Duval triangle method and others and utilizing the data gotten from DGA (Dissolved Gas Analysis) to detect faults in the power transformers. These traditional methods had a draw back of poor accuracy in detection and fault prediction. In this project machine learning techniques like the Naive Bayes, Support vector Machine, C4.5 Decision Tree and their ensembles also included in this is the Majority voting were introduced to solve the problems.

In this project a model is created to train and test data using this machine learning techniques. The results gotten from these experiments show that the machine learning techniques presents a brighter future for professionals and experts in this field of work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Background Study

The power transformer is a very ideal aspect of power generation in our world today, it uses electrical and mechanical functionalities to operate.A power transformer is a fixed electrical apparatus employed to transfer electrical energy between varying voltage levels, commonly from a higher voltage to a lower voltage or vice versa. It holds significant importance within electrical power systems as it enables the efficient transmission and distribution of electricity. A Transformer consists of several gases such as the H_2, CH_4, C_2H_6, C_2H_4, C_2H_2, CO_2, CO and N_2. (M. Demirci et *al, 2021*). which are dissolved in the oil from the transformer and using this gases found in the oil, to effectively detect faults. This gases diagnosed from the oil shows the age faults, electrical discharge and heat generation the main causes of power transformer faults related gases are: overheating, and Arcing which leads to degradation (Hazlee Azil Illias et *al, 2020*). These disturbances can result in the appearance of gases and molecules that are difficult to dissolve in the transformer oil. In order to detect these conditions researchers have developed methods for detecting errors, one of the techniques for detecting these disturbances is Dissolved Gas Analysis (DGA). The procedure for this method is carried out by removing dissolved gas from the sample oil and then determining the gas component content .(Rosmaliati *et al, 2022)*.

These Power transformers faults can lead to costly downtime, safety hazards, and significant financial losses. In recent years, machine learning has emerged as a game-changer in the field of fault diagnosis, revolutionizing how power transformers are monitored, diagnosed, and maintained. By using advanced algorithms and techniques, machine learning models have significantly enhanced the accuracy, efficiency, and effectiveness of diagnosing faults in power transformers. Machine learning models have the capability to offer advance indications of possible faults. Through ongoing monitoring and analysis of real-time data, these models can detect slight alterations and anomalies in the behavior of transformers, indicating the initial phases of a fault. Early detection of faults allows for prompt intervention and preventive maintenance,

effectively preventing severe failures, minimizing downtime, and prolonging the operational lifespan of power transformers

This study proposes a more accurate and more robust solution by using the data from the sample datasets of DGA to develop an improved system by the means of Support Vector Machine (SVM), Naive Bayes (NB) and C4.5 Decision tree algorithms and their ensembles using Bagging, boosting and Majority Voting techniques.

1.2 Statement of the Problem

In Nigeria and other developing countries, the challenge with detection of faults in a power transformer as a result of the present use of traditional methods like the Key Gas Method (KGM) and the Rogers' Method resulted to the unexpected power outages, aging facilities, economic and financial losses which are inimical to the development of a fully functional transformer.

1.3 Aim and Objectives

The aim of this project is to design an improved predictive model for the prediction of fault in power transformer. The following objectives in achieving our aim are:

a). Extract relevant DGA features from power transformer;

b). Design a machine learning model for fault prediction using boosting, bagging and majority voting ensemble classifiers with SVM, Naive Bayes and C4.5 Decision tree;

c). Implement the ensemble models using python programming language; and

d). Evaluate the performance of the ensemble model in fault prediction based on accuracy and confusion matrix.

1.4 Methodology of the Study

In the success of this project the following methods were applied, The proposed model utilizes an available dataset, the IEC TC10 dataset obtained from an existing literature. The methodology of the study consists of the following sequential steps:

Firstly, the dataset undergoes median imputation, tackling any missing values, and then, the data is subjected to min-max normalization, effectively standardizing its range.

The next step is the use of the following techniques Naive Bayes (NB), support vector machines (SVM), and the C4.5 Decision tree algorithms are used to classify the preprocessed dataset obtained from the previous step.

Thirdly, the trained models are combined using ensemble aggregation techniques such as bagging, boosting, and Majority voting These methods are then compared to attain the most effective approach in terms of accuracy and performance.

In the final step, the performance of the ensemble method is evaluated using appropriate evaluation metrics, such as accuracy, precision, recall, Mean Absolute Error (MAE) or F1-score. The results of the ensemble method are compared with those of the other intelligent methods to properly show the improvement in their results.

1.5 The Scope of Study

The scope of this study involves Dissolved Gas Analysis (DGA), Naive Bayes and Support Vector Machine (SVM) using the C4.5 Decision tree algorithm.

1.6 The significance of study

This study aims to investigate the use of Dissolved Gas Analysis and compare the effectiveness of different machine learning algorithms for power transformer fault diagnosis. The objective is to develop an accurate and reliable model that can identify and classify different fault types in power transformers. The study will focus on ensemble methods, Support Vector Machine (SVM), Naive Bayes, and C4.5 Decision Tree algorithms. The findings from this study will contribute to the development of accurate and efficient fault diagnosis systems, leading to enhanced maintenance and improved reliability of power transformer operations.

CHAPTER 2
LITERATURE REVIEW

2. 1. Overview

This chapter presents an overview of the principles related to power transformer fault diagnosis using Dissolved Gas Analysis (DGA), Naive Bayes and Support Vector Machine (SVM) using the C4.5 Decision tree algorithm. The subsequent sections delve into the application of machine learning techniques for the categorization and ensemble of power transformer faults, specifically focusing on DGA. A tabular format is used to illustrate the related studies conducted in this field. Building upon the previous explanations, it is evident that employing machine learning in power transformer fault diagnosis holds significant value.

A comprehensive examination of the relevant literature is presented in this section, which encompasses various traditional techniques with respect to the DGA data. The classification algorithm internally comprises three stages. The initial stage involves preprocessing the features extracted from the DGA data obtained from the power transformers. This is followed by feature selection approaches in the second stage. Finally, the third stage concentrates on utilizing machine learning and ensemble classifiers to effectively categorize unknown instances of power transformer faults using functional genomics. The subsequent sections of this chapter provides an introduction to the fundamentals of power transformers, fault diagnosis, and DGA, including symptoms, causes, and types of transformer faults.

2.2 Introduction to Power Transformer

Power transformers are essential components in electrical power transmission and distribution systems, where they facilitate the efficient and reliable transmission of electrical energy over long distances. The power transformer is an essential component in power systems, enabling voltage adjustment and distribution of electric energy. Due to its prolonged operation under load, power transformers have a higher probability of experiencing failures compared to other power equipment. If a transformer fault is not promptly diagnosed and repaired, it can lead to a cascading effect within a specific section of the power grid. Hence, it is crucial to conduct

regular detection and diagnosis of transformer faults as a preventive measure. This practice helps power grid personnel identify and address potential faults before they manifest. By implementing daily fault detection and diagnosis procedures, power grid staff can proactively undertake repairs, minimizing the occurrence of transformer faults and preventing their adverse impact on the power grid.(Xiaoli Huang *et al 2023*) In this section we will discuss on various components, faults and failures of a power transformer.

2.3 Components of a Power Transformer

In this section we will be discussing on the various components that make up the power transformer. A power transformer is composed of various essential components that work together to carry out the efficient transformation of electrical energy. This components play different roles in the overall delivery of the system.

Every Power Transformer is made up components like the windings, core, cooling systems, bushings, transformer tank, Tap changers, insulation. By combining these components, a power transformer efficiently transfers electrical energy, facilitates voltage transformation, and ensures safe and reliable operation within specified limits.

Figure 2.1 Power distribution transformer (https://www.cnrockwill.com/distribution-transformer)

a **Windings**

The primary and secondary windings are constructed using high-quality insulated copper or aluminum conductors. The number of turns in each winding determines the voltage ratio and power transformation capability of the transformer. The effectiveness of the windings lies in their ability to efficiently transfer electrical energy with minimal losses and withstand thermal and mechanical stresses during operation.

b Core:

The core, typically made of laminated silicon steel, provides a low-reluctance magnetic path, allowing for efficient flux transfer between the primary and secondary windings. Its design and construction minimize energy losses due to hysteresis and eddy currents, ensuring high magnetic efficiency and reducing core losses.

c Insulation:

Insulation materials, such as paper, varnish, and synthetic materials, are carefully selected to electrically isolate the windings and core. Effective insulation prevents electrical breakdowns, short circuits, and enhances the transformer's overall safety and reliability. It also ensures the insulation system's longevity and ability to withstand high electrical stresses.

d Transformer Tank:

The transformer tank is a sturdy enclosure that houses the core, windings, and other internal components. Its effectiveness lies in providing mechanical protection, preventing moisture ingress, and containing the transformer oil. The tank also acts as a heat sink, dissipating generated heat to maintain optimal operating temperatures.

e Cooling System:

The cooling system, which can be natural convection or forced cooling, ensures effective heat dissipation. By removing excess heat, it prevents overheating of the windings and core, maintaining the transformer's efficiency and extending its operational life.

f Tap Changer: The tap changer, if present, allows the adjustment of the turns ratio to regulate output voltage or compensate for variations in the input voltage. An

effective tap changer enables optimal voltage control and ensures the transformer's compatibility with different operating conditions.

e Bushings:

Bushings provide reliable and insulated connections for the transformer windings, allowing for safe electrical connections to the external circuitry. Their effectiveness lies in their ability to withstand high voltage stresses and provide a reliable interface for power transfer.

2.4 Common faults in a Power Transformer

Faults are the abnormal conditions, malfunction, failure, or flaws that disrupts the normal functioning and performance of the equipment, potentially leading to degraded system performance, equipment damage, or complete system failure**.** In order to assess and diagnose faults in a transformer, an important parameter used is the ratio of gases present in the oil chromatography data obtained through Dissolved Gas Analysis (DGA). Specifically, the evaluation index for fault judgment focuses on the proportions of gases such as H2, CH4, C2H6, C2H4, C2H2, and other hydrocarbons in relation to the total hydrocarbon content. This ratio serves as an assessment parameter for fault determination. Transformer faults are categorized into six types: normal operation, low energy discharge, high energy discharge, partial discharge, medium and low temperature overheating, and high temperature overheating. .(Xiaoli Huang *et al 2023*). These faults are further explained below

a. Low Energy Discharge (D1):

It refers to a fault condition characterized by the release of a low amount of energy, resulting in the generation of specific gas ratios in the oil.

b High Energy Discharge (D2):

In this fault type, a higher level of energy discharge occurs within the transformer, leading to distinctive gas ratios detected through DGA.

c Partial Discharge (PD):

This fault category signifies the occurrence of localized electrical discharges within the transformer, resulting in distinct gas ratios indicative of the fault.

d. Medium (T1) and Low Temperature Overheating (T2):

This type of fault refers to the condition where the transformer operates at higher

temperatures than normal but still within a moderate range. The associated gas ratios obtained from DGA help in identifying this fault.

e. High Temperature Overheating (T1):

This fault represents a severe condition where the transformer operates at significantly elevated temperatures, surpassing the acceptable limits. The gas ratios derived from DGA provide crucial information to detect this fault type

2.5 Datasets

These are the structured data that serve as the foundation for statistical analysis, machine learning model training, and predictive modeling. These set of data are gotten from the raw DGA from the power transformer system and are used for training and testing processes (Ibrahim *et al 2021*)

2.5.1 Dissolved Gas Analysis (DGA)

This is a technique used to assess the condition and health of power transformers and other oil-filled electrical equipment. It involves analyzing the gases dissolved in the insulating oil of the transformer. The presence and concentration levels of certain gases can provide valuable insights into the types of faults or abnormalities occurring within the Power Transformer. The gases found inside a power transformer include

1. Hydrogen gas (H_2)
2. Methane gas(CH_4)
3. Acetylene gas (C_2H_4)
4. Ethane gas (C_2H_6)
5. Ethylene gas (C_2H_4)
6. Carbon monoxide gas (CO)

7 Carbon dioxide gas (CO_2)

8. Nitrogen gas (N_2)

When electrical faults occur within a transformer, such as overheating, arcing, or insulation degradation, various gases are produced as byproducts. These gases dissolve in the transformer oil, and their concentrations can indicate the specific fault types and their severity. DGA is performed by extracting a sample of the insulating oil from the transformer and analyzing it by using methods like the Key Gas Method, Roger method or the IEC Ratio Method then the dissolved gases are measured and

quantified using gas chromatography analytical technique.
(Electrical Engineering & Electromechanics, 2021)

2.6 Introduction To Machine Learning

Machine learning (ML) is a field of research focused on developing techniques that enable computers to learn and improve their performance on tasks by utilizing data. ML algorithms have the ability to complete tasks without explicit programming, as they learn from available data. By learning from data, computers can perform specific tasks effectively. While it is possible to create algorithms that explicitly instruct computers to perform all steps for simpler tasks, more complex tasks pose challenges in manual algorithm creation for humans. In such cases, it becomes beneficial to assist the computer in developing its own algorithm through learning from data, rather than relying on human programmers to define every step.

machine learning (ML) plays a crucial role in leveraging available data to enhance performance. ML algorithms have the capability to autonomously complete tasks without explicit programming instructions. By learning from data, computers can effectively diagnose faults in various systems. For simpler tasks in fault diagnosis, it is feasible to develop algorithms that explicitly guide computers on how to solve the problem at hand, requiring no learning on the computer's part. However, for more complex fault diagnosis scenarios, manually constructing the necessary algorithms can be challenging for humans. In such cases, it becomes more advantageous to assist the computer in developing its own algorithm through ML techniques, allowing it to learn from available data.

By training ML algorithms on relevant fault data, computers can autonomously learn patterns and relationships, leading to more accurate and efficient fault diagnosis. Instead of human programmers defining each step in the algorithm, ML algorithms can extract valuable insights from the data and adapt their diagnostic approaches accordingly. This enables the computer to effectively diagnose faults more easily and accurately.

Machine learning can be classified into three main types: supervised learning, unsupervised learning, and reinforcement learning.

Supervised Learning: Supervised learning involves training a model using labeled data, where each training instance consists of input features and corresponding target labels. The goal is for the model to learn a mapping function that can accurately predict the labels for new, unseen instances. Supervised learning can be categorized into classification and regression

Unsupervised Learning: Unsupervised learning deals with unlabeled data, where the model aims to find patterns, structures, or relationships within the data without explicit target labels. The goal is often to discover inherent groupings or similarities in the data clustering and dimensionality reduction are types of. Unsupervised learning techniques

Reinforcement learning: This type of machine learning deals with a situation where an learner learns by interacting with its environment to earn rewards. The learner takes actions based on its current situation and receives feedback in the form of rewards. The objective is for the agent to learn the best strategy over time to maximize the total rewards it can achieve. This approach is commonly used in dynamic and complex situations, like playing games, robotics, or autonomous driving, where the agent needs to make a series of decisions.

2.6.1 Feature Selection

This involves the selection of the most relevant and important values from a dataset so as to carry out a task effectively. Feature selection reduces the dimensionality of the dataset, which prevents models from learning noise and promotes better generalization to unseen data. Feature selection offers transparency between relevant data and target variable. With feature selection classification performance, accuracy and generalization is increased. Machine learning feature selection strategies may be categorized into the following categories:

Supervised Models:

Supervised feature selection is a subset of feature selection methods in which the target variable and can be used for the labeled dataset.

Unsupervised Models:

Unsupervised Feature selection techniques disregards the target variable or dependent variable and can be used for the unlabeled dataset.

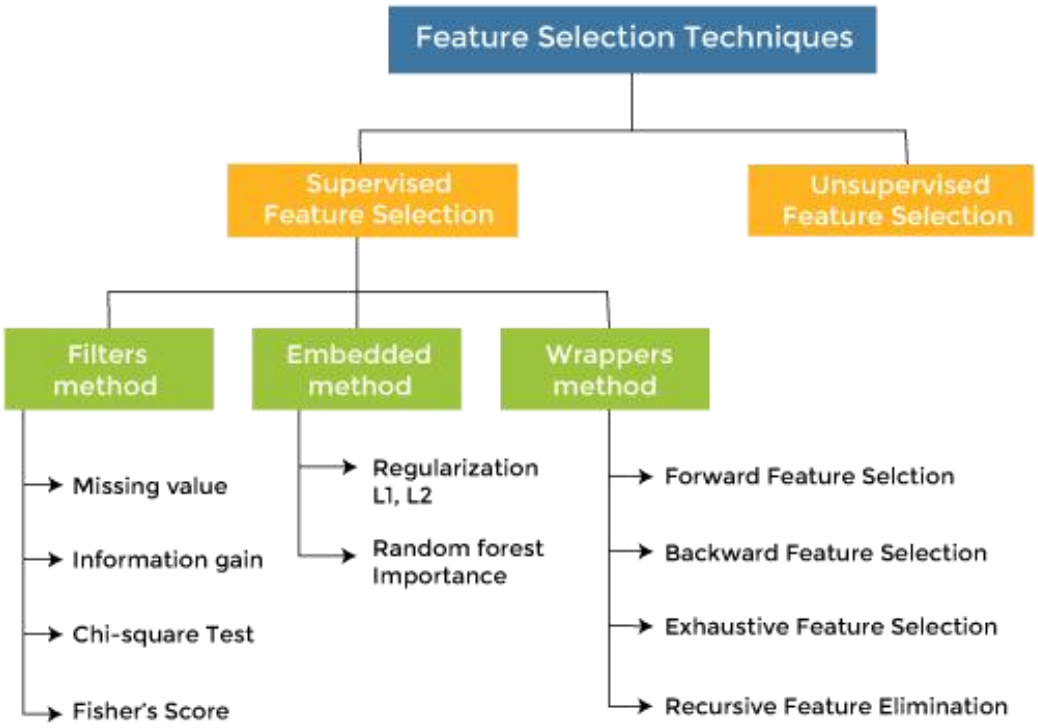

Figure 2.2 Feature selection chart (www.javatpoint.com)

these supervised and unsupervised methods can be categorized further into the following:

Filter Methods: Filter methods assess the relevance of features based on their intrinsic characteristics, such as statistical measures or correlation with the target variable. These methods evaluate features independently of any specific learning algorithm. Common filter methods include correlation-based feature selection, information gain, chi-square test, and variance threshold.

Wrapper Methods: Wrapper methods evaluate feature subsets by using a specific learning algorithm to assess their performance. These methods employ a search algorithm, such as forward selection, backward elimination, or recursive feature elimination, to build and evaluate different feature subsets. Wrapper methods consider the predictive performance of the learning algorithm to select the optimal subset of features.

Embedded Methods: Embedded methods incorporate feature selection as an integral part of the learning algorithm itself. These methods perform feature selection during the model training process by considering the interaction between feature selection and model performance.

2.6.2 Ensemble Methods

Ensemble method is a machine learning technique that combines several base models in order to produce one optimal predictive model, this is done by combining the weaker models or models that may actively function alone, but collectively combining a couple of weak or base models using each of their individual strengths, it establishes an improved and high performance model.

Ensemble is aimed at utilizing the strengths of each weak model, by combining each of them to yield a more accurate and reliable model to produce better results. This models are called weak because they either have a high bias or high variance hence cannot learn efficiently thus performing poorly.

The following are approaches for creating an ensemble model

1. Bagging: Bagging which is stands for Bootstrapped Aggregation, this is a technique used to used to reduce the variance of weak learners by combining weak learners of high variance. In Bagging we have the Bootstrap which deals with the resampling of subsets of the primary datasets with replacement to train the weak learners, and aggregation involves the individual training of the weak learner or models.
2. Boosting: Boosting is a sequential ensemble learning method that focuses on training weak learners. In this technique, each subsequent learner improves the errors of previous learners in the sequence. A sample of data is first taken from the initial

dataset. This sample is used to train the first model, and the model makes its prediction, the result gotten is used to train the next model and its further iterations hence improving on the errors from previous models and increasing accuracy.

3. Majority Voting: Majority voting is a technique in ensemble learning as it helps to improve the overall accuracy and robustness of the ensemble by allowing multiple models to make a decision using the result of their predictions based on the class that gets the most votes.

2.7 Intelligent Methods

4. Intelligent classification methods consist of computational techniques based on artificial intelligence, its used often due to the shortcomings of the traditional method. (M. Demirci *et al 2021*). Some of the intelligent methods are the Expert Systems, Artificial Neural Networks (ANN) , Stacking, also known as stacked generalisation, involves training multiple

models to generate predictions, and then using a meta-model to combine their outputs. Instead of simply averaging the predictions, the meta-model is trained to learn how to best weigh the predictions from the individual models. Stacking allows for more sophisticated combinations of models and can often lead to improved performance Fuzzy Logic and Machine Learning methods.

In this study emphasis is placed more on Support Vector Machine (SVM), Naive Bayes (NB) and C4.5 Decision tree algorithms and their ensembles using Bagging, boosting and Majority Voting techniques. In this study we engaged the SVM, NB and C4.5 Decsion tree algorithms and their ensembles

2.7.1 Support Vector Machine (SVM)

The SVM is a widely employed classifier and it is primarily used for data classification and prediction. It proves to be highly effective when faced with challenges such as limited training data, large amounts of input data, and non-linear datasets. This classifier handle large feature, rendering it suitable for large-scale data classification with remarkable accuracy. (Hazlee et *al 2020*).

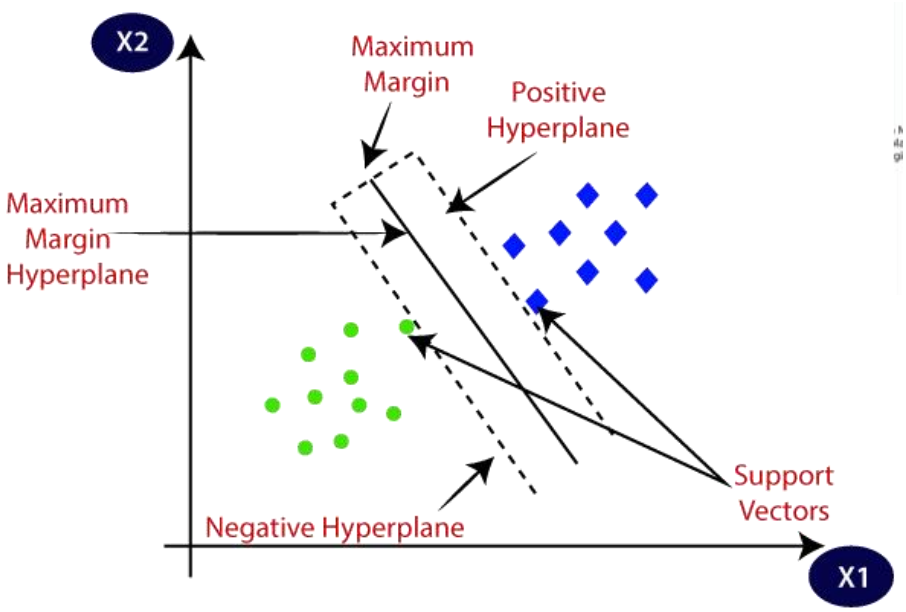

Figure 2.3 Support Vector Machine (www.javatpoint.com)

2.7.2 Naive Bayes (NB)

It is a classification technique based on Bayes' Theorem, it is a classifier type that involves the use of features where by the presence of a particular feature in a class is unrelated to the presence of any other feature, this technique is based on a probabilistic approach where each feature contributes to the overall result of the class but each feature can stand independent of the other in this technique.

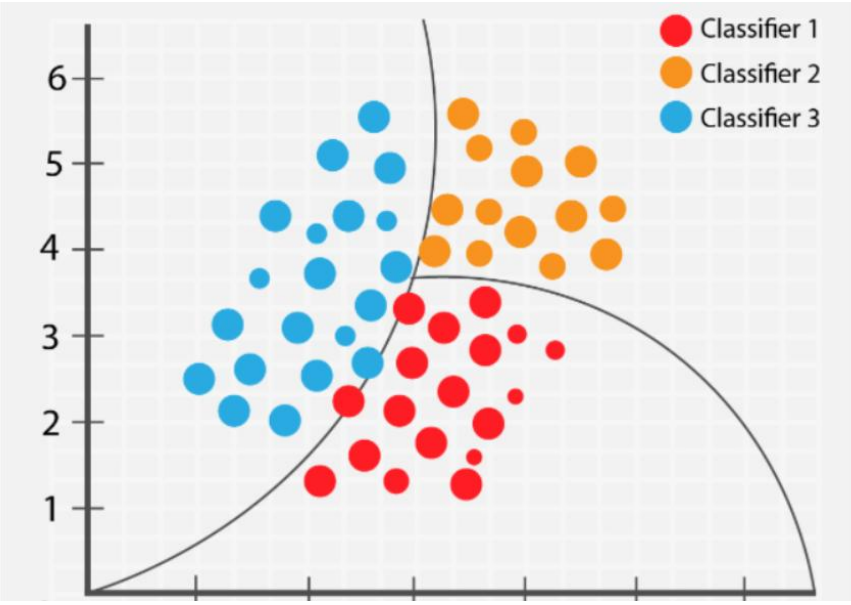

Figure 2.4 Naive Bayes Classifier (www.analyticsvidhya.com)

2.7.3 Decision Trees

The decision tree algorithm is a widely used method in machine learning for approach, where internal nodes make binary decisions to merge or separate classes, leading to terminal nodes that represent the final classification.

The process begins with tree building and then pruning . It involves binary recursive partitioning, repeatedly sorting the data into partitions. The tree structure is determined by all training data examples . Each node is split using binary split, dividing the data into two parts to minimize deviation. This split process continues until each node reaches the specified minimum node size, which represents the number of training instances in each node. At this point, the formed node becomes a terminal node. (Rosmaliati et *al 2022*).

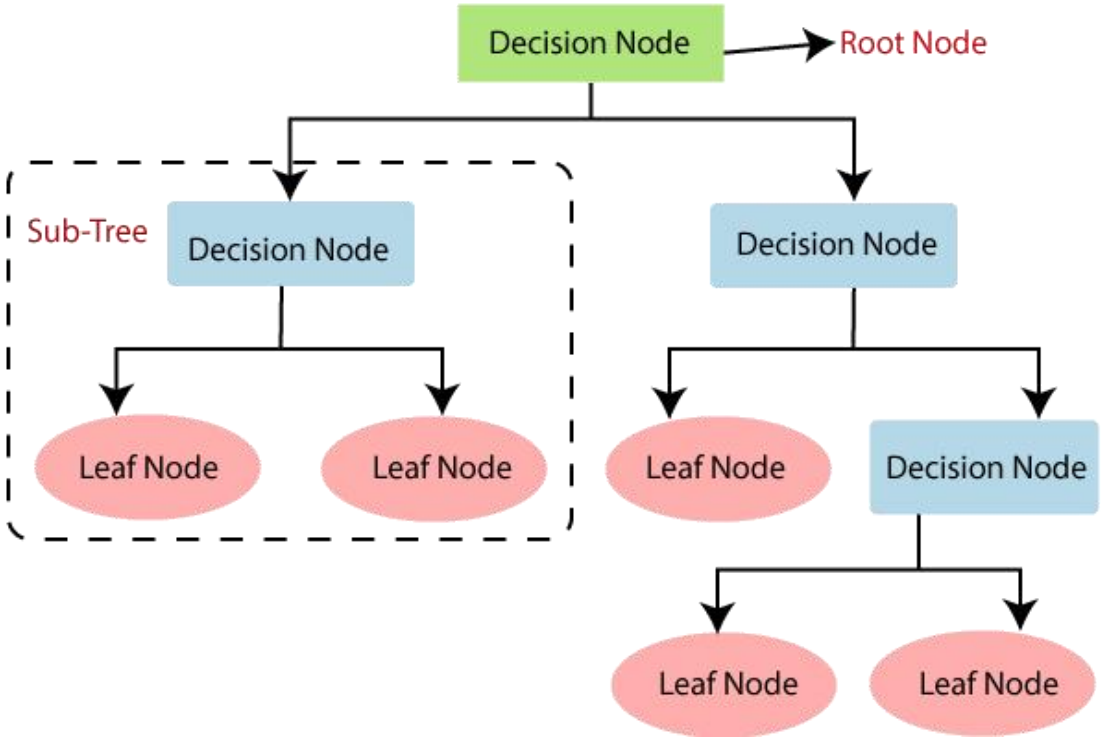

Figure 2.5 Decision Tree Algorithm in Machine Learning (www.javatpoint.com)

2.8 Knowledge Discovery Database and Data mining

Data mining is the process of applying computational algorithms and techniques to extract meaningful patterns, knowledge, and insights from large datasets. It involves the exploration and analysis of data to discover hidden relationships, trends, and structures that can be valuable for decision-making and predictive modeling. Knowledge Discovery in Databases, encompasses the entire process of finding

valuable knowledge from data. Within this process, data mining serves as a specific step that involves applying algorithms to extract patterns from the data. (fayyad *et a*l).

In the process of KDD and DM data selection begins the process, this is where relevant data sources are chosen for analysis. The selected data is then preprocessed, which involves cleaning the data, handling missing values, and transforming it into a suitable format. These six steps listed below are used in data preprocessing ;

1. Loading DGA Dataset.
2. Handling missing data.
3. Handling text labels.
4. Separating dependent and independent variables.
5. Handling data with categories.
6. Normalizing the data. Splitting training and testing data (Saravanan et al 2020)

This ensures that the data is of high quality and ready for analysis. Once the data is prepared, data mining techniques are applied. These techniques include clustering, classification, regression, association rule mining, and anomaly detection, among others. Clustering algorithms group similar instances together, while classification algorithms assign labels or categories to instances based on their characteristics. Regression analysis predicts numerical values based on historical data. Data mining helps in discovering hidden patterns. It enables the extraction of knowledge that can support informed decision-making, assist in predicting future outcomes, identify opportunities or risks, improve operational efficiency, and gain a deeper understanding of the data.

The KDD process is an iterative process and it requires multiple iterations of the above steps to extract accurate knowledge from the data.The following steps are included in KDD process:

1. Data Cleaning:

Data cleaning is defined as removal of noisy and irrelevant data from collection.

Cleaning in case of Missing values.

Cleaning noisy data, where noise is a random or variance error.

Cleaning with Data discrepancy detection and Data transformation tools.

2. Data Integration:

Data integration is defined as heterogeneous data from multiple sources combined in a common source (DataWarehouse). Data integration using Data Migration tools, Data Synchronization tools and ETL(Extract-Load-Transformation) process.

3. Data Selection:

Data selection is defined as the process where data relevant to the analysis is decided and retrieved from the data collection. For this we can use Neural network, Decision Trees, Naive bayes, Clustering, and Regression methods.

4. Data Transformation:

Data Transformation is defined as the process of transforming data into appropriate form required by mining procedure. Data Transformation is a two step process:

5. **Data Mapping**: Assigning elements from source base to destination to capture transformations.

6. Pattern Evaluation:

Pattern Evaluation is defined as identifying strictly increasing patterns representing knowledge based on given measures. It find interestingness score of each pattern, and uses summarization and Visualization to make data understandable by user.

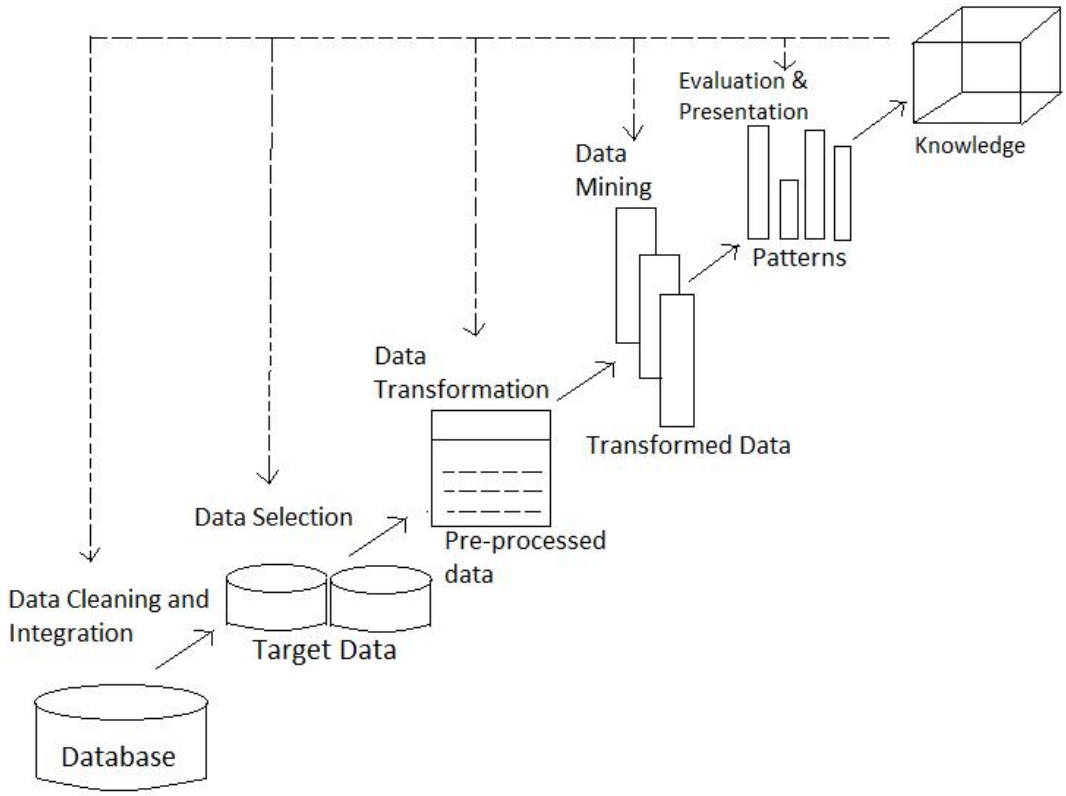

Figure 2.6 Data mining steps in dataprocessing (techblogmu.blogspot.com)

2.9 Review of Related Literature

In this section, we review in a tabular form the current state-of-art literature related to cancer classification and detection using ensemble machine learning techniques. Numerous techniques have been reviewed for the classification of cancers using ensemble as Shown in Table 2.1:

Table 2.1: Review of Related Literature

S/N	TITLES/AUTHOR(s)	MOTIVATION	METHODOLOGY	CONTRIBUTION TO KNOWLEDGE	LIMITATION
1	Optimal Dissolved Gas Rations Selected by Genetic Algorithm for power Transformer Fault Diagnosis Based on Support Vector Machine, **Jinzhong Li, Qiaogen Zhang, Ke Wang, Jianyi Wang, Tianchun Zhou and Yiyi Zhang**	This study is aimed at creating a new method to detect faults in a power transformer with more robustness high accuracy and early time detection to prevent destruction of insulating systems In the transformer.	Some methods used in achieving this are DGA (Dissolved Gas Analysis), SVM (Support Vector Machine), Back propagation Neural Network, IEC ratios, Genetic Algorithm, Genetic operations.	The proposed method in this study has a better classification capability over the major DGA data validated by the SVM and the method was used on a China transformer to test run the system using DGA sample and it attained a high accuracy of 87.18% alongside high robustness and reliability.	The Proposed method faced poor utilization of feature selection and classification Algorithm to attain its results.
2	Power Transformer Fault Diagnosis Using Graph Neural Network based on Dissolved Gas Data, **Weiyun Mao,**	This study is proposed to produce a model that out performs the machine learning baseline method	In this study the DGA, Multivariate time series forecasting AR,	In this study an improved Multivariate model was created, the	The method faced poor interpretability of the improved multivariate model

	Bengang Wei, Xiangyi Xu, Lu Chen, Tianyi Wu, Zhengrui Peng, Chen Ren	used with the DGA prior to this time, this is because it ignores the internal communication of the dissolved gas data in the time dimension, hence affecting the overall analysis.	ARMA, VAR, VARMA, Graph Neural Networks, Cross entropy loss Function, Multivariate time series neural network was used.	Graph Neural Network framework achieved an excellent result in the fault prediction and the proposed method produced a higher accuracy than the industrial detection method.	and lacked proper troubleshooting procedure to attain more stability in the process
3	Improved Genetic Algorithm and XGBoost Classifier For Power Transformer Fault Diagnosis, **Zhanhong Wu, Mingbiao Zhou , Zhenheng Lin , Xuejun Chen and Yonghua Huang.**	This study proposes an effective transformer fault diagnosis system that has an improved identification accuracy using an improved Genetic Algorithm with the XGBoost to form a hybrid diagnosis network. This tackles poor transformer fault recognition problem.	The methods used in carrying out this research are IGA (Improved Genetic Algorithm), DGA, Artificial Intelligence Algorithms, feature selection and classifier optimization, Traditional Genetic Algorithm (TGA), XGBoost.	This proposed modification improves the overall search capability of the Improved Genetic Algorithm, due to this proposed method of the IGA the fault diagnosis accuracy of a power transformer attained a 99.2% ratio compared to other methods.	
4	Power transformer faults diagnosis using undestructive methods (roger and iec) and artificial neural network for dissolved gas	The purpose of this study is to improve the quality of the diagnostics of the electrical power transformer by Artificial Neural Network (ANN) methods to reduce the	The methods used in this study comprises of DGA, Rogers and IEC ratios, Feed forward Neural Network, Artificial	The proposed method in this study rendered a solution to the limitation of previous methods used like the ratio techniques and it	This method had a draw back in performance due to the unavailability of deep learning to improve the accuracy

	Analysis applied on the functional transformer in the algerian North-eastern: A comparative study , **L. Bouchaoui, K.E. Hemsas, H. Mellah, S. Benlahneche**	failures of power transformers and to conduct proper verification of the abilities of various methods in the sustenance of power transformer health.	Neural Network, Neural Networks based on traditional methods.	improved the percentage of accuracy of the IEC method from 20% level to a 70% level and for the Roger method an increase of 40% to 70% was reached.	of results.
5	Novel Power Transformer Fault Diagnosis using Optimized Machine Learning methods, **Ibrahim B.M. Taha and Diaa-Eldin A. Mansour.**	This proposed method introduces a system that solves the insufficient limitations of the DGA approach in fault diagnosis. This method used data transformation and optimized machine learning methods to improve accuracy and reliability.	The various methods used in this study are DGA, Optimized Machine Learning(OML) like SVM, K-NN, DT, DA, NB, EN, Data transformation techniques.	The fault prediction accuracy of the Ensemble Method (EN) used in the OML using the dataset samples attained a high of 90.61% and this proposed method is superior to the other methods.	
6	A Power Transformers Fault Diagnosis Model Based on Three DGA Ratios and PSO Optimization SVM , **Hongzhe Ma, Wei Zhang, Rongrong Wu, Chunyan Yang**	This study renders a solution for the drawbacks in the fault diagnosis of a power transformer using DGA. It offers a more improved accuracy of fault diagnosis using optimized SVM.	The methods used in this study comprises of SVM, Particle Swarm Optimization (PSO), DGA and Genetic Algorithm (GA).	This study presented an accurate fault diagnosis model, it attained values of 57.14%, 60.71% and 85.71% for the SVM, GASVM and PSOSVM respectively, there by	

				proving that the optimized SVM attained higher .	
7	Classifier Method in Fault Diagnosis of Oil-immersed Power Transformer by considering Dissolved Gas Analysis, **Rosmaliati , Bernandus Anggo Seno Aji , Isa Hafidz1, Ardyono Priyadi , Mauridhi Hery Purnomo.**	This study is aimed at enhancing the accuracy of fault diagnosis in power transformers using artificial neural networks based on the classification techniques to solve the limitations of the DGA detection method.	The methods used in this study are the Neural Networks, DGA, Genetic Algorithm (SVM), Fuzzy Logic.	The proposed method using Neural Networks produced the highest and most accurate results when compared to others and its more accurate for DGA classification in fault diagnosis	This proposed model cannot utilize Big Data for power systems like the power transformer.
8	Data Driven Fault Diagnosis of Power Transformer using Dissolved Gas Analysis, **Arian Dhini , Akhmad Faqih, Benyamin Kusumoputro, Isti Surjandari, Andrew Kusiak.**	This proposed study is to introduce a data driven method for diagnosing faults in a power transformer based on DGA data using SVM because of its high accuracy while maintaining fast computation. This method was introduced because of the draw back of the DGA in poor categorization of power transformer state.	The methods used are the DGA, SVM, Extreme Learning Machine Radial Basis Function, FDD, Single Minority Oversampling Technique (SMOTE), Back Propagation Neural Network (BPNN)	The proposed approach performs better than others in accuracy and computation time.	This approach faced the drawback of unavailability of enough training data to process the methods to attain maximum result
9	Fault Detection and Protection of Power Transformers Using Fuzzy Logic , **Babagana Ali Dapshima, Ankit Pandey,**	This proposed method is used to detect and protect power transformer faults thereby detecting electrical and	The methods used in carrying out this study comprises of Fuzzy logic, usage of	This proposed approach proved the quick detection and identification of faults	There was an error in the output variable in current and the rate of the change with time

	Arya Srivastava, Dr. Jabir Ali.	mechanical faults early enough and accurately also saving costs and time. This solves the issue of power transformer malfunctions and power outages.	the MATLAB software.	in a power transformer, this study showed the relevance of zero output representing a normal state and if there be lesser of higher value it shows an abnormal change in states.	
10	Fault diagnosis of Power Transformers with Machine Learning methods using Traditional Methods Data, **M. Demirci H. Gozde M.C. Taplamacioglu**	This proposed study was carried out to solve the draw back of the traditional method of fault diagnosis with the use of Machine Learning and intelligent methods to achieve higher accuracy of the diagnosis.	This study made use of certain methods like the DGA, Intelligent methods (ANN, Fuzzy logic), K-NN, Decision tree, SVM and some traditional methods like the Duval Triangle method, Rogers ratio, Doermenburg ratios.	In this study the highest accuracy and diagnostic accuracy in all cases was achieved by one of the Decision tree ML method, while the traditional Duval triangle method reached the highest diagnostic ability using gas percentage.	
11	Hybrid Feature Selection Artificial Intelligence Gravitational Search Algorithm Technique for Automated Transformer Fault Determination Based on Dissolved Gas Analysis,	This work was proposed to solve the limitation of the DGA methods that relies on the concentration of the dissolved gases. This proposed work detects the fault type using Artificial	This work made use of Artificial Intelligence, SVM, Neural Network, oil sampling, Feature selection, Optimization	This work showed the performance of the SVM and ANN was optimized by the Gravitational Search Algorithm and it yielded a higher	

	Hazlee Azil Illias , Kai Choon Chan , Hazlie Mokhlis.	Intelligence.	techniques, GSA.	accuracy than when it was ANN and SVM alone. It also produces a higher accuracy in fault detection with less features or dissolved gases used.	
12	Application of Parzen Window Estimation for Incipient Fault Diagnosis in Power Transformers **Dr. Md Mominul Islam , Dr. Gareth Lee1 , Dr. S N Hettiwatte**	This proposed method in this study serves as a solution to the inaccuracy and the inability of the traditional methods In properly classifying cases.	The methods used in this study are Probability Density Function (PDF), Data processing and normalization, Feature Extraction, Density function estimation, Traditional methods, DGA.	The Proposed method attained an overall highest performance of 95% accuracy compared to other methods and it showed better reliability than other conventional ratio based diagnosis	The success of this work was dependent on the number of training samples and it affected the accuracy.
13	A Multinomial DGA Classifier for Incipient Fault Detection in Oil-Impregnated Power Transformers. Algorithms, **Odongo, G., Musabe, R., & Hanyurwimfura, D**	This study proposes a solution to the inconclusive interpretation and accuracy of incipient fault diagnosis by Novel Multinomial classification and model (KosaNet) based on Decision Trees.	The methods used in this work are the DGA and other ML methods like the Decision Trees, K-NN, Random forest with classification model	The Proposed method KosaNet has a far higher classification ability using multiclass DGA and reached an accuracy high of 99.8% and showed an increase in the network uptime which led to less	This proposed method faced the inability to explore KosaNet with other option like the time series data using real time transformers with smart sensors

				power disruptions in the system.	
14	Fault Prediction of Transformer Using Machine Learning and DGA **Saravan D., Ahmed H., Ajit S. Hannan M., Rabrindra N. S**	This study was aimed at tackling or solving the limitations of the DGA method on human dependency by proposing multilayer Artificial Neural Network (MANN) and SVM classifier model.	The methods used in this study are Artificial Neural Network Models, SVM classifier models, datasets.	In this study the SVM Classifier model reached an overall accuracy of 80% and 86% fault prediction. Both models presented in this study MANN and SVM classifier model successfully attained good fault prediction and accuracy in fault classification	This study could not conduct further analysis using other machine learning methods and was unable to use gas ratios a input to the machine learning models.
15	K-NN and Mean-Shift Algorithm Applied in Fault Diagnosis in Power Transformers by DGA **Alex R. Soto Enriquez , Shigeaki L. Lima, Osvaldo R. Saavedra**	To reduce economic loss caused by the interruption and instability in power transformers, this work proposes a method to accurately detect faults and tackle the problem of instability and inaccuracy	In this work DGA was used as a general method while K-NN classifier application was used with the Mean shift algorithm to conduct this work.	In the test of diagnosis failures by DGA, the K-NN classifier attained a high value of 97.73% with weighted classification while using mean shift algorithm with Joint data, it reached a high of 100% and 99.81% respectively.	

CHAPTER 3
RESEARCH METHODOLOGY

3.1 Introduction

Research methodology is the in depth systematic approach applied in solving problems. A well-constructed research methodology assures a better, well-articulated, accurate and credible study. With that established this study explores current research methods in machine learning, emerging themes, and the implications of those themes in machine learning research which reveals that Machine learning uses quantitative research methods with experimental research design being the de facto research approach. The study also reveals that researchers nowadays use more than one algorithm to address a problem. The most commonly used algorithms in addressing both classification and prediction problems are; Naïve Bayes, Support Vector Machine, Random Forest, Artificial Neural Networks, and Decision Tree. (Kamiriet *al 2021*).

In this study, a deep analysis is carried out on the power transformer faults using the datasets gotten from the Dissolved Gas Analysis (DGA). The next phase is the data preprocessing whereby the data is cleaned, transformed and reduced that is the data is checked for missing values or noisy data then it is normalized then reduced which further means that the data is sampled by removing irrelevant volumes of data and retaining the meaningful information. (Fayyad et al 1996)

3.2 Data

Data is a fundamental instrument in research, data in this study are the gas concentrations given by the chromatographic analysis tests of the oil samples gotten from the power transformer , these samples are taken over the life of the transformer (training step), or suspected in the data set sample (L. Bouchaoui, *al 2021*).. These gases are Hydrogen gas (H_2), Methane gas (CH_4), Carbon monoxide gas (CO),Carbon dioxide gas (CO_2), Ethylene gas (C_2H_4) ,Ethane gas (C_2H_6), Acetylene (C_2H_2)

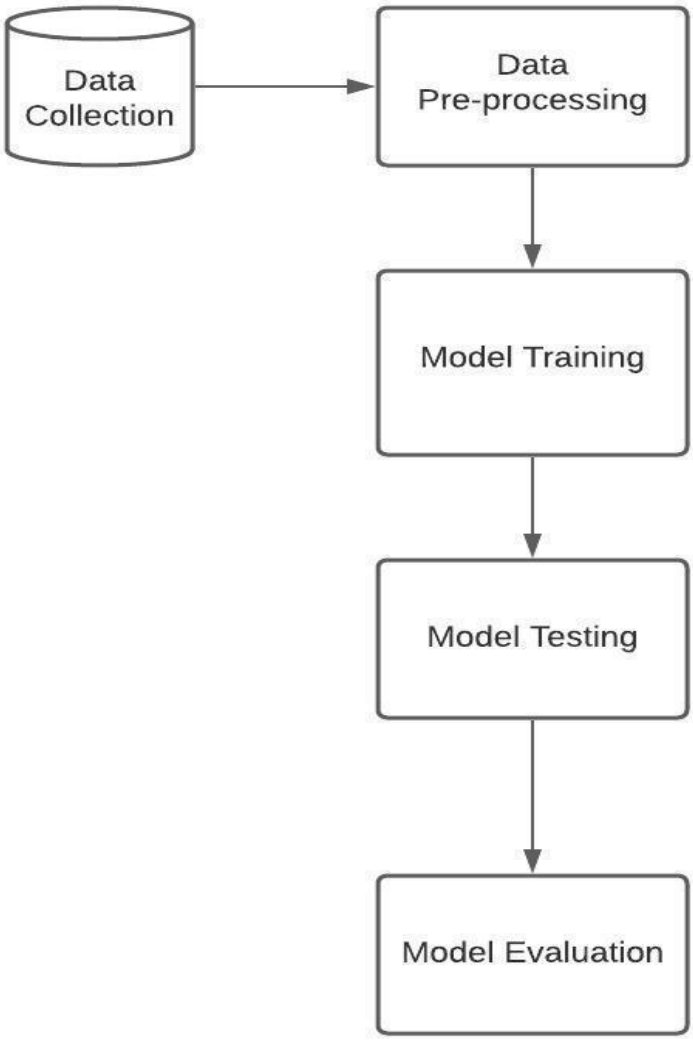

Fig. 3.1 Mode of General Research

3.3 Data processing

Data processing is the stage where by the data is cleaned from any form of noisy data and where the missing values are evaluated. Data transformation which is also the normalization or standardization of the data. This is done before the data is taken into classification using several algorithms like the support vector machine or the Decision trees which are you used for effective classification and performance evaluation.

3.3.1 Noisy data

Noisy data are unstructured or meaningless data, this is a form of corrupted data that cannot be interpreted by the machines there by causing unnecessary disturbance in the system. Data processing as earlier stated is carried out to eliminate the noisy data in the datasets to create a more stable and reliable data for analysis

3.3.2 Missing Values

Missing values refers to the absence of a data in certain cells in rows and column of a dataset. There is a high tendency of a dataset to have missing values due to reasons like poor data collection, errors in the data entry or intentional omissions in some cases (fayyad et al) shows us that there is a method or methods of dealing with the issue of missing values and these methods are the mean/median imputation and also the K-Nearest Neighbor (K-NN). These methods have proven effective in recent years. Researches proved that these methods are dependable but varies in accordance to the values being used, in this study we are using a numerical datasets which is well accepted by the mean/median imputation.

3.3.2.1 Mean/ median imputation

In this technique, The missing values are replaced with the mean or the median of the column of the missing value by first calculating the mean of the available data then replace the missing values with the calculated mean and this steps works with the median.

ALGORTHM 1

Step 1:

Select the featues from the available set of data

Step 2:

Calculate the mean/median for the availble date from the column where the missing values are located.

Step 3:

Impute the missing values using the values attained from step two.

3.3.3 Data Normalization

Data Normalization is a technique used in data processing to transform numerical features in a dataset to a common range. This technique normalize the attributes into a range of usually zeros and ones (0, 1) (fayyad *et al* 1996). In data normalization an approach often used to normalize data is the "Min-max normalization". This technique involves the subtraction of the minimum value from each value in the feature and then dividing it by the range of the features

$$V' = \frac{V - \text{Min}(A)}{\text{Max}(A) - \text{Min}(A)} \qquad (3.1)$$

V' = New value of each entry in data

V= Old value

A= Attribute data | Max(A)= Maximum value | Min (A)= Minimum Value.

A main benefit of using min-max normalization is the improved accuracy and performance of models. By scaling input variables to a specific range, it makes it easier for algorithms to identify patterns and relationships between different features. This technique can help prevent certain input variables from overpowering others within a model.

Another advantage of using min-max normalization is that it simplifies the interpretation of model results because all input features are transformed into a consistent range, they become more comparable and easier to understand.

3.4 Classification Algorithms

Classification algorithms are a crucial type of machine learning algorithm used to categorize inputs into predefined groups. One of the main purposes of this algorithm is to learn from labeled examples and apply that knowledge to predict new labels for unlabeled instances. Some of the most common types of classification algorithms include decision trees, support vector machines, k-nearest neighbors, and logistic

regression. In this study we are using the support vector machines, Naive Bayes and the decision tree to classify the data.

3.4.1 Support Vector Machine (SVM)

The Support vector machine Is a type of classifier used for the statistical learning, data classification and prediction. This machine learning algorithm can be used for both classification and regression tasks by dealing with the issues of large inputs data and small training data (Hazlee et al 2020) went on to say that the support vector machine constructs a dividing hyperplane in a higher dimensional space to classify data by maximizing the space between data points. It also does the task of converting datasets from a lower plane to a higher hyperplane using kernel functions like the poly or the Radial Basis Function (RBF), but the RBF is more preferred (Saravan et al 2020).

The Support vector machine ML algorithm is a superior performance classifier technique that improves the generalization ability of learning machines by seeking the minimum structured risk and achieve good result (Xioali et al), a major advantage of the support vector machine is the ability to handle high dimensional data effectively while maximizing the margin between different classes. This leads to better performance classification.

3.4.2 Naive Bayes

Naive Bayes is a classification algorithm that works based on the Bayes theorem. It assumes that all features are independent of each other, which is not always true in real-world scenarios but can still yield fairly accurate results. The algorithm uses probability theory to predict the class of an input instance by calculating the conditional probabilities of each feature given its class.

One advantage of using naive Bayes is that it requires less training data compared to other algorithms and can be trained quickly even with large datasets. Additionally, it performs well in situations where there are many irrelevant features or noise in the data since it only takes into account relevant information for classification.

3.4.3 Decision Trees

The Decision tree algorithm is a learning algorithm used for classification tasks, The way this works is y splitting the datasets into subsets based on the most significant attribute there by making decisions at each node, each node forms a binary decision whether to combine or separate classes (Rosmaliati et al 2020). The process starts with the tree building and then pruning the data (V.T Tram, 2009). The goal of the decision tree algorithm is to create a model that can make accurate classifications by using a series of decisions based on the input data of each node. Below are the steps involved:

ALGORTHM 2

Step 1:

Selecting the most informative feature as the root node.

Step 2:

Creating branches from the root node to represent possible values of that feature.

Step 3:

Recursively repeating step 1-2 on each subset of data created by following one branch from the root.

3.4.3.1 C4.5 Decision Tree Algorithm

The C4.5 Decision Tree Algorithm is an algorithm for classification tasks. It uses information gain ratio to select the best attributes for splitting and can handle both discrete and continuous data types, making it more versatile than other decision tree algorithms. It has the ability to deal with missing values effectively due to its robust nature, it is often used in several fields like healthcare, finance, marketing among others for predictive purposes since it delivers robust models even on small datasets without losing prediction accuracy .

3.5 Ensemble methods

Ensemble methods in machine learning involve combining multiple models to improve predictive accuracy. This can be done through techniques such as bagging, boosting, and stacking.

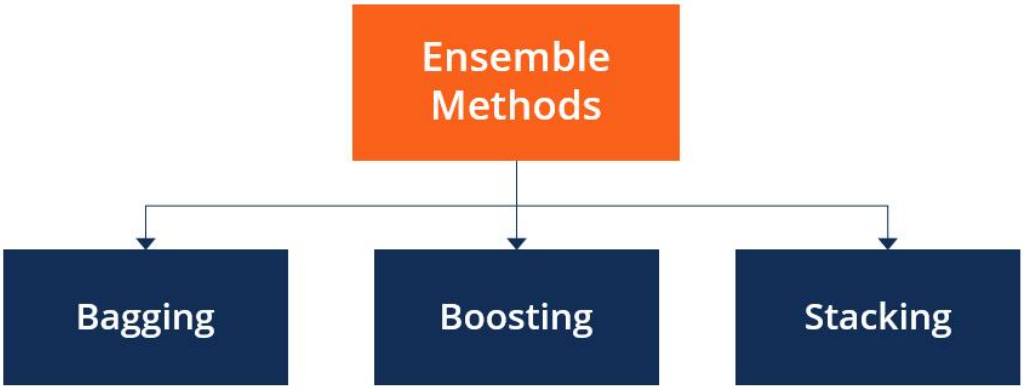

Fig 3.2 Ensemble methods (https://corporatefinanceinstitute.com/resources/data-science/ensemble-methods/)

3.5.1 Bagging

Bagging also known as bootstrap aggregating. involves using subsets of data to train several independent models on different samples from the dataset, while boosting adjusts weights on misclassified examples to emphasize harder-to-classify observations. This approach in machine learning involves creating multiple copies of the base learner and training them on different subsets of the data by sampling with replacement. Each copy generates its own set of predictions, which are then combined using majority voting or averaging to produce a final prediction from the ensemble.

This technique is useful in reducing variance and preventing overfitting by taking advantage of diversity between independently trained models. Bagging is especially effective when dealing with datasets containing mislabeled samples or when base learners tend to overfit during training. (breiman 1996)

3.5.2 Boosting

Boosting is an ensemble method where multiple weak models are combined to create a stronger model. The weak models are trained sequentially and each subsequent model attempts to correct the errors made by its predecessor. By doing this, the final boosted model becomes better at predicting outcomes than any individual weak model. Boosting has become increasingly popular in recent years due its ability to improve accuracy and reduce variance in prediction models. In Boosting, algorithm evaluate the output of every model, focus on misclassified labels and give them more weight

age before inducing for the next model. In this way at last we get most accurate prediction. (Suyash et al 2022).

3.5.3 Stacking

Stacking is a type of ensemble method in machine learning where multiple models are combined to create a stronger prediction model. The idea behind stacking is to use base-level models or individual learners trained on the same or different datasets, each with its own strengths and weaknesses, and combine their output. The output of base level classifiers works as input for meta level classifier while making predictions using the new input (Suyash et al 2022).

The stacking approach is more significant than the others due to its ability to combine multiple models, address their weakness and create a stronger model, it also holds the ability to reduce overfitting and improve generalization by learning from the outputs of other learners. These attributes help increase accuracy in prediction and better performance.

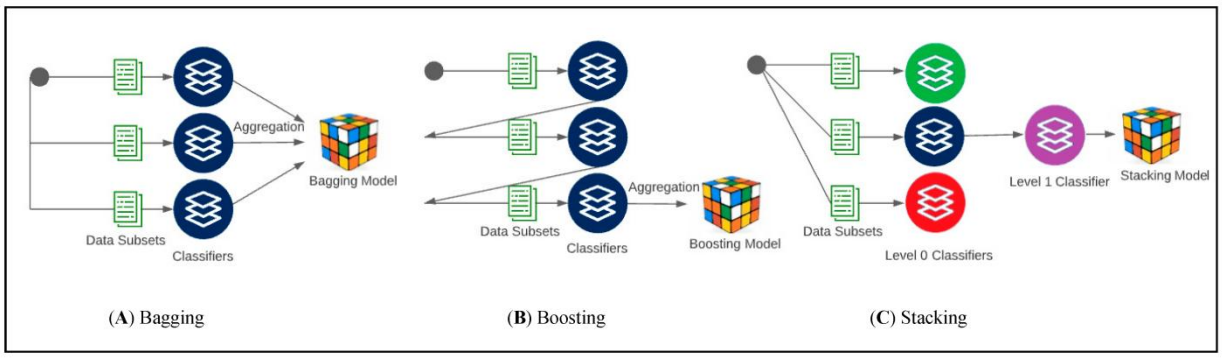

Fig 3.3 Ensemble techniques (https://www.mdpi.com/2076-3417/13/1/30Ensemble techniques: Bagging, Boosting, and Stacking workflow.)

3.6 Voting

Voting is a type of ensemble method in machine learning where multiple independent models are trained on the same dataset and their predictions are used to make final decision by choosing the most frequent prediction**.** This approach is commonly applied for classification tasks. The main idea behind voting is that combining multiple diverse models can produce more accurate and robust predictions compared to individual learners, especially when some of these models may have some amount

of errors or biases in their own predictions.

3.6.1 Majority Voting

Majority Voting is a technique used in machine learning where multiple models or algorithms are constructed and their predictions are combined to reach a final decision. The prediction with the most votes or the majority of votes is considered the final decision.

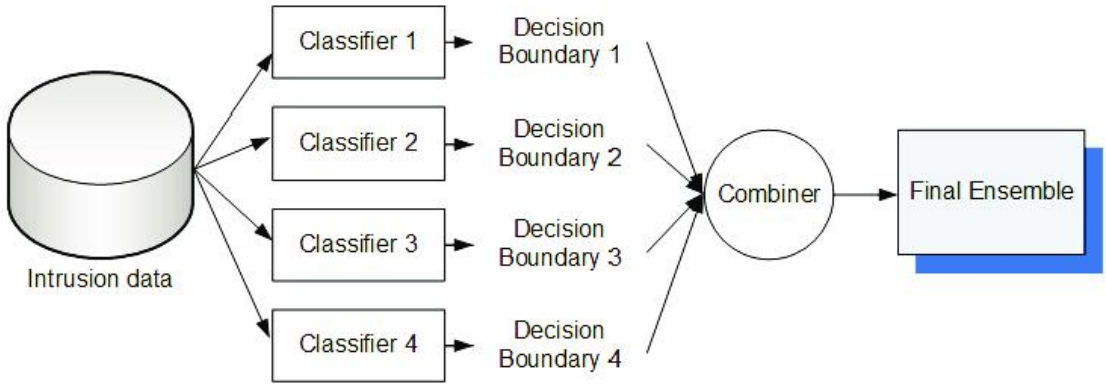

Fig 3.4 Majority voting (https://www.researchgate.net/figure/Classifier-ensemble-using-majority-voting_fig1_318260780.)

Further said, the concept of majority voting involves aggregating the predictions from multiple classification models to determine the ultimate class label for a given instance. The crux of this technique lies in selecting the class label that garners the highest count of predictions across the ensemble of classifiers. An ensemble of diverse classification models, such as decision trees, support vector machines, or neural networks, is employed to generate individual predictions for the same dataset or problem. Each of these models independently produces a class label prediction. Then, a vote counting mechanism is applied to tally the occurrence of each predicted class label across all the models.

The class label that emerges with the highest frequency among these predictions is designated as the final prediction. This approach capitalizes on the wisdom of the ensemble, leveraging their collective knowledge to yield a more robust and dependable decision. It's particularly beneficial for reducing the impact of potential outliers or inaccuracies in the individual model predictions, thereby enhancing the overall predictive accuracy of the system. (Dharmaraj et al 2018)

ALGORTHM 3

Step1

Train multiple models or algorithms on the same dataset.

Step 2

Make predictions using each model on a new set of data.

Step 3

Assign a weight to each model based on its performance in predicting the outcome correctly.

Step 4

Combine the predictions made by all models and select the prediction with majority votes as final output.

Step 5

In case of a tie, choose one at random or use another criterion to break it.

Step 6

Evaluate the accuracy of your final decision based on known outcomes and adjust weights if necessary to improve future predictions.

3.7 Performance Evaluation Methods

Performance Evaluation It involves a set of techniques and metrics designed to quantify and analyze how well a model performs its intended task such as accuracy, recall, precision, AUC and F-measure The confusion matrix embodies the performance evaluation methods that would be used in evaluating the performance of the model proposed in this project:

3.7.1 Confusion Matrix

This evaluation method (or prediction) produces four outcomes ± true positive, true negative, false positive and false negative. x True Positive (TP):

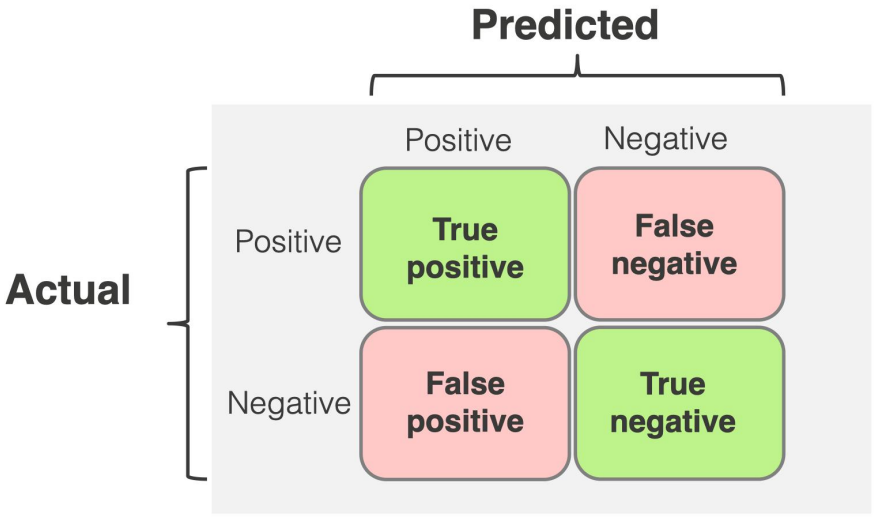

Fig 3.5 Confusion matrix metrics (https://www.evidentlyai.com/classification-metrics/confusion-matrix)

1. **True Positives (TP)**: These are instances where the model correctly predicted the positive class (e.g., "yes" or "1") when the actual class was indeed positive.
2. **True Negatives (TN):** These are instances where the model correctly predicted the negative class (e.g., "no" or "0") when the actual class was indeed negative.
3. **False Positives (FP)**: Also known as Type I errors, these are instances where the model incorrectly predicted the positive class when the actual class was negative. This is sometimes referred to as a "false alarm" or "Type I error."
4. **False Negatives (FN):** Also known as Type II errors, these are instances where the model incorrectly predicted the negative class when the actual class was positive. This is sometimes referred to as a "miss" or "Type II error."

3.7.2 Accuracy

Accuracy is a common statistical measure employed to assess the effectiveness of a binary classification test in identifying or ruling out a specific condition. Essentially, accuracy represents the proportion of correct predictions, encompassing both true positives and true negatives, relative to the total number of cases assessed. In this way, it provides a basis for comparing the likelihood assessments before and after the test. For the sake of semantic clarity, accuracy is often denoted as the "Rand accuracy" or

"Rand index," serving as a key evaluative parameter. The formula for binary accuracy is as follows:

$$\text{Accuracy} = \frac{Tp+Tn}{Tp+Tn+Fp+Fn} \qquad (3.2)$$

Where TP means True positive; FP means False positive; TN means True negative; FN means False negative.

3.7.3 Precision and Recall

Precision is determined by dividing genuine positives by any positive predictions. While Recall (or True Positive Rate) is computed by dividing true positives by everything that should have been forecasted as positive

The mathematical equation for precison:

$$\text{Precision} = \frac{Tp}{Tp+Fp} \qquad (3.3)$$

The mathematical equation for Recall:

$$\text{Recall} = \frac{Tp}{Tp+Fn} \qquad (3.4)$$

Where TP means True positive; FP means False positive; FN means False negative.

3.7.4 F1-Score

The F1-score is a statistical measure commonly used in binary classification tasks to evaluate the model's performance, taking into account both precision and recall.The

F1-score combines precision and recall into a single metric. It ranges from 0 to 1, with higher values indicating better model performance. A high F1-score indicates a good balance between precision and recall, which is crucial in scenarios where both false positives and false negatives have significant consequences It is particularly useful when dealing with imbalanced datasets, where one class significantly outnumbers the other.

$$F_1 = 2 \times \left(\frac{\text{Precision x Recall}}{\text{Precision + Recall}}\right) \qquad (3.5)$$

3.7.5 AUC ROC

The ROC (Receiver Operating Characteristic) curve provides a visual representation of how well a receiver or classifier performs at various decision thresholds. It accomplishes this by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR). As the TPR increases, so does the FPR. The ROC AUC (Area Under the Curve) is a single numerical metric that summarizes the overall performance of the classifier. Higher AUC values indicate better classifier performance, whereas a value of 0.5 suggests performance equivalent to random guessing. This curve allows us to assess the trade-off between sensitivity and specificity, with a perfect classifier residing at the top-left corner of the plot. In summary, the ROC curve and AUC are valuable tools for evaluating the performance of classification models, facilitating the understanding and comparison of different models.

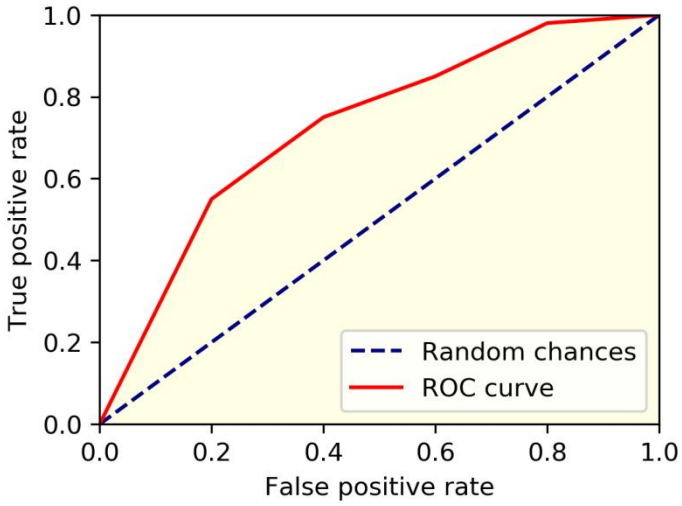

Fig 3.6 ROC Curve Transforms the Way We Look at a Classification Problem | by Huy Bui | Towards Data Science)

CHAPTER 4
IMPLEMENTATION AND DESIGN

4.1 Overview

System implementation refers to the actual building and evaluation of a model that has been suggested. The purpose of this step is to physically create the model, document the procedure, and provide the final results.

4.2 System Requirements

System requirements refer to the necessary hardware and software specifications that must be met in order for a system to perform optimally, efficiently, and consistently. If these requirements are not
fulfilled, it can result in decreased performance or technical issues.

4.2.1 Hardware Requirements

A good laptop with the following hardware is needed to run the model conveniently:
a. Processor: Intel i5, and above
b. RAM: 8GB or more
c. Memory: above 12GB of space
d. Processor Speed: 2.5GHZ and above
e. GPU: Because the model uses several cores for processing during its training stages, a GPU is necessary to increase the model's performance.
68f. TPU: Because the model uses several cores for processing during its training stages, a TPU is necessary to increase the model's performance.
g. Internet: A stable internet connection from a reliable Internet Service Provider (ISP) because the software used 1s required to run online.

4.2.2 Software Requirements

A computer's software is an intangible Component of the computer. To run the model, the following

software is required:

a. Operating System: either a Windows 7, 8, 10, or Linux operating system should be sufficient but Linux is recommended,

b. Integrated Development Environment (IDE): Google Co laboratory (Google Colab)

4.3 Tools For Model Development

Software developers use these programs to create, debug, and maintain other programs and applications.

4.3.1 Google Colab

Google Colab, short for Google Colaboratory, is a cloud-based platform provided by Google for interactive computing, coding, data analysis, and machine learning. It offers a web-based environment where users can create and share Jupyter notebooks, which combine code, text, and multimedia elements. Colab is designed to make it easy for individuals and teams to work on coding and data-related projects without the need for complex local setups. It provides free access to computing resources like GPUs and TPUs, making it particularly valuable for machine learning tasks. Colab also integrates with Google Drive, supports multiple programming languages, and facilitates collaboration among users.

4.3.2 Programming Languages Used

In the development of a model, the decision of which programming language to use is critical since it helps the programmer to convey his or her ideas in a convenient manner.

- Python

Python, a versatile programming language, finds common application in fault diagnosis assignments. Its user-friendly and easily understandable nature expedites the creation of diagnostic algorithms. Python's comprehensive assortment of libraries and frameworks supports efficient examination and visualization of data, which plays a pivotal role in fault identification. Its capacity for dynamic typing facilitates accommodation of various diagnostic data types, while its commitment to strong typing bolsters dependability in fault detection..

B. Scikit-Learn

Another powerful library used in the proposed model is Scikit-Learn. Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.It is built upon some of the familiar technologies like NumPy, pandas, and Matplotlib. The functionality that scikit-learn provides includes:

a. Regression, including Linear and Logistic Regression

b. Classification, including K-Nearest Neighbors

C. Clustering, including K-Means and K-Means++

d. Model selection

e. Preprocessing, including Min-Max Normalisation

4.4 System Testing

System testing is a major procedure done to assess the model's performance. In general, model performance testing comprises running the models against the test dataset and evaluating the outcomes in terms of accuracy, recall, and precision.

4.5 Results

This section includes images of some of the model's intriguing feature graphs, as well as the suggested model's outcome at the end.

Data Exploration: The data analysed in Chapter 3 was sourced from the New IEC publication 60599 and IEC TC 10 databases. This publications contains the five faults found in the electrical compartments and extensive databases of faulty equipment found in the power transformer

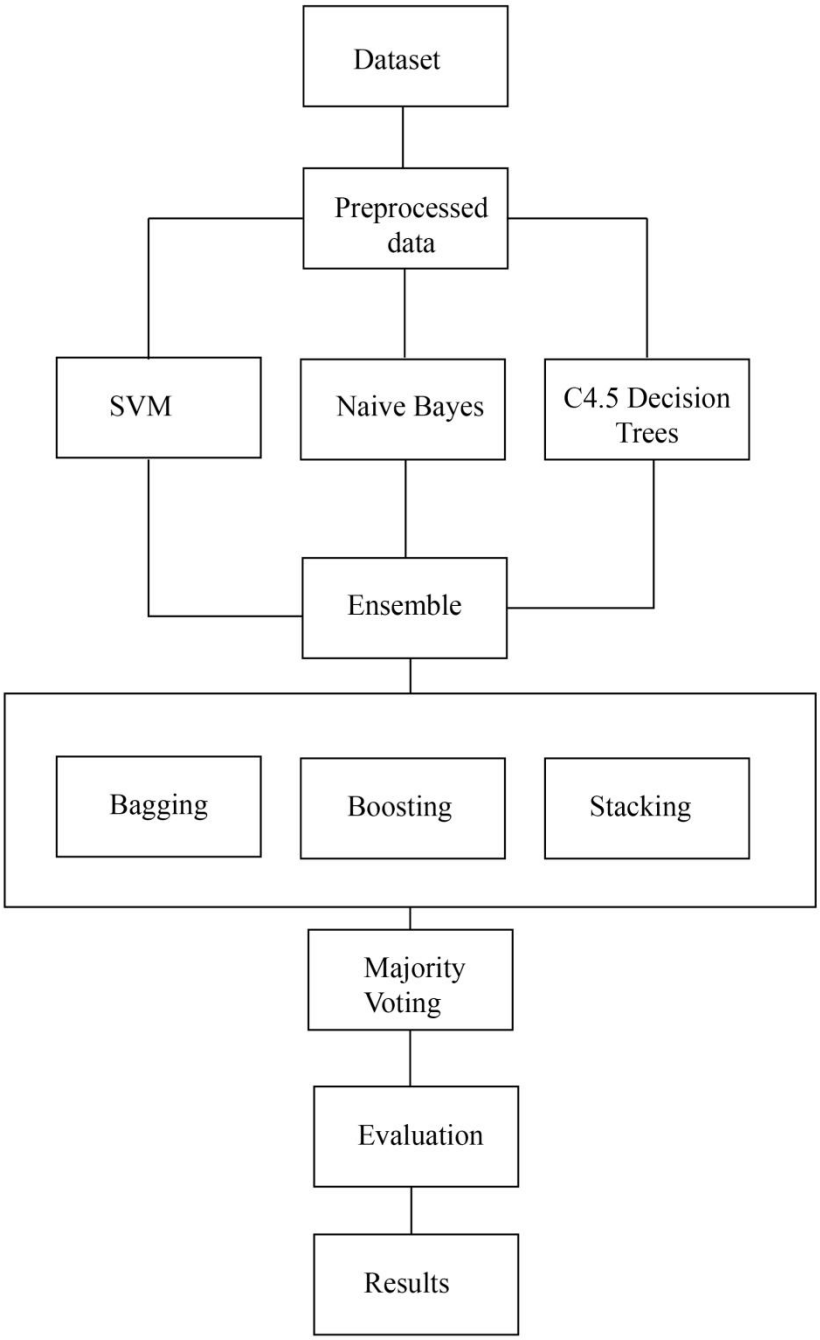

Figure 4.1: Project Architecture

4.6 Dataset

Table 4.1 shows the description of all non-processed or raw datasets used in the project work. All datasets were derived from the New IEC publication 60599 and IEC TC 10 database.The datasets contain values from the following Hydrogen gas (H_2), Methane gas (CH_4), Carbon monoxide gas (CO),Carbon dioxide gas (CO_2), Ethylene gas (C_2H_4) ,Ethane gas (C_2H_6), Acetylene (C_2H_2) and the faults respectively.

Table 4.1 description of original dataset used before normalization

H2	CH4	C2H2	C2H4	C2H6	CO	CO2	fault
32930	2397			157	313	560	PD
37800	1740	8	8	249	56	197	PD
92600	10200				6400	103151	PD
8266	1061			22	107	498	PD
9340	995	7	6	60	60	620	PD
36036	4704	10	5	554	6	347	PD
33046	619		2	58	51	1	PD
40280	1069	1	1	1060	1		PD
26788	18342		27	2111	704		PD
78	20	28	13	11		784	D1
305	100	541	161	33	440	3700	D1
35	6	482	26	3	200	2240	D1
543	120	1880	411	41	76	2800	D1
1230	163	692	233	27	130	115	D1
645	86	317	110	13	74	114	D1
60	10	4	4	4	780	7600	D1
6870	1028	5500	900	79	29	388	D1
10092	5399	37565	6500	530	42	413	D1
650	81	270	51	170	380	2000	D1
210	22	7	6	6	19	74	D1
385	60	159	53	8	465	1250	D1
4230	690	1180	196	5	438	791	D1
7600	1230	1560	836	318	4970	4080	D1
595	80	244	89	9	524	2100	D1
120	25	40	8	1	500	1600	D1
8		101	43		192	4067	D1
6454	2313	6432	2159	121	3628	225	D1
2177	1049	705	440	207	4571	3923	D1
1790	580	619	336	321	956	4250	D1
1330	10	182	66	20	231	1820	D1
4	1	52	7	2	93	519	D1
1900	285	7730	957	31	681	732	D1
57	24	30	27	2	540	2518	D1

1000	500	500	400	1	200	1000	D1
440	89	757	304	19	299	1190	D2
210	43	187	102	12	167	1070	D2
2850	1115	3675	1987	138	2330	4330	D2
7020	1850	4410	2960		2140	1000	D2
545	130	239	153	16	660	2850	D2
7150	1440	1760	1210	97	608	2260	D2
620	325	244	181	38	1480	2530	D2
120	31	94	66		48	271	D2
755	229	460	404	32	845	5580	D2
5100	1430	1010	1140		117	197	D2
13500	6110	4040	4510	212	8690	1460	D2
1570	1110	1830	1780	175	135	602	D2
3090	5020	2540	3800	323	270	400	D2
1820	405	634	365	35	1010	8610	D2
535	160	680	305	16	172	338	D2
13	3	6	3	1	4	51	D2
137	67	104	53	7	196	1678	D2
1084	188	769	166	8	38	199	D2
34	21	56	49	4	95	315	D2
7940	2000	5390	3120	355	1130	285	D2
150	130	30	55	9	120	200	D2
8200	3790	5830	4620	250	31	85	D2
260	215	277	334	35	130	416	D2
75	15	26	14	7	105	322	D2
530	345	250	266	85	3900	20000	D2
60	5	21	21	2	188	2510	D2
90	28	32	31	8	1380	11700	D2
220	77	240	170	22	1800	13800	D2
5900	1500	2300	1200	68	750	335	D2
420	250	800	530	41	300	751	D2
2800	2800	3600	3500	234	92	718	D2
99	170	190	200	20	140	1160	D2
310	230	760	610	54	150	631	D2
800	160	600	260	23	490	690	D2
1500	395	323	395	28	365	576	D2
20000	13000	57000	29000	1850	2600	2430	D2
305	85	130	197	25	813	8380	D2
1900	530	434	83	35	1890	7570	D2
110	62	250	140	90	680	6470	D2
3700	1690	3270	2810	128	22	86	D2
2770	660	763	712	54	522	1490	D2
245	120	167	131	18	829	4250	D2
1170	255	325	312	18	5	1800	D2
4419	3564	2025	2861	668	909	9082	D2
810	580	490	570	111	1100	6800	D2
5000	1200	1100	1000	83	140	265	D2
10000	6730	10400	7330	345	1980	3830	D2

1570	735	1740	1330	87	711	4240	D2
1270	3450	8	1390	520	483	44500	T1&T2
3420	7870	33	6990	1500	573	4640	T1&T2
360	610	9	260	259	12000	74200	T1&T2
1	27	1	4	49	53	254	T1&T2
3675	6392	5	7691	2500	101	833	T1&T2
48	610		10	29	1900	970	T1&T2
12	18		4	4	559	1710	T1&T2
66	60		7	2	76	90	T1&T2
1450	940	61	322	211	2420	3560	T1&T2
	18900	330	540	410	3900	710	T1&T2
960	4000	6	1560	1290	15800	50300	T1&T2
24700	61000	1560	42100	26300	14400	30400	T1&T2
14	44	1	7	124	128	2746	T1&T2
2031	149		3	20	556	3008	T1&T2
480	1075		1132	298	464	1000	T1&T2
40000	400	6	600	70	800	218	T1&T2
8800	64064		95650	72128	290	90300	T3
6709	10500	750	17700	1400	290	1500	T3
1100	1600	26	2010	221		1430	T3
290	966	57	1810	299	72	756	T3
2500	10500	6	13500	4790	530	2310	T3
1860	4980	1600	10700		158	1300	T3
860	1670	40	2050	30	10	690	T3
150	22	11	60	9			T3
400	940	24	820	210	390	1700	T3
6	2990	67	26076	29990	6	26	T3
100	200	11	670	110	100	650	T3
290	1260	8	820	231	228	826	T3
1550	2740	184	5450	816	1140	9360	T3
3910	4290	1230	6040	626	1800	11500	T3
12705	23498	5188	34257	6047	4004	8539	T3
1	8	6	100	8	300	5130	T3
300	700	36	1700	280	760	9250	T3
107	143	2	222	34	193	1330	T3
134	134		45	157	1008	10528	N
100	200	20	200	200	1000	10000	N
	225	3	110	225	785	4500	N
105	125	10	166	71	2000	18000	N
100	50	15	50	65	500	5000	N
100	70	10	170	70	1000	10000	N
150		8	220				N
	224	5	112	224	2150	6720	N
200	50	3	200	50	1000	20000	N
85		70	35	80			N
175		3	375	100			N
80		4	100	200	800		N
150		15	100	200	1000	10000	N

125	100	20	150	100	500	6000	N
200	3		200	50	1000	20000	N
50	30	5					N
100	70	10	170	70	1000	10000	N
95	280	10	150	250	700	8000	N
60	40	3	60	50	540	5100	N
84	79	56	166	52	673	8068	N
66	111	15	110	90	865	12670	N
235	180	336	145	270	672	4256	N
250	150	150	250	150	1000	10000	N
150		150	220				N
200	50	30	200	50	1000	20000	N
134	224	154	224	550	672	3584	N
250		280	150	15			N
150		22	320	80			N
	150	150	200	550	800	5000	N
150		150	200	200	1000	10000	N
200	100	50	100	100	500	10000	N
250	190	180	250	180	1000	10000	N
75	35	80	110	50	400	5300	N
151	131	266	250	73	848	11818	N
8288	1120	19712	1344	784	448	5600	N
31	6	1	3	8	260	705	N
22	7		5	4	180	650	N
80	18	1		20			N
170	16	1		8			N
50	10	5	10	10	1000	10000	N
120		5	10				N
150		3	8				N
85		5	9				N
70		4	30				N
135		4	30				N
1000		16	20				N
6	11	1	3	7	250	800	N
150	120	1	40	130	1100	4000	N
20	30	2	4	25	330	900	N
300	30	2	4	25	330	900	N

Table 4.1 shows the description of all non-processed or raw datasets used in the project work. All datasets were derived from the New IEC publication 60599 and IEC TC 10 database.The datasets contain values from the following Hydrogen gas (H_2), Methane gas (CH_4), Carbon monoxide gas (CO),Carbon dioxide gas (CO_2), Ethylene gas (C_2H_4) ,Ethane gas (C_2H_6), Acetylene (C_2H_2) and the faults respectively.

Table 4.2 Description of Power transformer faults.

Type of fault (s)	Appearances
Partial Discharge (PD)	9
Low Energy(D1)	25
High Energy (D2)	48
Low Thermal Faults <700C (T1&T2)	16
High Thermal Faults >700 (T3)	18
Normal (N)	50

Table 4.3 Description of preprocessed dataset

H2	CH4	C2H2	C2H4	C2H6	CO	CO2	fault
1	0.069938164	0	0.003015626	0.001705809	0.006457705	0.013981541	1
1	0.045829805	0	0	0.006377011	0.00127011	0.005001058	1
0.897643601	0.098272233	0.000300734	0.001261144	0	0.061408019	1	1
1	0.126031053	0.009582727	0.02159146	0	0.010310529	0.057738962	1
1	0.105956717	0.000107135	0	0.005785301	0.005785301	0.065781016	1
1	0.130415476	0.000138769	0	0.015236879	2.78E-05	0.009491826	1
1	0.01870177	0.003026176	3.03E-05	0.001724921	0.001513088	0	1
1	0.026515057	0	0	0.026291616	0	0.05087018	1
1	0.684391465	0.002765218	0	0.077874519	0.025298008	0.075595082	1
0.086675291	0.01164295	0.021992238	0.002587322	0	0.632600259	1	2
0.074175075	0.018271066	0.138532861	0.034905918	0	0.11098991	1	2
0.014304873	0.001341082	0.214126062	0.010281627	0	0.088064372	1	2
0.181949982	0.028633563	0.66654585	0.13410656	0	0.012685756	1	2
1	0.113050707	0.552784705	0.17123857	0	0.085619285	0.073150457	2
1	0.115506329	0.481012658	0.153481013	0	0.096518987	0.159810127	2
0.007372301	0.000789889	0	0	0	0.102159031	1	2
0.185995624	0	0.06345733	0.002188184	0.131291028	0.245076586	1	2
1	0.146031282	0.799736881	0.127320567	0.007308873	0	0.052477708	2
0.267835727	0.14276577	1	0.172107774	0.013005357	0	0.009887269	2
0.307337096	0.015392509	0.112365316	0	0.061056952	0.168804515	1	2
1	0.078431373	0.004901961	0	0	0.06372549	0.333333333	2
0.303542673	0.041867955	0.1215781	0.036231884	0	0.367954911	1	2
1	0.162130178	0.278106509	0.045207101	0	0.102485207	0.186035503	2
1	0.125240319	0.170557539	0.071134304	0	0.638835485	0.516616314	2
0.280248685	0.033955045	0.112386418	0.038259206	0	0.246293639	1	2
0.074421513	0.015009381	0.024390244	0.004377736	0	0.312070044	1	2
0	0.053215078	0.022912047	0.008622814	0.015274698	0.045331362	1	2

1	0.34612348	0.996526133	0.321806411	0	0.553765988	0.016421917	2
0.451420715	0.192942255	0.11411549	0.053391384	0	1	0.851512374	2
0.373886485	0.065920081	0.075846271	0.003817765	0	0.161618733	1	2
0.729281768	0	0.095027624	0.030939227	0.005524862	0.122099448	1	2
0.005791506	0	0.098455598	0.011583012	0.001930502	0.177606178	1	2
0.2427588	0.032991298	1	0.12027536	0	0.084426549	0.091050786	2
0.021860095	0.008744038	0.011128776	0.009936407	0	0.213831479	1	2
1	0.499499499	0.499499499	0.399399399	0	0.199199199	1	2
0.359521776	0.059777968	0.630230572	0.243381725	0	0.23911187	1	3
0.187145558	0.029300567	0.165406427	0.085066163	0	0.146502836	1	3
0.646946565	0.233062977	0.84375	0.441078244	0	0.522900763	1	3
1	0.256115108	0.624460432	0.415827338	0	0.297841727	0.13381295	3
0.186661962	0.040225829	0.078687368	0.048341567	0	0.227240649	1	3
1	0.190415426	0.23578619	0.157805189	0	0.072451439	0.306678009	3
0.233547352	0.115168539	0.082664526	0.057383628	0	0.578651685	1	3
0.370833333	0	0.2625	0.145833333	0.1625	0.070833333	1	3
0.130317231	0.035508291	0.077144917	0.06705119	0	0.146539293	1	3
1	0.270377734	0.186878728	0.212723658	0	0.009343936	0.025248509	3
1	0.443859121	0.28807947	0.323449729	0	0.638019266	0.093919326	3
0.84660767	0.575221239	1	0.970501475	0.02359882	0	0.275516224	3
0.593684211	1	0.477894737	0.743157895	0.011157895	0	0.027368421	3
0.208163265	0.043148688	0.069854227	0.038483965	0	0.113702624	1	3
0.781626506	0.21686747	1	0.435240964	0	0.234939759	0.484939759	3
0.24	0.04	0.1	0.04	0	0.06	1	3
0.077797726	0.035906643	0.058049072	0.027528426	0	0.113105925	1	3
1	0.167286245	0.707249071	0.146840149	0	0.027881041	0.177509294	3
0.096463023	0.054662379	0.167202572	0.144694534	0	0.292604502	1	3
1	0.224036577	0.666884389	0.370346179	0.00914435	0.110385369	0	3
0.738219895	0.633507853	0.109947644	0.240837696	0	0.581151832	1	3
1	0.460154242	0.70987881	0.561757865	0.026808667	0	0.006610356	3
0.590551181	0.472440945	0.635170604	0.784776903	0	0.249343832	1	3
0.215873016	0.025396825	0.06031746	0.022222222	0	0.311111111	1	3
0.022344966	0.013055486	0.008285212	0.009088627	0	0.191564148	1	3
0.023125997	0.001196172	0.007575758	0.007575758	0	0.074162679	1	3
0.007013342	0.001710571	0.002052686	0.001967157	0	0.117345193	1	3
0.014370736	0.003991871	0.015822325	0.010741762	0	0.129046306	1	3
1	0.245541838	0.382716049	0.194101509	0	0.116941015	0.045781893	3
0.499341238	0.275362319	1	0.644268775	0	0.341238472	0.93544137	3
0.771949829	0.771949829	1	0.971493729	0.040478905	0	0.178449259	3
0.069298246	0.131578947	0.149122807	0.157894737	0	0.105263158	1	3
0.362606232	0.249291785	1	0.787535411	0	0.135977337	0.817280453	3
1	0.176319176	0.742599743	0.305019305	0	0.601029601	0.858429858	3
1	0.249320652	0.200407609	0.249320652	0	0.228940217	0.372282609	3
0.329102448	0.202175884	1	0.492293744	0	0.013599275	0.010516772	3

0.033512867	0.007181329	0.012567325	0.020586475	0	0.094314782	1	3
0.247511612	0.065693431	0.052952887	0.006370272	0	0.246184472	1	3
0.007490637	0	0.029338327	0.012172285	0.004369538	0.096441948	1	3
1	0.453507341	0.883088635	0.758020663	0.028820011	0	0.017400761	3
1	0.223122239	0.261045655	0.242268041	0	0.172312224	0.528718704	3
0.053638941	0.024102079	0.03520794	0.026701323	0	0.191635161	1	3
0.64902507	0.139275766	0.178272981	0.171030641	0.00724234	0	1	3
0.445804611	0.344188258	0.161278821	0.260637034	0	0.028642738	1	3
0.104499925	0.070115114	0.056660188	0.068620123	0	0.147854687	1	3
1	0.227171039	0.206833435	0.186495831	0	0.011592434	0.03701444	3
0.960218797	0.635007459	1	0.694679264	0	0.162605669	0.346593734	3
0.357091259	0.156031784	0.398025524	0.29930171	0	0.150252829	1	3
0.02836465	0.077362222	0	0.031061764	0.011507687	0.010676077	1	4
0.432180681	1	0	0.887712135	0.187188975	0.068903917	0.587852495	4
0.004731032	0.008100713	0	0.00338316	0.003369681	0.161623377	1	4
0	0.102766798	0	0.011857708	0.18972332	0.205533597	1	4
0.477491543	0.830991413	0	1	0.324616185	0.012490242	0.107728337	4
0.02010582	0.317460317	0.048148148	0	0.01005291	1	0.507936508	4
0.004689332	0.008206331	0.056858148	0	0	0.325322392	1	4
0.646464646	0.585858586	1	0.050505051	0	0.747474747	0.888888889	4
0.396970563	0.251214633	0	0.074592741	0.042869391	0.674192626	1	4
0	1	0.001344447	0.012637806	0.005646679	0.193331541	0.021780048	4
0.018968465	0.079413051	0	0.030898318	0.025529884	0.314033483	1	4
0.389300135	1	0	0.682032301	0.416218035	0.216016151	0.485195155	4
0.004735883	0.015664845	0	0.002185792	0.044808743	0.046265938	1	4
0.674875208	0.048585691	0.032612313	0	0.005657238	0.184026622	1	4
0.367604268	0.94471387	0	1	0.191076625	0.352085354	0.871968962	4
1	0.009851478	0	0.014852228	0.00160024	0.019852978	0.005300795	4
0.091042292	0.669426158	0	1	0.753822646	0.001978043	0.944007787	5
0.368696152	0.586444572	0.026421597	1	0.063756462	0	0.069500287	5
0.541330645	0.793346774	0	1	0.09828629	0.23891129	0.70766129	5
0.132915003	0.518539646	0	1	0.138049059	0.00855676	0.398745009	5
0.184822884	0.777678968	0	1	0.354527938	0.038832074	0.170742552	5
0.168391345	0.461900282	0.143932267	1	0	0.008278457	0.115710254	5
0.416666667	0.81372549	0.014705882	1	0.009803922	0	0.333333333	5
0.069083782	0.006369427	0.000979912	0.024987751	0	0.240568349	1	5
0.224343675	0.546539379	0	0.474940334	0.11097852	0.218377088	1	5
0	0.099519744	0.002034418	0.869463714	1	0	0.000667022	5
0.135053111	0.286798179	0	1	0.150227618	0.135053111	0.969650986	5
0.225239617	1	0	0.6485623	0.178115016	0.17571885	0.653354633	5
0.148866609	0.278552746	0	0.573888405	0.068875327	0.10418483	1	5
0.302004782	0.336950524	0.055545338	0.497884863	0	0.107963951	1	5
0.287607841	0.644365848	0.039136615	1	0.067530493	0	0.149902489	5
0	0.001364788	0.000974849	0.019302008	0.001364788	0.058295964	1	5

0.028652051	0.07206425	0	0.180594747	0.026481441	0.07857608	1	5
0.079066265	0.106174699	0	0.165662651	0.024096386	0.143825301	1	5
0.008489936	0.008489936	0.005341982	0	0.010683965	0.091863016	1	0
0.008016032	0.018036072	0	0.018036072	0.018036072	0.098196393	1	0
0.067155882	0.049366244	0	0.02379364	0.049366244	0.173893707	1	0
0.005280712	0.00639244	0	0.008671484	0.003390773	0.110617009	1	0
0.017051153	0.007021063	0	0.007021063	0.01003009	0.097291876	1	0
0.009009009	0.006006006	0	0.016016016	0.006006006	0.099099099	1	0
0.069539667	0.105778648	0	0.103819785	0.03036239	0.240940255	1	0
0.044676098	0.032613552	0	0.015934475	0.032613552	0.319434103	1	0
0.009851478	0.002350353	0	0.009851478	0.002350353	0.049857479	1	0
0.024813896	0.093796526	0.017369727	0	0.022332506	0.230769231	1	0
0.084025403	0.107962872	0	0.18172936	0.047386419	0.242794333	1	0
0.03714565	0.107526882	0	0.046920821	0.095796676	0.389051808	1	0
0.01352028	0.020931397	0	0.008512769	0.018527792	0.098647972	1	0
0.017558528	0.013377926	0	0.02173913	0.013377926	0.080267559	1	0
0.009851478	0	0.004900735	0.009851478	0.002350353	0.049857479	1	0
0.02200489	0.012224939	0	0.095354523	0.031784841	0.24205379	1	0
0.009009009	0.006006006	0	0.016016016	0.006006006	0.099099099	1	0
0.010638298	0.03379224	0	0.017521902	0.030037547	0.086357947	1	0
0.011183049	0.007259172	0	0.011183049	0.00922111	0.105356092	1	0
0.003992016	0.003368263	0.000499002	0.014221557	0	0.07747006	1	0
0.004030028	0.007585934	0	0.007506914	0.005926511	0.067167128	1	0
0.021892484	0.008513744	0.046460715	0	0.030406227	0.128192654	1	0
0.010152284	0	0	0.010152284	0	0.086294416	1	0
0.04040404	0.077777778	0.04040404	0.075757576	0	0.217171717	1	0
0.008512769	0.001001502	0	0.008512769	0.001001502	0.048572859	1	0
0	0.026086957	0.005797101	0.026086957	0.12057971	0.155942029	1	0
0.115479115	0.102702703	0.13022113	0.066339066	0	0.238329238	1	0
0.063116371	0.099605523	0	0.146942801	0.028599606	0.235700197	1	0
0.031958763	0	0	0.010309278	0.082474227	0.134020619	1	0
0	0.00751269	0	0.005076142	0.005076142	0.086294416	1	0
0.015075377	0.005025126	0	0.005025126	0.005025126	0.045226131	1	0
0.00712831	0.00101833	0	0.00712831	0	0.083503055	1	0
0.007597341	0	0.008547009	0.014245014	0.002849003	0.069325736	1	0
0.006641124	0.004938272	0.016432524	0.015070243	0	0.065985526	1	0
0.406976744	0.034883721	1	0.046511628	0.01744186	0	0.26744186	0
0.042613636	0.007102273	0	0.002840909	0.009943182	0.367897727	1	0
0.027863777	0.004643963	0.150154799	0.001547988	0	0.27244582	1	0
0.038555393	0.00829673	0	0.097120547	0.009272816	0.243533431	1	0
0.082479258	0.007320644	0	0.097120547	0.003416301	0.243533431	1	0
0.004502251	0.00050025	0	0.00050025	0.00050025	0.099549775	1	0
0.056234719	0.107090465	0	0.002444988	0.031784841	0.24205379	1	0
0.071812408	0.107962872	0	0.002442599	0.032730826	0.242794333	1	0

0.039119804	0.107090465	0	0.00195599	0.031784841	0.24205379	1	0
0.032258065	0.107526882	0	0.012707722	0.032258065	0.242424242	1	0
0.06402737	0.107526882	0	0.012707722	0.032258065	0.242424242	1	0
0.483775811	0.102261554	0	0.001966568	0.026548673	0.237954769	1	0
0.006257822	0.012515645	0	0.002503129	0.007509387	0.311639549	1	0
0.037259315	0.029757439	0	0.009752438	0.032258065	0.274818705	1	0
0.020044543	0.031180401	0	0.002227171	0.025612472	0.365256125	1	0
0.331848552	0.031180401	0	0.002227171	0.025612472	0.365256125	1	0

Table 4.3 shows the description of processed datasets used in the project work. These datasets has been normalized, which further means that the data has been corrected from the noisy data, missing values and it has under gone the min-max normalization so as to attain this stable data set for further evaluation in this work.

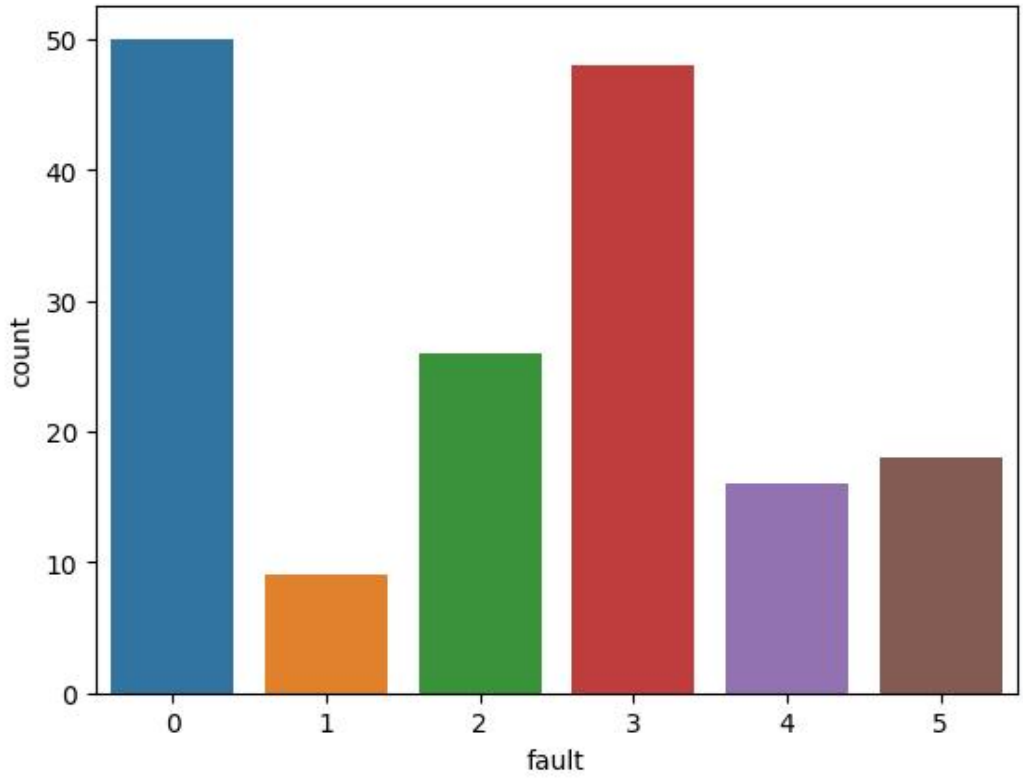

Fig 4.2: fault chart ranking

4.7 Evaluation of classifiers during training and testing phases in percentage (%)

The results for the accuracy, recall and precision are given in tables 4.4 to 5.0. From the results given the Decision Tree classifier presented the highest training score with a 100.00%, 100.00%, 100.00% in precision, accuracy and recall respectively, the next best performing in the training phase is the ensemble boosting with the Naive Bayes which attained 79.65%, 74.14%, 74.14% in Precision, Recall and Accuracy respectively. The classifier with the best testing performance is the Decision Tree classifier with a test score of 60.78%, 60.78%, 59.99% in accuracy, precision and recall respectively, the Majority voting attained a high test score of 62.75%, 67.65%, 62.75% for the accuracy, recall and precision.

The Naive Bayes and the Support Vector Machine classifiers performed the poorest with the Naive Bayes attaining a training percentage of 61.21%, 61.21%, 65.93% in accuracy, recall and precision and a test percentage of 52.94%, 50.00% and 52.94% while for the support vector machine attained a training percentage of 60.34%, 60.34% and 61.33% in accuracy, recall and precision and a test percentage of 56.86%, 45.17% and 56.86%. From the results presented in this work the Decision Tree classifier is the best performing classifier amongst all three classifiers and their ensembles that was tested and trained in this project.

Table 4.4 Support Vector Machine train and test results

	Precision	Recall	Accuracy
Training	60.34%	61.33%	60.34%
Testing	45.17%	56.86%	56.86%

Table 4.5 Naive Bayes train and test results

	Precision	Recall	Accuracy
Training	65.93%	61.21%	61.21%
Testing	50.00%	52.94%	52.94%

Table 4.6 Decision Tree train and test results.

	Precision	Recall	Accuracy
Training	100.00%	100.00%	100.00%
Testing	59.99%	60.78%	60.78%

Table 4.7 Ensemble bagging (SVM) train and test results.

	Precision	Recall	Accuracy
Training	66.67%	62.93%	62.93%
Testing	50.74%	60.78%	60.78%

Table 4.8 Ensemble boosting (NB) train and test results.

	Precision	Recall	Accuracy
Training	79.90%	74.14%	74.14%
Testing	63.86%	52.94%	52.94%

Table 4.9 Ensemble stacking train and test results

	Precision	Recall	Accuracy
Training	72.65%	73.28%	73.28%
Testing	51.68%	60.78%	60.78%

Table 5.0 Majority Voting train and test results

	Precision	Recall	Accuracy
Training	75.44%	67.24%	67.24%
Testing	67.65%	62.75%	62.75%

This confusion matrix is as a result of a multi-class classification problem with six classes (Class 0, Class 1, Class 2, Class 3, Class 4, and Class 5). Each row and column in the matrix corresponds to a class, and the numbers within the matrix represent the counts of samples classified into those classes in the matrix. Where the True Positives (correctly classified as the class), False Negatives (incorrectly classified as other classes when they belong to the class), False Positives (incorrectly classified as the class when they belong to other classes), and True Negatives (correctly classified as other classes) for each class in a multi-class classification are identified below

Table 5.1 Tabular representation of SVM matrix

	True Positive	False Positive	True Negative	False Negative
Class 0	15	0	15	0
Class 1	0	0	15	1
Class 2	0	0	15	12
Class 3	7	4	36	6
Class 4	1	3	72	3
Class 5	6	0	138	0

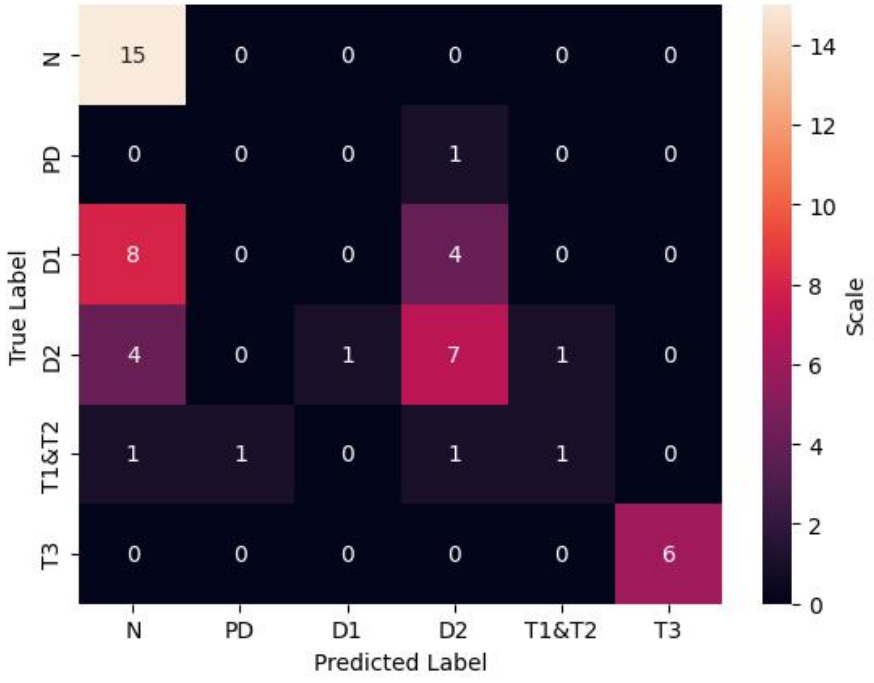

Fig 4.3 Confusion matrix for Support Vector Machine Classifier

This confusion matrix evaluates the performance of a Naive Bayes classifier. It accomplishes this by measuring key metrics such as True Positives, False Negatives, False Positives, and True Negatives for each class, we have TP (instances correctly identified as belonging to the specific class), FN (instances wrongly classified as different classes when they actually belong to the class), FP (instances incorrectly categorized as the specific class when they should belong to other classes), and TP (instances accurately classified as different classes).These metrics provide valuable information regarding the Naive Bayes model's accuracy in classifying data across multiple categories. Below are the values for the metrics values and the figure of the matrix.

Table 5.2 Tabular representation of NB matrix

	True Positive	False Positive	True Negative	False Negative
Class 0	15	12	43	0
Class 1	0	1	68	1
Class 2	3	3	55	9
Class 3	5	7	48	11
Class 4	0	3	65	3
Class 5	4	0	65	2

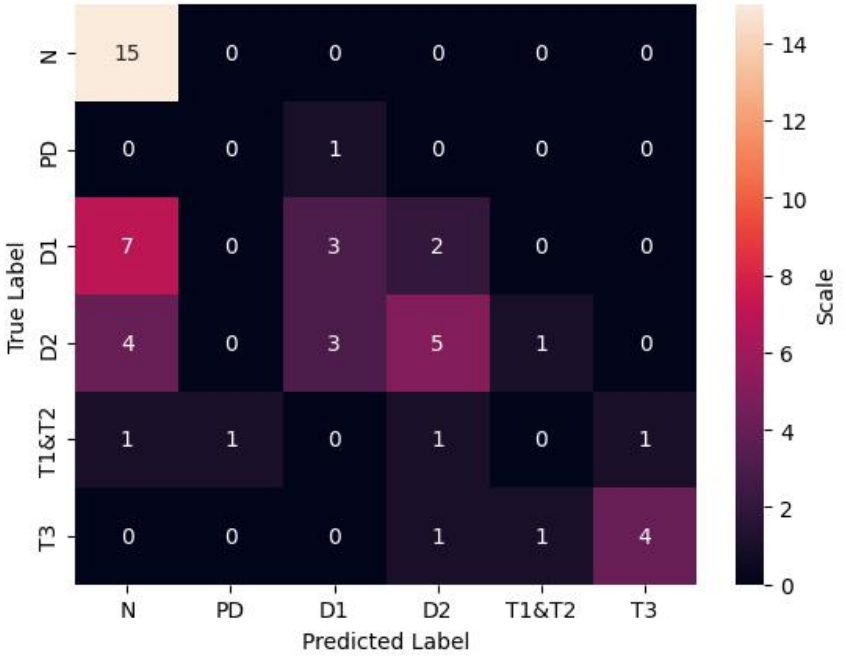

Fig 4.4 Confusion matrix for Naive bayes classifier

This confusion matrix has emerged from a multi-class classification problem encompassing six classes (Class 0 to Class 5). It shows the distribution of samples among these classes within the matrix. It unveils the decision tree model's performance in terms of correctly assigning instances to their respective classes (True Positives), instances erroneously classified as different classes when they should belong to a specific class (False Negatives), instances incorrectly labeled as the target class when they should be in other categories (False Positives), and instances accurately categorized as different classes (True Negatives). This metric values helps evaluates the performance of the decision tree model as it handles multi-class classification tasks.

Table 5.3 Tabular representation of DT matrix results

	True Positive	False Positive	True Negative	False Negative
Class 0	11	3	47	4
Class 1	1	1	68	0
Class 2	5	2	53	10
Class 3	10	10	47	3
Class 4	1	2	64	3
Class 5	4	4	63	1

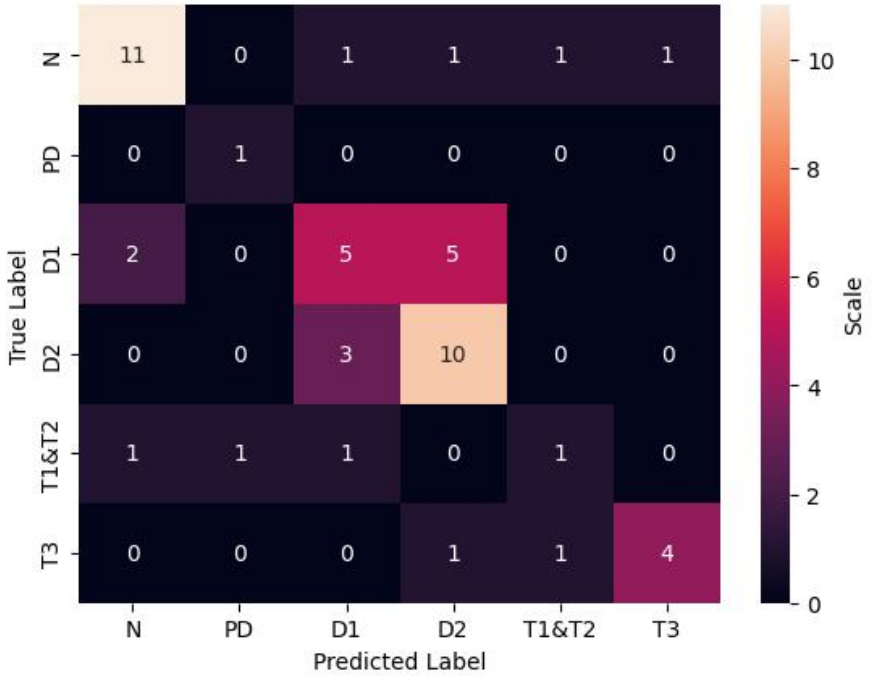

Fig 4.5 Confusion matrix for Decision Tree classifier

This confusion matrix, stemming from a multi-class classification task involves six categories (Class 0 to Class 5), represents the allocation of samples to these classes within the 6 x 6 confusion matrix. This aids in understanding the ensemble bagging model's effectiveness in correctly identifying instances as the specific class (True Positives), instances misclassified as different classes when they should belong to the class (False Negatives), instances incorrectly classified as the class when they should be placed in other classes (False Positives), and instances correctly classified as different classes (True Negatives). This information offers insights into how well ensemble bagging performs in multi-class classification scenarios.

Table 5.4 Tabular representation of Ensemble Bagging matrix results

	True Positive	False Positive	True Negative	False Negative
Class 0	15	9	52	0
Class 1	0	1	74	1
Class 2	0	1	63	12
Class 3	9	8	68	4
Class 4	0	2	71	4
Class 5	6	0	74	1

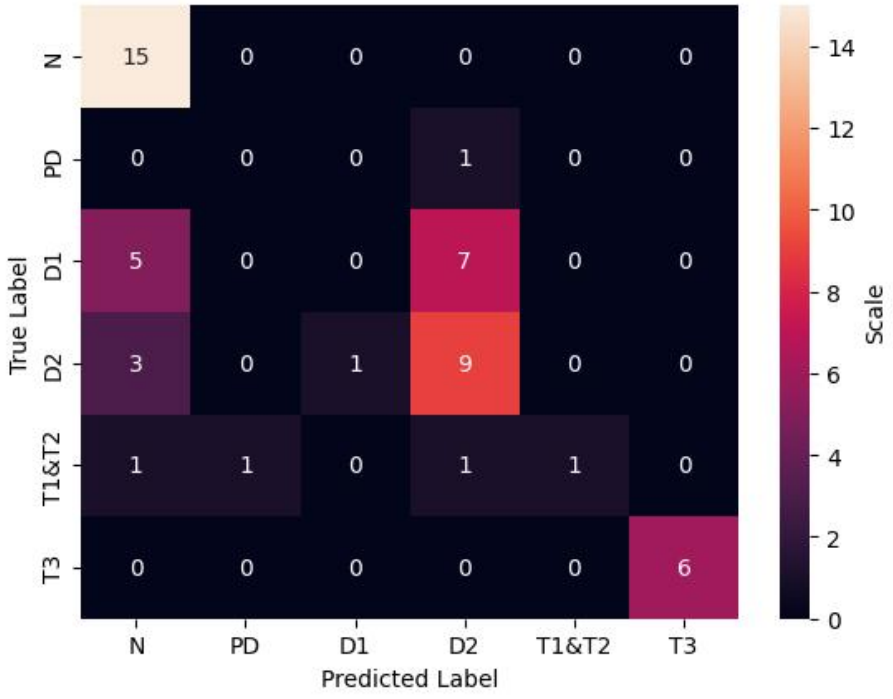

Fig 4.6 Confusion matrix for Ensemble Bagging (SVM)

This confusion matrix has arisen from a multi-class classification problem with six categories (Class 0 to Class 5), illustrating how samples are distributed among these classes within the matrix. When assessing the performance of an ensemble boosting technique, it reveals how well the model operates in correctly assigning instances to their respective classes (True Positives), instances incorrectly classified as different classes when they should belong to a specific class (False Negatives), instances wrongly classified as the target class instead of other categories (False Positives), and instances accurately categorized as different classes (True Negatives). This matrix enhances the evaluation process of the ensemble boosting .

Table 5.5 Tabular representation of Ensemble Boosting matrix

	True Positive	False Positive	True Negative	False Negative
Class 0	13	1	60	3
Class 1	0	1	67	1
Class 2	1	3	63	12
Class 3	12	13	63	1
Class 4	1	6	68	4
Class 5	0	0	63	6

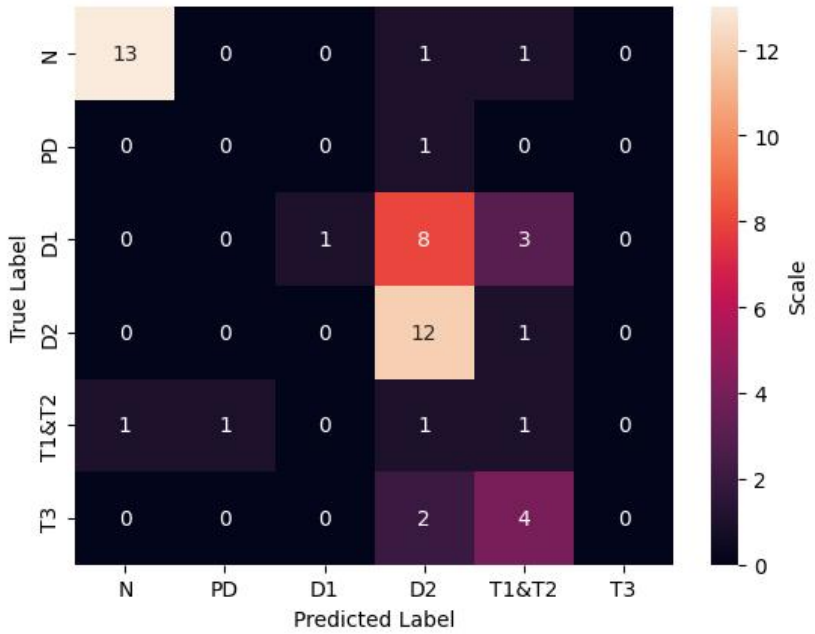

Fig 4.7 Confusion matrix for Ensemble Boosting (NB)

The multi-class classification context with six classes, the confusion matrix offers a holistic assessment of the ensemble stacking model's performance. It includes metrics like True Positives (TP), False Negatives (FN), False Positives (FP), and True Negatives (TN) for all classes. Where (True Positives), instances incorrectly classified as different classes when they should belong to a specific class (False Negatives), instances wrongly classified as the target class instead of other categories (False Positives), and instances accurately categorized as different classes (True Negatives). These metrics collectively provide a comprehensive evaluation, reflecting the model's accuracy, misclassifications, and the ability to make accurate predictions for each class in the multi-class classification problem.

T able 5.6 Tabular representation of Ensemble stacking matrix

	True Positive	False Positive	True Negative	False Negative
Class 0	13	7	84	2
Class 1	0	6	85	1
Class 2	1	2	84	12
Class 3	11	4	82	2
Class 4	1	3	93	3
Class 5	5	1	95	1

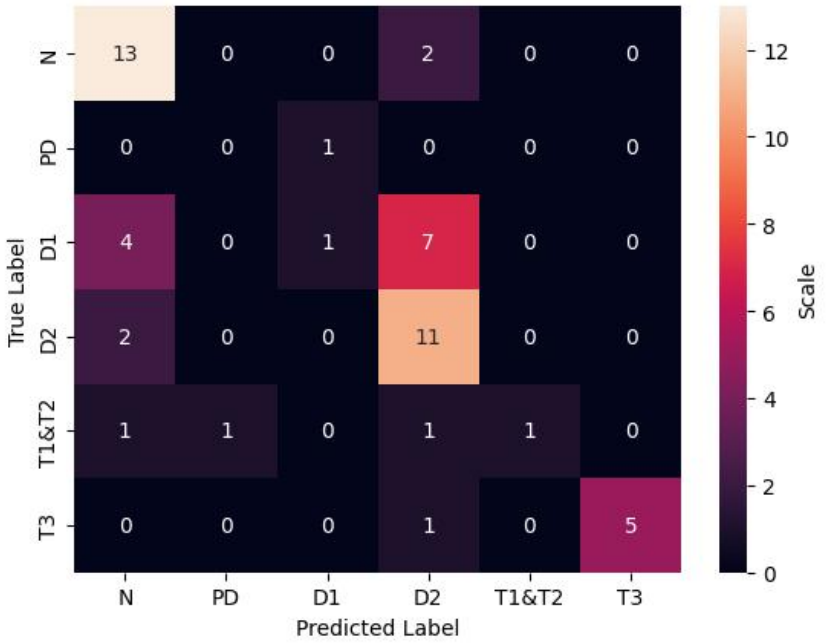

Fig 4.8 Confusion matrix for Ensemble Stacking

This multi-class classification problem with its confusion matrix illustrates how the samples were distributed among the various classes. In relation to majority voting, it highlights the key metrics such as True Positives (samples correctly predicted as their respective classes), False Negatives (samples misclassified as other classes when they belong to a specific class), False Positives (samples incorrectly labeled as a particular class when they should belong to other classes), and True Negatives (samples accurately classified as other classes). The table below illustration the classification of the samples according to its classes.

Table 5.7 Tabular representation of Majority Voting matrix

	True Positive	False Positive	True Negative	False Negative
Class 0	15	12	58	0
Class 1	1	2	82	0
Class 2	2	1	76	6
Class 3	8	4	68	5
Class 4	1	5	77	2
Class 5	4	0	79	2

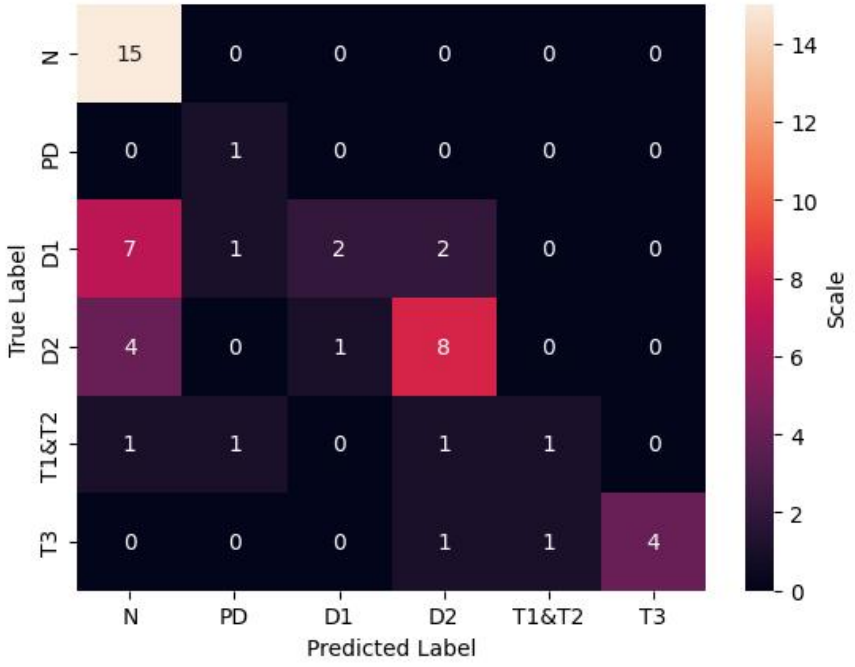

Fig 4.9 Confusion matrix for Majority Voting

The provided visual representations depict the confusion matrices generated for the implemented classifiers. It is evident from these matrices that the models excel in correctly identifying instances without faults, resulting in a high true positive rate for normal cases. However, a noticeable number of misclassifications occur when the models attempt to discern the specific type of faults present. This observation implies that while our model demonstrates sensitivity in detecting the absence of faults, its accuracy in predicting the occurrence and identifying the precise fault type remains limited. To enhance the model's effectiveness, addressing these limitations is crucial, as it can lead to the development of a more robust and efficient system. Such improvements would enable better resource allocation for maintenance and reduce unnecessary downtime for transformers operating under normal conditions.

CHAPTER 5
SUMMARY AND RECOMMENDATION

5.1 Summary

Power transformers are very vital in our world today, it helps in generating enough power for daily running of businesses, homes, schools, factories etc. In Nigeria the power transformer holds a very critical position in the effectiveness of living. In this work I proposed a fault detection technique that detects faults faster and more accurately, prior to now the Dissolved Gas Analysis (DGA) played a good role in detecting faults but it had its drawbacks but now using machine learning techniques like the support Vector Machines, Naive bayes and Decision Trees, it can be rightly said that these present more accurate results.

The Decision Tree Performed the best and is highly recommended for further usage and also as time goes on Further investigation and improvement is possible. In the future, we will tackle this issue to optimize this techniques. In addition, this method might be tested on Larger datesets.

5.2 Recommendation

The results of the study indicate that the proposed classifier technique performance, with accuracy scores attaining a 100.00% training and 60.78% testing. This highlights the potential of the proposed technique to be used as a major diagnosis system by mechanical professionals and others in the diagnosis of Faults in a power transformer and other related field. The implementation of such a technique in this field would help professionals make more accurate and less error prone decisions.

5.3 Future works

Future research could also consider using techniques on a larger dataset This would provide a more comprehensive understanding and in depth knowledge of this technique that could not have been attained in this work. Further, incorporating additional evaluation measures such as F1-score and area under the curve (AUC) would provide a much stronger results.

REFERENCES

Alex R. S., Shigaeki L.L., Osvaldo R. S. (2019) K-NN and Mean-Shift Algorithm Applied in Fault Diagnosis in Power Transformers by DGA, *Intelligent System Application to Power Systems (ISAP). pp. 1-6.*

Arian D., Akhmad F., Benjamin K., Isti S., Andrew K. (2020), Data Driven Fault Diagnosis of Power Transformer using Dissolved Gas Analysis, *International Journal of Technology*, 11(2), pp. 389-397.

Babagana A. D., Ankit P., Arya S., Dr. Jabir A., (2023) Fault Detection and Protection of Power Transformers Using Fuzzy Logic, Vol. 7, pp. 1-12.

Bouchaoui L., Hemsas K.E, Mellech H., Benlahneche S. (2021). Power transformer faults diagnosis using undestructive methods (roger and iec) and artificial neural network for dissolved gas Analysis applied on the functional transformer in the algerian North-eastern: A comparative study

Demirci M., Gozde H. Taplamaciaoglu M. C., (2021). Fault diagnosis of Power Transformers with Machine Learning methods using Traditional Methods Data, *Technical and physical problems of engineering,* Vol.13 pp. 225-230

Dr. Md M.I, Dr. Gareth L., Dr. Hettiwatte S. N., (2018). Application of Parzen Window Estimation for Incipient Fault Diagnosis in Power Transformers , pp 1-18

Hazlee A. I., kai C C., Hazlie M, (2020). Hybrid Feature Selection Artificial Intelligence Gravitational Search Algorithm Technique for Automated Transformer Fault Determination Based on Dissolved Gas Analysis, *Generation, transmission and distribution*, Vol. 4 , pp. 1575-1582

Hongzhe M., Wei Z., Rongru W. & Chungang Y. (2018). A Power Transformers Fault Diagnosis Model Based on Three DGA Ratios and PSO Optimization SVM, *materials science and engineering*. 339, pp 1-4.

Ibrahim B.M., Diaa-Eldin A. M. (2021) Novel Power Transformer Fault Diagnosis using Optimized Machine Learning methods, Vol. 28, pp 740-751

Jackson K., Geoffrey M. (2021). Research Methods in Machine Learning: A Content Analysis, Vol. 10. pp 78-91

Jinzhong L., Qioagen Z., Wang K., Jianyi W, Tianchun Z., Yiyi Z. (2016). Optimal Dissolved Gas Rations Selected by Genetic Algorithm for power Transformer Fault Diagnosis Based on Support Vector Machine, Vol. 23. pp 1198-1204.

Odongo, G., Musabe, R., & Hanyurwimfura, D. (2021). A Multinomial DGA Classifier for Incipient Fault Detection in Oil-Impregnated Power Transformers. Algorithms, 14(4), 128. MDPI AG.

Weiyun M., Bengeng W., Xiangyi X., Lu C., Tianyi W., Zhangrui P., Chen R., (2022).Power Transformer Fault Diagnosis Using Graph Neural Network based on Dissolved Gas Data, *conference series,* pp 1-7.

Rosamaliati, Bernandus A. S. A., Isa H., Ardyono P., Mauridhi H. P. (2022). lassifier Method in Fault Diagnosis of Oil-immersed Power Transformer by considering Dissolved Gas Analysis, *Engineering technology*, Vol. 10, pp. 223-245.

Rosamaliati, Bernandus A. S. A., Isa H., Ardyono P., Mauridhi H. P. (2022). lassifier Method in Fault Diagnosis of Oil-immersed Power Transformer by considering Dissolved Gas Analysis, *Engineering technology*, Vol. 10, pp. 223-245.

Zhanhong W., Mingbiao z., Zhenheng L., XueJen C., & Yonguah H. (2021) Improved Genetic Algorithm and XGBoost Classifier For Power Transformer Fault Diagnosis, Vol. 9, pp 1-8.

APPENDIX A

SOURCE CODE

-


```
  {
 "metadata": {
  "kernelspec": {
   "language": "python",
   "display_name": "Python 3",
   "name": "python3"
  },
  "language_info": {
   "name": "python",
   "version": "3.10.12",
   "mimetype": "text/x-python",
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "pygments_lexer": "ipython3",
   "nbconvert_exporter": "python",
   "file_extension": ".py"
  },
  "colab": {
   "provenance": []
  }
 },
 "nbformat_minor": 0,
 "nbformat": 4,
 "cells": [
  {
   "cell_type": "markdown",
   "source": [
    "## Import libraries"
```

```
   ],
   "metadata": {
    "_uuid": "8f2839f25d086af736a60e9eeb907d3b93b6e0e5",
    "_cell_guid": "b1076dfc-b9ad-4769-8c92-a6c4dae69d19",
    "execution": {
     "iopub.status.busy": "2023-09-09T08:07:48.971029Z",
     "iopub.execute_input": "2023-09-09T08:07:48.971338Z",
     "iopub.status.idle": "2023-09-09T08:07:49.298428Z",
     "shell.execute_reply.started": "2023-09-09T08:07:48.971312Z",
     "shell.execute_reply": "2023-09-09T08:07:49.297222Z"
    },
    "id": "_zI0MHPtUoLb"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "import pandas as pd\n",
    "import seaborn as sns\n",
    "import numpy as np\n",
    "\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.svm import LinearSVC\n",
    "from sklearn.tree import DecisionTreeClassifier, plot_tree\n",
    "import matplotlib.pyplot as pl\n",
    "from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, confusion_matrix\n",
    "from sklearn.naive_bayes import GaussianNB\n",
    "from sklearn.tree import DecisionTreeClassifier\n",
    "from sklearn.multiclass import OneVsRestClassifier\n",
    "from sklearn.neural_network import MLPClassifier\n",
    "from sklearn.linear_model import LogisticRegression\n",
    "from sklearn.ensemble import (\n",
    "    RandomForestClassifier,\n",
```

```
    "    VotingClassifier,\n",
    "    BaggingClassifier,\n",
    "    AdaBoostClassifier,\n",
    "    StackingClassifier,\n",
    ")\n",
    "from sklearn import metrics\n",
    "import matplotlib.pyplot as plt"
   ],
   "metadata": {
    "execution": {
     "iopub.status.busy": "2023-09-09T09:42:13.547413Z",
     "iopub.execute_input": "2023-09-09T09:42:13.547805Z",
     "iopub.status.idle": "2023-09-09T09:42:13.553875Z",
     "shell.execute_reply.started": "2023-09-09T09:42:13.547770Z",
     "shell.execute_reply": "2023-09-09T09:42:13.553241Z"
    },
    "trusted": true,
    "id": "GCiIfHCkUoLg"
   },
   "execution_count": 29,
   "outputs": []
  },
  {
   "cell_type": "code",
   "source": [
    "import warnings\n",
    "warnings.filterwarnings('ignore')"
   ],
   "metadata": {
    "execution": {
     "iopub.status.busy": "2023-09-09T09:42:13.555178Z",
     "iopub.execute_input": "2023-09-09T09:42:13.556046Z",
     "iopub.status.idle": "2023-09-09T09:42:13.569917Z",
     "shell.execute_reply.started": "2023-09-09T09:42:13.555989Z",
```

```
      "shell.execute_reply": "2023-09-09T09:42:13.568930Z"
    },
    "trusted": true,
    "id": "Ibqrs8FsUoLi"
  },
  "execution_count": 30,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "from google.colab import drive\n",
    "drive.mount ('/content/drive')\n",
    "# Load the dataset\n",
    "data = pd.read_csv('/content/drive/My Drive/preprocessed.csv')\n"
  ],
  "metadata": {
    "execution": {
      "iopub.status.busy": "2023-09-09T09:42:13.571651Z",
      "iopub.execute_input": "2023-09-09T09:42:13.572064Z",
      "iopub.status.idle": "2023-09-09T09:42:13.607421Z",
      "shell.execute_reply.started": "2023-09-09T09:42:13.572028Z",
      "shell.execute_reply": "2023-09-09T09:42:13.606360Z"
    },
    "trusted": true,
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "grKXja-UUoLi",
    "outputId": "712f72c6-692d-4ae8-abcc-c684163028c3"
  },
  "execution_count": 31,
  "outputs": [
    {
```

```
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount(\"/content/drive\", force_remount=True).\n"
      ]
     }
    ]
   },
   {
    "cell_type": "code",
    "source": [
     "# split dataset into training and testing datasets\n",
     "train, test = train_test_split(data, test_size=0.3, random_state=42)\n",
     "\n",
     "# extract feature\n",
     "features = train.columns.to_list()[1:-1]\n",
     "features"
    ],
    "metadata": {
     "execution": {
      "iopub.status.busy": "2023-09-09T09:42:13.608892Z",
      "iopub.execute_input": "2023-09-09T09:42:13.609457Z",
      "iopub.status.idle": "2023-09-09T09:42:13.617814Z",
      "shell.execute_reply.started": "2023-09-09T09:42:13.609428Z",
      "shell.execute_reply": "2023-09-09T09:42:13.616882Z"
     },
     "trusted": true,
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "id": "Invh_jGOUoLk",
     "outputId": "1b69d10a-84cc-4124-cc68-ceccf6f756a3"
    },
```

```
  "execution_count": 32,
  "outputs": [
   {
    "output_type": "execute_result",
    "data": {
     "text/plain": [
      "['H2', 'CH4', 'C2H2', 'C2H4', 'C2H6', 'CO', 'CO2']"
     ]
    },
    "metadata": {},
    "execution_count": 32
   }
  ]
 },
 {
  "cell_type": "code",
  "source": [
   "# initialise training data\n",
   "X_train = train.loc[:, features]\n",
   "y_train = train.fault\n",
   "\n",
   "X_test = test.loc[:, features]\n",
   "y_test = test.fault\n",
   "\n",
   "sns.countplot(x='fault', data=data)\n",
   "plt.show()\n"
  ],
  "metadata": {
   "execution": {
    "iopub.status.busy": "2023-09-09T09:42:13.619901Z",
    "iopub.execute_input": "2023-09-09T09:42:13.620417Z",
    "iopub.status.idle": "2023-09-09T09:42:13.633517Z",
    "shell.execute_reply.started": "2023-09-09T09:42:13.620388Z",
    "shell.execute_reply": "2023-09-09T09:42:13.631959Z"
```

```
      },
      "trusted": true,
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 449
      },
      "id": "AZSg757WUoLl",
      "outputId": "2d952350-7b43-459c-b8a7-3ca5411030a8"
    },
    "execution_count": 33,
    "outputs": [
      {
        "output_type": "display_data",
        "data": {
          "text/plain": [
            "<Figure size 640x480 with 1 Axes>"
          ],
n"
        },
        "metadata": {}
      }
    ]
  },
  {
    "cell_type": "code",
    "source": [
      "X_train.info()"
    ],
    "metadata": {
      "execution": {
        "iopub.status.busy": "2023-09-09T09:42:13.638464Z",
        "iopub.execute_input": "2023-09-09T09:42:13.638813Z",
        "iopub.status.idle": "2023-09-09T09:42:13.654653Z",
        "shell.execute_reply.started": "2023-09-09T09:42:13.638787Z",
```

```
      "shell.execute_reply": "2023-09-09T09:42:13.653903Z"
    },
    "trusted": true,
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "VSRgkultUoLl",
    "outputId": "2c592d81-c04c-4447-90aa-e7f9ad2d2b51"
  },
  "execution_count": 34,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "<class 'pandas.core.frame.DataFrame'>\n",
        "Int64Index: 116 entries, 95 to 102\n",
        "Data columns (total 7 columns):\n",
        " #   Column  Non-Null Count  Dtype  \n",
        "---  ------  --------------  -----  \n",
        " 0   H2      116 non-null    float64\n",
        " 1   CH4     116 non-null    float64\n",
        " 2   C2H2    116 non-null    float64\n",
        " 3   C2H4    116 non-null    float64\n",
        " 4   C2H6    116 non-null    float64\n",
        " 5   CO      116 non-null    float64\n",
        " 6   CO2     116 non-null    float64\n",
        "dtypes: float64(7)\n",
        "memory usage: 7.2 KB\n"
      ]
    }
  ]
},
{
```

```
"cell_type": "code",
"source": [
 "y_train.info()"
],
"metadata": {
 "execution": {
  "iopub.status.busy": "2023-09-09T09:42:13.656213Z",
  "iopub.execute_input": "2023-09-09T09:42:13.656500Z",
  "iopub.status.idle": "2023-09-09T09:42:13.673305Z",
  "shell.execute_reply.started": "2023-09-09T09:42:13.656475Z",
  "shell.execute_reply": "2023-09-09T09:42:13.671969Z"
 },
 "trusted": true,
 "colab": {
  "base_uri": "https://localhost:8080/"
 },
 "id": "O1lUYXglUoLm",
 "outputId": "66c414eb-6708-4486-a05d-4cd4c1956559"
},
"execution_count": 35,
"outputs": [
 {
  "output_type": "stream",
  "name": "stdout",
  "text": [
   "<class 'pandas.core.series.Series'>\n",
   "Int64Index: 116 entries, 95 to 102\n",
   "Series name: fault\n",
   "Non-Null Count  Dtype\n",
   "--------------  -----\n",
   "116 non-null    int64\n",
   "dtypes: int64(1)\n",
   "memory usage: 1.8 KB\n"
  ]
```

```
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "# function to evaluate the respective models\n",
    "evals = dict()\n",
    "def evaluate_classifier(model, name, X_train, X_test, y_train, y_test):\n",
    "    train_accuracy = metrics.accuracy_score(y_train, model.predict(X_train))\n",
    "    test_accuracy = metrics.accuracy_score(y_test, model.predict(X_test))\n",
    "\n",
    "    train_precision = metrics.precision_score(y_train, model.predict(X_train), average='weighted')\n",
    "    test_precision = metrics.precision_score(y_test, model.predict(X_test), average='weighted')\n",
    "\n",
    "    train_recall = metrics.recall_score(y_train, model.predict(X_train), average='weighted')\n",
    "    test_recall = metrics.recall_score(y_test, model.predict(X_test), average='weighted')\n",
    "\n",
    "    evals[str(name)] = [train_accuracy, test_accuracy, train_precision, test_precision, train_recall, test_recall]\n",
    "    print(f\"{name} : Accuracy [Training] {train_accuracy:.2%} | [Test] {test_accuracy:.2%}\")\n",
    "    print(f\"{name} : Precision [Training] {train_precision:.2%} | [Test] {test_precision:.2%}\")\n",
    "    print(f\"{name} : Recall [Training] {train_recall:.2%} | [Test] {test_recall:.2%}\")\n",
    "\n",
    "    actual = y_test\n",
    "    predicted = model.predict(X_test)\n",
    "    confusion_matrix = metrics.confusion_matrix(actual, predicted)\n",
```

```
      "\n",
      "    labels = ['N', 'PD', 'D1', 'D2', 'T1&T2', 'T3']\n",
      "\n",
      "          ax = sns.heatmap(confusion_matrix, annot=True,cbar_kws={'label': 'Scale'})\n",
      "    ax.set_xticklabels(labels)\n",
      "    ax.set_yticklabels(labels)\n",
      "\n",
      "    ax.set(ylabel=\"True Label\", xlabel=\"Predicted Label\")\n",
      "\n",
      "    ax.grid(False)\n",
      "\n",
      "    plt.savefig(f'{name}-cm.jpg')"
     ],
     "metadata": {
      "execution": {
       "iopub.status.busy": "2023-09-09T09:42:13.674694Z",
       "iopub.execute_input": "2023-09-09T09:42:13.675657Z",
       "iopub.status.idle": "2023-09-09T09:42:13.688296Z",
       "shell.execute_reply.started": "2023-09-09T09:42:13.675628Z",
       "shell.execute_reply": "2023-09-09T09:42:13.686694Z"
      },
      "trusted": true,
      "id": "czxO97vaUoLm"
     },
     "execution_count": 36,
     "outputs": []
    },
    {
     "cell_type": "code",
     "source": [
      "# train support vector machine classifier\n",
      "svm_model = OneVsRestClassifier(LinearSVC(dual=False)).fit(X_train, y_train)\n",
```

```
      "evaluate_classifier(svm_model, \"Support Vector Machine\", X_train, X_test, y_train, y_test)"
     ],
     "metadata": {
      "execution": {
       "iopub.status.busy": "2023-09-09T09:42:13.723112Z",
       "iopub.execute_input": "2023-09-09T09:42:13.723665Z",
       "iopub.status.idle": "2023-09-09T09:42:14.164178Z",
       "shell.execute_reply.started": "2023-09-09T09:42:13.723636Z",
       "shell.execute_reply": "2023-09-09T09:42:14.162892Z"
      },
      "trusted": true,
      "colab": {
       "base_uri": "https://localhost:8080/",
       "height": 504
      },
      "id": "aKVAKonDUoLo",
      "outputId": "65637d4e-fb37-41bf-8e50-787ae3540586"
     },
     "execution_count": 37,
     "outputs": [
      {
       "output_type": "stream",
       "name": "stdout",
       "text": [
        "Support Vector Machine : Accuracy [Training] 60.34% | [Test] 56.86%\n",
        "Support Vector Machine : Precision [Training] 61.33% | [Test] 45.17%\n",
        "Support Vector Machine : Recall [Training] 60.34% | [Test] 56.86%\n"
       ]
      },
      {
       "output_type": "display_data",
       "data": {
        "text/plain": [
```

```
            "<Figure size 640x480 with 2 Axes>"
           ],
          },
          "metadata": {}
         }
        ]
       },
       {
        "cell_type": "code",
        "source": [
         "# Create a Gaussian Naive Bayes classifier\n",
         "nb_classifier = GaussianNB()\n",
         "\n",
         "# Train the classifier on the training data\n",
         "nb_classifier.fit(X_train, y_train)\n",
         "\n",
         "evaluate_classifier(nb_classifier, \"naive bayes\", X_train, X_test, y_train, y_test)\n",
         "\n",
         "y_true = np.array([1, 0, 1, 1, 0, 1, 0, 0, 1, 0])  # Actual labels\n",
         "y_pred = np.array([1, 0, 0, 1, 1, 1, 0, 1, 0, 0])  # Predicted labels\n",
         "\n",
         "\n",
         "TN = np.sum((y_true == 0) & (y_pred == 0))\n",
         "\n",
         "# Calculate False Negatives (FN)\n",
         "FN = np.sum((y_true == 1) & (y_pred == 0))\n",
         "\n",
         "# Calculate False Positives (FP)\n",
         "FP = np.sum((y_true == 0) & (y_pred == 1))\n",
         "\n",
         "print(\"True Negatives:\", TN)\n",
         "print(\"False Negatives:\", FN)\n",
         "print(\"False Positives:\", FP)\n",
```

```
  "\n",
  "\n",
  "\n",
  "\n",
  "\n",
  "\n",
  "\n"
],
"metadata": {
  "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 558
  },
  "id": "ssngOVTHag__",
  "outputId": "e8223716-720f-4906-ccdc-1a3e950d365d"
},
"execution_count": 38,
"outputs": [
  {
    "output_type": "stream",
    "name": "stdout",
    "text": [
      "naive bayes : Accuracy [Training] 61.21% | [Test] 52.94%\n",
      "naive bayes : Precision [Training] 65.93% | [Test] 50.00%\n",
      "naive bayes : Recall [Training] 61.21% | [Test] 52.94%\n",
      "True Negatives: 3\n",
      "False Negatives: 2\n",
      "False Positives: 2\n"
    ]
  },
  {
    "output_type": "display_data",
    "data": {
      "text/plain": [
```

```
            "<Figure size 640x480 with 2 Axes>"
           ],
           },
          "metadata": {}
         }
        ]
       },
       {
        "cell_type": "code",
        "source": [
         "#create c4.5 decisiontree classifier\n",
         "clf = DecisionTreeClassifier(criterion='entropy')  # C4.5 uses information gain (entropy)\n",
         "clf.fit(X_train, y_train)\n",
         "\n",
         "# Train the classifier on the training data\n",
         "clf.fit(X_train, y_train)\n",
         "evaluate_classifier(clf, \"dt\", X_train, X_test, y_train, y_test)\n",
         "\n",
         "\n",
         "# Make predictions on the test data\n",
         "y_pred = clf.predict(X_test)\n",
         "\n",
         "# Calculate accuracy\n",
         "accuracy = accuracy_score(y_test, y_pred)\n",
         "\n",
         "# Calculate precision\n",
         "precision = precision_score(y_test, y_pred, average='weighted')\n",
         "\n",
         "# Calculate recall\n",
         "recall = recall_score(y_test, y_pred, average='weighted')\n",
         "\n",
         "# Calculate F1-score\n",
         "f1 = f1_score(y_test, y_pred, average='weighted')\n",
```

```
      "\n",
      "# Generate a confusion matrix\n",
      "confusion = confusion_matrix(y_test, y_pred)\n",
      "\n",
      "print(f\"Accuracy: {accuracy}\")\n",
      "print(f\"Precision: {precision}\")\n",
      "print(f\"Recall: {recall}\")\n",
      "print(f\"F1 Score: {f1}\")\n",
      "print(\"Confusion Matrix:\")\n",
      "print(confusion)\n",
      "\n",
      "plt.figure(figsize=(12, 8))\n",
      "plot_tree(clf, filled=True, feature_names=X_train.columns)  # Adjust feature names if needed\n",
      "plt.title(\"C4.5 Decision Tree\")\n",
      "plt.show()\n"
    ],
    "metadata": {
      "colab": {
        "base_uri": "https://localhost:8080/",
        "height": 1000
      },
      "id": "g1vQL903cesA",
      "outputId": "cdf2c925-6b12-46bc-b436-e7a6d5e656e2"
    },
    "execution_count": 39,
    "outputs": [
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "dt : Accuracy [Training] 100.00% | [Test] 56.86%\n",
          "dt : Precision [Training] 100.00% | [Test] 56.69%\n",
          "dt : Recall [Training] 100.00% | [Test] 56.86%\n",
```

```
        "Accuracy: 0.5686274509803921\n",
        "Precision: 0.5669117647058823\n",
        "Recall: 0.5686274509803921\n",
        "F1 Score: 0.5605521065967314\n",
        "Confusion Matrix:\n",
        "[[11  0  1  1  1  1]\n",
        " [ 0  1  0  0  0  0]\n",
        " [ 2  0  3  5  2  0]\n",
        " [ 1  0  3  9  0  0]\n",
        " [ 1  1  1  0  1  0]\n",
        " [ 0  0  0  1  1  4]]\n"
       ]
      },
      {
       "output_type": "display_data",
       "data": {
        "text/plain": [
         "<Figure size 640x480 with 2 Axes>"
        ],
3       },
       "metadata": {}
      }
     ]
    },
    {
     "cell_type": "code",
     "source": [
      "# build bagging ensemble model\n",
      "bag_ensemble = BaggingClassifier(\n",
      "    estimator=svm_model, n_estimators=10, random_state=0\n",
      ").fit(X_train, y_train)\n",
      "evaluate_classifier(bag_ensemble, \"Bagging Ensemble\", X_train, X_test, y_train, y_test)\n",
      "\n",
```

```
  "y_true = np.array([1, 0, 1, 1, 0, 1, 0, 0, 1, 0])  # Actual labels\n",
  "y_pred = np.array([1, 0, 0, 1, 1, 1, 0, 1, 0, 0])  # Predicted labels\n",
  "\n",
  "\n",
  "TN = np.sum((y_true == 0) & (y_pred == 0))\n",
  "\n",
  "# Calculate False Negatives (FN)\n",
  "FN = np.sum((y_true == 1) & (y_pred == 0))\n",
  "\n",
  "# Calculate False Positives (FP)\n",
  "FP = np.sum((y_true == 0) & (y_pred == 1))\n",
  "\n",
  "print(\"True Negatives:\", TN)\n",
  "print(\"False Negatives:\", FN)\n",
  "print(\"False Positives:\", FP)"
],
"metadata": {
  "execution": {
    "iopub.status.busy": "2023-09-09T09:42:16.352509Z",
    "iopub.execute_input": "2023-09-09T09:42:16.352911Z",
    "iopub.status.idle": "2023-09-09T09:42:16.861595Z",
    "shell.execute_reply.started": "2023-09-09T09:42:16.352890Z",
    "shell.execute_reply": "2023-09-09T09:42:16.860751Z"
  },
  "trusted": true,
  "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 558
  },
  "id": "CUi8cxi3UoLp",
  "outputId": "1702f281-3483-4052-98b1-684214588b92"
},
"execution_count": 40,
"outputs": [
```

```
      {
       "output_type": "stream",
       "name": "stdout",
       "text": [
         "Bagging Ensemble : Accuracy [Training] 62.93% | [Test] 60.78%\n",
         "Bagging Ensemble : Precision [Training] 66.67% | [Test] 50.74%\n",
         "Bagging Ensemble : Recall [Training] 62.93% | [Test] 60.78%\n",
         "True Negatives: 3\n",
         "False Negatives: 2\n",
         "False Positives: 2\n"
       ]
      },
      {
       "output_type": "display_data",
       "data": {
         "text/plain": [
           "<Figure size 640x480 with 2 Axes>"
         ],
         "image/png": "i        },
       "metadata": {}
      }
     ]
    },
    {
     "cell_type": "code",
     "source": [
       "# create boosting ensemble model\n",
       "ada_ensemble = AdaBoostClassifier(\n",
       "    estimator=nb_classifier, n_estimators=100, random_state=0\n",
       ").fit(X_train, y_train)\n",
       "\n",
       "evaluate_classifier(ada_ensemble, \"Adaboost Ensemble\", X_train, X_test,
y_train, y_test)\n",
       "\n",
```

```
  "y_true = np.array([1, 0, 1, 1, 0, 1, 0, 0, 1, 0])  # Actual labels\n",
  "y_pred = np.array([1, 0, 0, 1, 1, 1, 0, 1, 0, 0])  # Predicted labels\n",
  "\n",
  "\n",
  "TN = np.sum((y_true == 0) & (y_pred == 0))\n",
  "\n",
  "# Calculate False Negatives (FN)\n",
  "FN = np.sum((y_true == 1) & (y_pred == 0))\n",
  "\n",
  "# Calculate False Positives (FP)\n",
  "FP = np.sum((y_true == 0) & (y_pred == 1))\n",
  "\n",
  "print(\"True Negatives:\", TN)\n",
  "print(\"False Negatives:\", FN)\n",
  "print(\"False Positives:\", FP)\n",
  "\n"
],
"metadata": {
  "execution": {
    "iopub.status.busy": "2023-09-09T09:42:16.863443Z",
    "iopub.execute_input": "2023-09-09T09:42:16.864237Z",
    "iopub.status.idle": "2023-09-09T09:42:33.453115Z",
    "shell.execute_reply.started": "2023-09-09T09:42:16.864208Z",
    "shell.execute_reply": "2023-09-09T09:42:33.452185Z"
  },
  "trusted": true,
  "colab": {
    "base_uri": "https://localhost:8080/",
    "height": 558
  },
  "id": "2p5ckeugUoLq",
  "outputId": "292977b5-f4da-486a-91d7-b9d836d5d61e"
},
"execution_count": 41,
```

```
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Adaboost Ensemble : Accuracy [Training] 74.14% | [Test] 52.94%\n",
      "Adaboost Ensemble : Precision [Training] 79.90% | [Test] 63.86%\n",
      "Adaboost Ensemble : Recall [Training] 74.14% | [Test] 52.94%\n",
      "True Negatives: 3\n",
      "False Negatives: 2\n",
      "False Positives: 2\n"
     ]
    },
    {
     "output_type": "display_data",
     "data": {
      "text/plain": [
       "<Figure size 640x480 with 2 Axes>"
      ],
     },
     "metadata": {}
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "# create stacking ensemble model\n",
    "stack_ensemble = StackingClassifier(\n",
    "        estimators=[ ('dt' , clf ), ('svm', svm_model), ('nb', nb_classifier)],
final_estimator=LogisticRegression()\n",
    ").fit(X_train, y_train)\n",
    "\n",
```

```
"evaluate_classifier(stack_ensemble, \"Stack Ensemble\", X_train, X_test, y_train, y_test)\n",
"\n",
"_true = np.array([1, 0, 1, 1, 0, 1, 0, 0, 1, 0])  # Actual labels\n",
"y_pred = np.array([1, 0, 0, 1, 1, 1, 0, 1, 0, 0])  # Predicted labels\n",
"\n",
"\n",
"TN = np.sum((y_true == 0) & (y_pred == 0))\n",
"\n",
"# Calculate False Negatives (FN)\n",
"FN = np.sum((y_true == 1) & (y_pred == 0))\n",
"\n",
"# Calculate False Positives (FP)\n",
"FP = np.sum((y_true == 0) & (y_pred == 1))\n",
"\n",
"print(\"True Negatives:\", TN)\n",
"print(\"False Negatives:\", FN)\n",
"print(\"False Positives:\", FP)"
],
"metadata": {
"execution": {
"iopub.status.busy": "2023-09-09T09:42:33.454257Z",
"iopub.execute_input": "2023-09-09T09:42:33.457158Z",
"iopub.status.idle": "2023-09-09T09:42:41.962000Z",
"shell.execute_reply.started": "2023-09-09T09:42:33.457112Z",
"shell.execute_reply": "2023-09-09T09:42:41.961088Z"
},
"trusted": true,
"colab": {
"base_uri": "https://localhost:8080/",
"height": 558
},
"id": "cgEH-qOcUoLr",
"outputId": "ab5b1f1a-6efb-4ff3-ff0e-a6c290bd7a6f"
```

```
   },
   "execution_count": 20,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Stack Ensemble : Accuracy [Training] 75.00% | [Test] 60.78%\n",
      "Stack Ensemble : Precision [Training] 75.21% | [Test] 63.24%\n",
      "Stack Ensemble : Recall [Training] 75.00% | [Test] 60.78%\n",
      "True Negatives: 3\n",
      "False Negatives: 2\n",
      "False Positives: 2\n"
     ]
    },
    {
     "output_type": "display_data",
     "data": {
      "text/plain": [
       "<Figure size 640x480 with 2 Axes>"
      ],
     },
     "metadata": {}
    }
   ]
  },
  {
   "cell_type": "code",
   "source": [
    "# create voting classifier\n",
    "voting_ensemble = VotingClassifier(\n",
    "                  estimators=[('dt' , clf ), ('svm', svm_model), ('nb', nb_classifier)],voting=\"hard\"\n",
    ").fit(X_train, y_train)\n",
```

```
"\n",
"evaluate_classifier(voting_ensemble, \"Voting Ensemble\", X_train, X_test, y_train, y_test)\n",
"\n",
"_true = np.array([1, 0, 1, 1, 0, 1, 0, 0, 1, 0])  # Actual labels\n",
"y_pred = np.array([1, 0, 0, 1, 1, 1, 0, 1, 0, 0])  # Predicted labels\n",
"\n",
"\n",
"TN = np.sum((y_true == 0) & (y_pred == 0))\n",
"\n",
"# Calculate False Negatives (FN)\n",
"FN = np.sum((y_true == 1) & (y_pred == 0))\n",
"\n",
"# Calculate False Positives (FP)\n",
"FP = np.sum((y_true == 0) & (y_pred == 1))\n",
"\n",
"print(\"True Negatives:\", TN)\n",
"print(\"False Negatives:\", FN)\n",
"print(\"False Positives:\", FP)"
],
"metadata": {
"execution": {
"iopub.status.busy": "2023-09-09T09:42:41.963419Z",
"iopub.execute_input": "2023-09-09T09:42:41.964614Z",
"iopub.status.idle": "2023-09-09T09:42:44.676948Z",
"shell.execute_reply.started": "2023-09-09T09:42:41.964582Z",
"shell.execute_reply": "2023-09-09T09:42:44.676054Z"
},
"trusted": true,
"colab": {
"base_uri": "https://localhost:8080/",
"height": 558
},
"id": "TZDDTkhoUoLt",
```

```
      "outputId": "4a5ab60f-a7c5-4dfc-ad3e-3afa8ecc0435"
     },
     "execution_count": 21,
     "outputs": [
      {
       "output_type": "stream",
       "name": "stdout",
       "text": [
        "Voting Ensemble : Accuracy [Training] 67.24% | [Test] 60.78%\n",
        "Voting Ensemble : Precision [Training] 75.44% | [Test] 65.36%\n",
        "Voting Ensemble : Recall [Training] 67.24% | [Test] 60.78%\n",
        "True Negatives: 3\n",
        "False Negatives: 2\n",
        "False Positives: 2\n"
       ]
      },
      {
       "output_type": "display_data",
       "data": {
        "text/plain": [
         "<Figure size 640x480 with 2 Axes>"
        ],
        "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAiYAAAGwCAYAAACdGa6FAAAAOXRF
       },
       "metadata": {}
      }
     ]
    },
    {
     "cell_type": "code",
     "source": [
      "evals"
     ],
```

```
"metadata": {
  "execution": {
    "iopub.status.busy": "2023-09-09T09:42:44.678251Z",
    "iopub.execute_input": "2023-09-09T09:42:44.678926Z",
    "iopub.status.idle": "2023-09-09T09:42:44.684273Z",
    "shell.execute_reply.started": "2023-09-09T09:42:44.678897Z",
    "shell.execute_reply": "2023-09-09T09:42:44.683441Z"
  },
  "trusted": true,
  "colab": {
    "base_uri": "https://localhost:8080/"
  },
  "id": "m3qlhQ9DUoLt",
  "outputId": "28c1fa97-cab9-4916-b4ca-f95d134643bb"
},
"execution_count": null,
"outputs": [
  {
    "output_type": "execute_result",
    "data": {
      "text/plain": [
        "{'Support Vector Machine': [0.603448275862069,\n",
        "  0.5686274509803921,\n",
        "  0.6132572408434478,\n",
        "  0.45168067226890757,\n",
        "  0.603448275862069,\n",
        "  0.5686274509803921],\n",
        " 'naive bayes': [0.6120689655172413,\n",
        "  0.5294117647058824,\n",
        "  0.6593483004487062,\n",
        "  0.4999688764394647,\n",
        "  0.6120689655172413,\n",
        "  0.5294117647058824],\n",
        " 'Adaboost Ensemble': [0.7413793103448276,\n",
```

```
            "  0.5294117647058824,\n",
            "  0.798951432534964,\n",
            "  0.6385994397759104,\n",
            "  0.7413793103448276,\n",
            "  0.5294117647058824],\n",
            " 'Stack Ensemble': [0.7327586206896551,\n",
            "  0.6078431372549019,\n",
            "  0.7265102411200415,\n",
            "  0.5167642814701638,\n",
            "  0.7327586206896551,\n",
            "  0.6078431372549019],\n",
            " 'Voting Ensemble': [0.6724137931034483,\n",
            "  0.6274509803921569,\n",
            "  0.7543843608498781,\n",
            "  0.6764705882352942,\n",
            "  0.6724137931034483,\n",
            "  0.6274509803921569]}"
          ]
        },
        "metadata": {},
        "execution_count": 30
      }
    ]
  }
 ]
}
```