MODULAR DATASET ON SENSOR ANOMALY SUPPORTING MACHINE LEARNING RESEARCH IN MANUFACTURING SYSTEM

CO8811 – PROJECT REPORT

Submitted by

SANTHOSH KUMAR B 211420118043

THARUN J P 211420118053

in partial fulfillment for the award the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER AND COMMUNICATION ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

BONAFIDE CERTIFICATE

Certified that this project report **"MODULAR DATASET ON SENSOR ANOMALY SUPPORTING MACHINE LEARNING RESEARCH IN MANUFACTURING SYSTEM"** is the Bonafide work of **SANTHOSH KUMAR B (211420118043) AND THARUN J P (211420118053)** who carried out the project work under my supervision.

SIGNATURE

Dr.B.ANNI PRINCY M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR,
COMPUTER AND COMMUNICATION ENGINEERING,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI, POONAMALLEE,
CHENNAI- 600123.

SIGNATURE

Dr.V.MUTHU M.Tech.,(Ph.D.).,

SUPERVISOR

ASSOCIATE PROFESSOR,
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING,
PANIMALAR ENGINEERING COLLEGE,
NAZARATHPETTAI, POONAMALLEE,
CHENNAI- 600123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors Tmt. **C.VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR, M.E., Ph.D**., and **Dr.SARANYASREE SAKTHIKUMAR B.E., M.B.A., Ph.D**., for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.MANI, M.E., Ph.D.,** for his timely concern and encouragement provided to us throughout the course.

We thank our HOD of Computer and Communication Engineering Department, **Dr. B. ANNI PRINCY, M.E., Ph.D**., Professor, for the support extended throughout the project.

We would like to thank our supervisor, **Dr. V. MUTHU, M.Tech., (Ph.D.)**., Associate Professor, and all the faculty members of the Department of Artificial intelligence and Machine learning for their advice and suggestions for the successful completion of the project.

SANTHOSH KUMAR B

THARUN J P

ABSTRACT

Detecting small deviations in manufacturing systems is crucial to prevent substantial economic losses. The conventional industrial practice involves simplistic brute-force checking of a limited set of parameters with predefined pessimistic bounds. To enhance efficiency, employing machine learning techniques can refine monitored parameters, establish more accurate bounds, and predict potential issues. However, progress is impeded by the scarcity of realistic datasets mimicking manufacturing system behaviors. This paper presents MDAS (Modular Dataset on Anomalies in Sensors) tailored for machine learning research in analog sensor data. MDAS is crafted through a modular manufacturing simulation environment, offering evaluations with four supervised machine learning algorithms. MDAS is produced through a modular manufacturing simulation environment that simulates a specific production process. Evaluations with four supervised machine learning algorithms reveal the multilayer perceptron as the most suitable for both accuracy and execution time. The dataset and code are made publicly available, facilitating further research to advance the state of the art.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

CNN	Convolutional Neural Network
LCNN	Lookup based Convolutional Neural Network
RNN	Recurrent Neural Network
DEX	Dalvik Executables
TCP	Transmission Control Protocol
IP	Internet Protocol
HTTP	Hyper Text Transfer Protocol
ADT	Android Development

CHAPTER 1

INTRODUCTION

Aim:

The aim of creating a modular ice cream factory dataset on anomalies in sensors to support machine learning research in manufacturing systems is to provide a valuable resource for the development and evaluation of machine learning models and anomaly detection algorithms in the context of manufacturing.

The usage of appropriate machine learning techniques can be very valuable in this context to narrow down the set of parameters to monitor, define more refined bounds, and forecast impending issues. One of the factors hampering progress in this field is the lack of datasets that can realistically mimic the behaviors of manufacturing systems. In this paper, we propose a new dataset called MDAS (Modular Ice cream factory Dataset on Anomalies in Sensors) to support machine learning research in analog sensor data. MDAS is created using a modular manufacturing simulation environment that simulates the ice cream-making process. Using MDAS, we evaluated three different supervised machine learning algorithms (Logistic Regression, Decision Tree and Random Forest) fortwo different problems: anomaly detection and anomaly classification. The results showed that Decision Tree is the most suitable algorithm with respect to model accuracy and execution time. Anomaly detection, also known as outlier detection, is a crucial task in machine learning and data mining, aimed at identifying patterns or instances that deviate significantly from the norm within a dataset. The anomalies often represent events, observations, or data points that differ from the expected behavior or distribution. This field finds

applications across various domains, including cybersecurity, finance, healthcare, manufacturing, and network monitoring, where detecting unusual behavior or events is critical for maintaining system integrity, security, and performance.

The traditional methods for anomaly detection often rely on rule-based systems or statistical techniques, which may have limitations in capturing complex patterns or adapting to dynamic environments. With the advancements in machine learning, anomaly detection has witnessed a paradigm shift towards data-driven approaches, leveraging algorithms such as supervised, unsupervised, and semi-supervised learning.

Machine learning algorithms commonly used for anomaly detection include clustering algorithms like k-means, density-based methods such as isolation forest and DBSCAN, Gaussian mixture models, autoencoders for deep learning-based anomaly detection, and ensemble methods like isolation forests and one-class SVMs.In this introduction, we will explore the fundamental concepts of anomaly detection using machine learning techniques, discuss its significance across various domains, and highlight the shift towards data-driven approaches for more effective anomaly detection in complex and dynamic environments.

1.2 EXISTING SYSTEM

This research aims to create an effective prediction model using different types of ML methods to detect anomalies in Ice cream factory. First of all the datasets are collected, and then the preprocessing is accomplished via the missing values imputation. The Mean Value Imputation (MVI) method is used to impute the

missing values of the dataset. Then, the categorical feature values are converted to their equivalent numerical values using the One Hot Encoding (OHE) technique. Shows that all datasets used in this work have a object and numerical features as converted into numerical features is used to alleviate this issue. After completing the initial preprocessing, the datasets feature values are scaled using different Feature Scaling techniques. The result showed that Multilayer Perceptron is the most suitable with respect to accuracy. But the accuracy is less. Then now we create a new system for better anomaly prediction. So now we move on to the proposed system.

1.3 PROPOSED SYSTEM

This research aims to create an effective prediction model using different types of ML methods to detect anomalies in Ice cream factory. First of all, the datasets are collected, and then the preprocessing is accomplished via the missing values imputation and Create an instance of RFE with the classifier and the desired number of features to select Logistic Regression classification of modeling and performance evaluation. Then we are using Decision Tree, Logistic regression and Random Forest Algorithm for prediction accuracy .Decision Tree gives a best results with respect to high accuracy. Compare to existing system our new system gives results are more Accuracy.

CHAPTER 2

LITERATURE SURVEY

1. **Title:** Anomaly Detection in Industrial IoT: A Survey
    - **Authors:** Mohammadi, Mohammadreza, et al. (2019)

    - **Description:** This comprehensive survey explores various approaches to anomaly detection in Industrial Internet of Things (IIoT) environments, which are highly relevant to manufacturing systems. The authors delve into different techniques, including statistical methods, machine learning algorithms, and deep learning architectures. They discuss challenges specific to anomaly detection in industrial settings, such as data heterogeneity and the need for real-time detection. Additionally, the survey provides insights into available datasets and evaluation metrics crucial for benchmarking anomaly detection algorithms in manufacturing systems.

2. **Title:** Anomaly Detection in Industrial IoT: A Review
    - **Authors:** Xu, Jin, et al. (2020)

    - **Description:** Focusing on anomaly detection within Industrial IoT (IIoT) frameworks, this review paper examines the latest advancements in detection techniques and their applications in manufacturing systems. The authors discuss traditional methods such as statistical process control

(SPC) and newer approaches like machine learning and deep learning-based anomaly detection. They highlight the challenges associated with deploying anomaly detection systems in real-world manufacturing environments, such as interpretability and scalability. Furthermore, the review discusses the importance of feature engineering and model interpretability in ensuring the effectiveness of anomaly detection solutions.

3. **Title:** Anomaly Detection in Cyber-Physical Systems: A Survey
    - **Authors:** Mahmud, Shamim Hossain, et al. (2021)

    - **Description:** This survey paper provides a comprehensive overview of anomaly detection techniques in cyber-physical systems (CPS), which are integral components of modern manufacturing systems. The authors cover various aspects of anomaly detection, including traditional rule-based methods, statistical approaches, and machine learning algorithms. They also discuss the challenges of anomaly detection in CPS, such as dealing with noisy sensor data and ensuring the reliability of anomaly detection systems in dynamic environments. Moreover, the survey highlights the importance of anomaly interpretation and feedback mechanisms for enhancing the resilience of manufacturing systems.

4. **Title:** Anomaly Detection in Smart Manufacturing Systems: A Review
    - **Authors**: Wang, Yuxi, et al. (2020)

- **Description:** Focusing specifically on anomaly detection in smart manufacturing systems, this review paper provides insights into the latest advancements in detection techniques and their applications. The authors explore various anomaly detection methods, including statistical process monitoring, machine learning-based approaches, and hybrid models. They discuss the role of data preprocessing, feature selection, and model validation in developing effective anomaly detection systems for smart manufacturing. Furthermore, the review emphasizes the importance of integrating anomaly detection with other components of smart manufacturing systems, such as predictive maintenance and quality control, to achieve holistic system optimization.

5. **Title:** Deep Learning for Anomaly Detection: A Review
    - **Authors:** Chalapathy, R., et al. (2019)

    - **Description:** This review paper focuses on the application of deep learning techniques for anomaly detection, a critical aspect of modern manufacturing systems. The authors provide a comprehensive overview of deep learning architectures used for anomaly detection, including autoencoders, recurrent neural networks (RNNs), and convolutional neural networks (CNNs). They discuss the advantages of deep learning approaches in handling complex data patterns and achieving high detection accuracy. Additionally, the review covers challenges such as model interpretability, scalability, and the need for labeled data in training deep learning-based anomaly detection models.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 INTRODUCTION

In a world where most of the commodities are mass produced by companies using automated manufacturing systems, the quality of those systems is of vital importance. Even a small deviation in parts of the system could potentially result in bad or malfunctioning products leading to customer dissatisfaction, environmental impacts or huge economic losses to the industry. This is the main reason why all components and sensors in the system have to be continuously monitored to identify anomalies, and prompt remedial actions should be provided if something goes wrong. In a generic sense, an ‘anomaly’ is a deviation from expected behavior, and can occur for different reasons, including faults in the system or its configuration, or due to unanticipated external interference. Such interference, or even some system or configuration faults, belong to the realm of cybersecurity threats if the root cause is an act of bad intention. Regardless of the cause, the consequences of anomalies must be kept at acceptable levels. In the context of this paper, anomalies are data points or patterns in the data that deviate from normal behavior. Anomalies might be induced in the data for a variety of reasons, such as malicious activity or system failure [1]. goal of anomaly detection is to find those data points based on the knowledge gained from previous observations. There are two different problems to

solve when detecting anomalous behavior: Anomaly Detection (AD) and Anomaly Classification (AC). AD is the process of detecting whether a behavior is deviating from the normal one, and AC is the process of indicating which type of anomaly is happening when more than one type is possible. One of the widely used methods for AD and AC is Machine Learning (ML). The ability of ML techniques to build the model automatically based on the given training data makes them an ideal candidate for solving AD and AC problems. During the years, many ML algorithms were used for this purpose with considerable success [1], [2], [3]. Some of these algorithms are Multilayer Perceptron [4], Decision Tree [5], Support Vector Machine [6], [7] or Random Forest [4], [5]. Though there exists a plethora of ML techniques and a multitude of research publications describing their applications in certain contexts, often the designers of industrial systems are overwhelmed and perplexed by the difficult question of how to choose a specific method that is relevant for their problem at hand. In general, the applicability of ML methods is dependent on multiple characteristics of the data which necessitates a two-dimensional approach for answering the above challenge: *a*) a relevant dataset that captures the context of the application and *b*) a proof of applicability of selected ML techniques on the given dataset. These two dimensions are interrelated and must be solved together.

3.2 Architecture Diagram

The architecture for the Modular Dataset on Sensor Anomalies in Manufacturing Systems includes data acquisition, preprocessing, anomaly detection models (like Decision Trees and SVM), model evaluation, visualization, and continuous learning. It supports machine learning research by providing standardized data and tools for effective anomaly detection in manufacturing environments.

Architecture Diagram

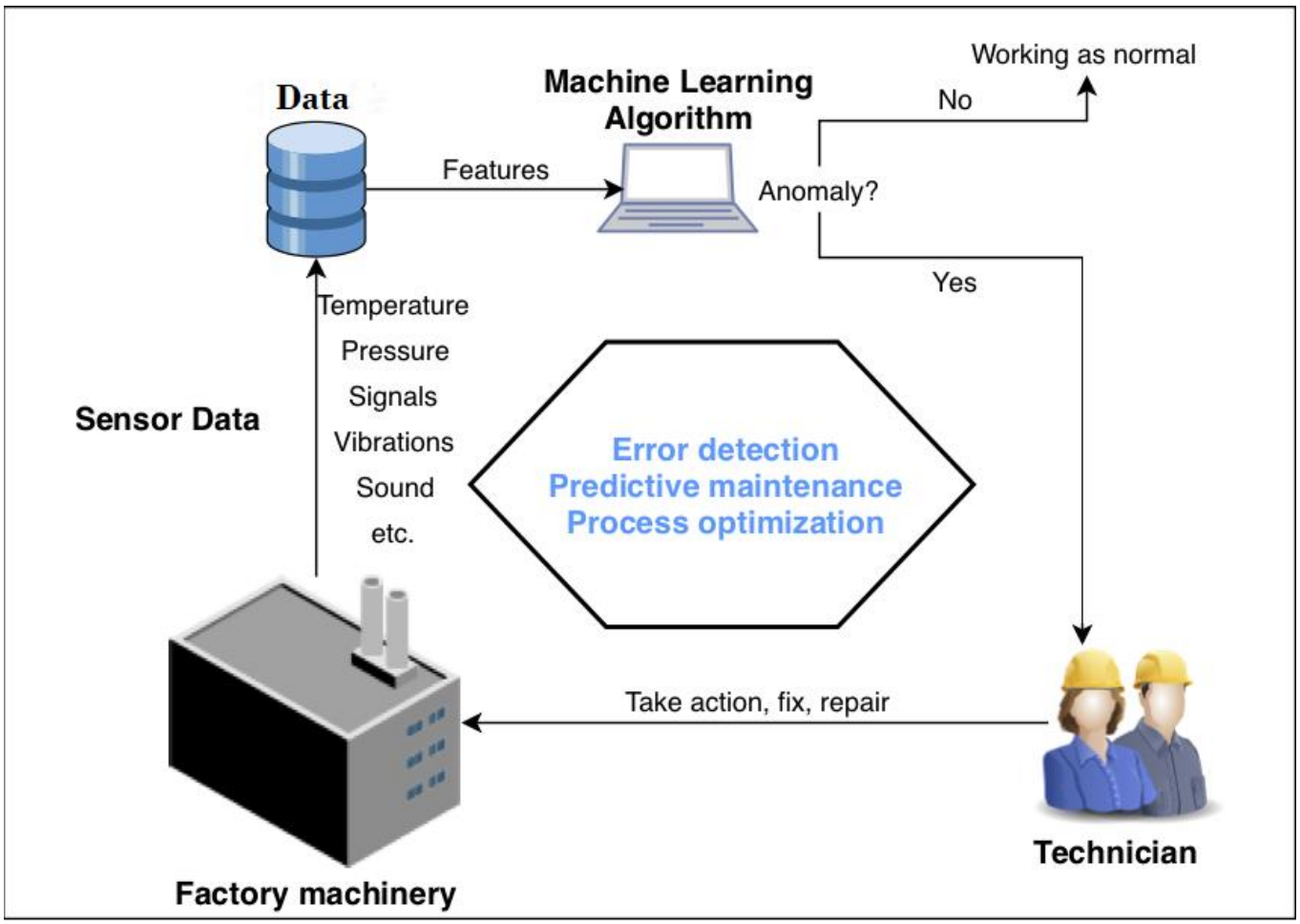

FIG 3.2.1 ARCHITECTURE DIAGRAM

SYSTEM DESIGN

MODULES

- Data Pre-Processing
- Algorithm Implementation
- Prediction

MODULE EXPLANATION

Data Pre-Processing

We collect the Modular Ice cream factory Dataset on Anomalies in Sensors from the publicly available repositories: Kaggle and Github. Normal, Freeze, Ramp and Step these are the data sets that classify the three different anomalies from the sensors. The preprocessing is accomplished via the missing values imputation,Then the categorical feature values are converted to their equivalent numerical values using the One Hot Encoding (OHE) technique. In the dataset there is unwanted columns are presented, these columns are removed and get a neat and clean dataset to predict more accuracy.

Algorithm Implementation

The Classification Algorithms to produce the best results. Weare using Decision Tree, Logistic regression and Random Forest Algorithm to predict anomalies by using MIDAS dataset using ML.On an analysis conducted within various algorithms, the Decision Tree was found to provide highest efficiency. Then, the classifiers are applied to each clustered dataset in order to estimate its performance. The best performing models are identified from the above results based on their low rate of error.

- Decision Tree
- Logistic Regression
- Random Forest

Decision Trees (DTs) are a non-parametric supervised learning method used for classification. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

A decision tree is a supervised machine learning algorithm used for both classification and regression tasks. Its working process can be broken down into several steps:

1. The first step in building a decision tree is to gather the dataset that contains features (attributes) and corresponding labels (classifications or outcomes). Each data point in the dataset represents an instance with its features and associated label.
2. The decision tree algorithm selects the best feature to split the data into subsets. It evaluates different features based on criteria such as entropy or Gini impurity to determine the feature that provides the most information gain or reduction in impurity when splitting the dataset.
3. Once the best feature is selected, the dataset is split into subsets based on the values of that feature. Each subset represents a branch or node in the decision tree. This process continues recursively for each subset until a stopping criterion is met, such as reaching a maximum tree depth or having a minimum number of instances in a node.
4. The decision tree algorithm determines when to stop splitting based on predefined conditions, such as reaching a maximum tree depth, having a minimum number of instances in a node, or when further splitting does not significantly improve the purity of the subsets.
5. After the decision tree is fully grown, pruning techniques may be applied to reduce the size of the tree and prevent overfitting. Pruning involves removing

branches or nodes that do not significantly improve the performance of the tree on unseen data.

6. Once the decision tree is constructed, it can be used to make predictions on new data. To classify a new instance, it traverses the tree from the root node to a leaf node based on the feature values of the instance. The label associated with the leaf node reached by the instance determines its classification.

Overall, the decision tree algorithm recursively partitions the feature space into subsets based on the values of features, aiming to create a tree structure that accurately predicts the labels of unseen instances. It is a transparent and interpretable model, making it useful for understanding decision-making processes in machine learning tasks.

Logistic regression is a statistical method used for binary classification problems, where the output variable (dependent variable) can take only two possible values, usually represented as 0 or 1. It's called "regression" because it estimates the probability of the outcome (0 or 1) based on one or more predictor variables (independent variables).

For instance, in the example below, decision trees learn from data to approximate a sine curve with a set of if-then-else decision rules. The deeper the tree, the more complex the decision rules and the fitter the model.

Prediction

The anomalies detections are classified in to four types that’s are Normal, Freeze, Ramp and Step. Several standard performance metrics such as accuracy, precision and error in classification have been considered for the computation of performance efficacy of this model.Preprocessed data are trained and input given by

the user goes to the trained dataset and predict the results that the accurate classification.

Machine Learning Introduction

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task.Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning tasks

Machine learning tasks are classified into several broad categories. In supervised learning, the algorithm builds a mathematical model from a set of data

that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback. Semi algorithms develop mathematical models from incomplete training data, where a portion of the sample input doesn't have labels.Classification algorithms and regression algorithms are types of supervised learning. Classification algorithms are used when the outputs are restricted to a limited set of values. For a classification algorithm that filters emails, the input would be an incoming email, and the output would be the name of the folder in which to file the email. For an algorithm that identifies spam emails, the output would be the prediction of either "spam" or "not spam", represented by the Boolean values true and false. Regression algorithms are named for their continuous outputs, meaning they may have any value within a range. Examples of a continuous value are the temperature, length, or price of an object.

Active learning algorithms access the desired outputs (training labels) for a limited set of inputs based on a budget and optimize the choice of inputs for which it will acquire training labels. When used interactively, these can be presented to a human user for labeling. Reinforcement learning algorithms are given feedback in the form of positive or negative reinforcement in a dynamic environment and are used in autonomous vehicles or in learning to play a game against a human opponent. Other specialized algorithms in machine learning include topic modeling, where the computer program is given a set of natural language documents and finds other documents that cover similar topics. Meta learning algorithms learn their own inductive bias based on previous experience. In developmental robotics, robot

learning algorithms generate their own sequences of learning experiences, also known as a curriculum, to cumulatively acquire new skills through self-guided exploration and social interaction with humans. These robots use guidance mechanisms such as active learning, maturation, motor synergies, and imitation.

Supervised learning

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It

has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

In the case of semi-supervised learning algorithms, some of the training examples are missing training labels, but they can nevertheless be used to improve the quality of a model. In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

Unsupervised learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Supervised learning is a machine learning paradigm where the algorithm learns from labeled data, consisting of input-output pairs. The algorithm aims to learn a mapping function from the input variables to the output variable, given a dataset containing examples of inputs along with their corresponding correct outputs. During the training phase, the algorithm learns from the labeled data to build a predictive model. The model aims to generalize from the training examples to

make accurate predictions on unseen data. The learning algorithm adjusts its parameters iteratively to minimize the discrepancy between predicted outputs and actual labels in the training dataset.

Types of Supervised Learning

Classification: In classification tasks, the goal is to predict the discrete class label of instances. For example, given an image, the task might involve classifying it into categories such as "cat" or "dog."Regression: In regression tasks, the goal is to predict a continuous numerical value. For instance, predicting the price of a house based on its features like size, number of bedrooms, etc.Structured Prediction: This involves predicting structured objects as output, such as sequences (e.g., natural language sentences), trees, or graphs.

Once the model is trained, it needs to be evaluated to assess its performance on unseen data. Common evaluation metrics vary depending on the task; for classification, metrics like accuracy, precision, recall, F1-score are often used, while for regression tasks, metrics like mean squared error (MSE), mean absolute error (MAE), or R-squared are common.Overfitting and Underfitting: Supervised learning models are prone to overfitting, where the model learns to memorize the training data instead of generalizing from it. To address this, techniques such as regularization, cross-validation, and early stopping are employed. Underfitting, on the other hand, occurs when the model is too simple to capture the underlying structure of theCluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more pre designated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the

structure of the data, often defined by some similarity metric and evaluated, for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

Semi-supervised learning

Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy.

Semi-supervised learning is a machine learning paradigm that falls between supervised learning (where all training data is labeled) and unsupervised learning (where no labeled data is available). In semi-supervised learning, the algorithm learns from a combination of labeled and unlabeled data, leveraging the information contained in both types of data to improve learning performance.

Labeled and Unlabeled Data In semi-supervised learning, the training dataset contains a small portion of labeled examples, similar to supervised learning, as well as a larger portion of unlabeled examples. The labeled data provides explicit supervision for learning, while the unlabeled data provides additional information that can help in generalization and improving model performance.

Regularize the learning process: By imposing constraints on the model's behavior based on the distribution of the unlabeled data, the model is encouraged to produce outputs that are consistent with the unlabeled examples

4.4 Design and Implementation Constraints

4.4.1 Constraints in Analysis

- Constraints as Informal Text
- Constraints as Operational Restrictions
- Constraints Integrated in Existing Model Concepts
- Constraints as a Separate Concept
- Constraints Implied by the Model Structure

4.4.2 Constraints in Design

- Determination of the Involved Classes
- Determination of the Involved Objects
- Determination of the Involved Actions
- Determination of the Require Clauses
- Global actions and Constraint Realization

4.4.3 Constraints in Implementation

A hierarchical structuring of relations may result in more classes and a more complicated structure to implement. Therefore it is advisable to transform the hierarchical relation structure to a simpler structure such as a classical flat one. It is rather straightforward to transform the developed hierarchical model into a bipartite, flat model, consisting of classes on the one

hand and flat relations on the other. Flat relations are preferred at the design level for reasons of simplicity and implementation ease. There is no identity or functionality associated with a flat relation. A flat relation corresponds with the relation concept of entity-relationship modeling and many object oriented methods.

Other Nonfunctional Requirements

The application at this side controls and communicates with the following three main general components.

- embedded browser in charge of the navigation and accessing to the web service;
- Server Tier: The server side contains the main parts of the functionality of the proposed architecture. The components at this tier are the following.

Web Server, Security Module, Server-Side Capturing Engine, Preprocessing Engine, Database System, Verification Engine, Output Module.

Safety Requirements

1. The software may be safety-critical. If so, there are issues associated with its integrity level
2. The software may not be safety-critical although it forms part of a safety-critical system. For example, software may simply log transactions.

3. If a system must be of a high integrity level and if the software is shown to be of that integrity level, then the hardware must be at least of the same integrity level.

4. There is little point in producing 'perfect' code in some language if hardware and system software (in widest sense) are not reliable.

5. If a computer system is to run software of a high integrity level then that system should not at the same time accommodate software of a lower integrity level.

6. Systems with different requirements for safety levels must be separated.

7. Otherwise, the highest level of integrity required must be applied to all systems in the same environment.

CHAPTER 4

SYSTEM REQUIREMENTS

UML Diagrams

The UML (Unified Modeling Language) description provides a visual representation of the architecture for the Modular Dataset on Sensor Anomalies Supporting Machine Learning Research in Manufacturing Systems. The diagram illustrates the key components and their relationships within the system, including data acquisition, preprocessing, anomaly detection models, evaluation, visualization/reporting, and continuous learning.

4.1 Use Case Diagram

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

Use case: A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

Actor: An actor is a person, organization or external system that plays a role in one or more interaction with the system.

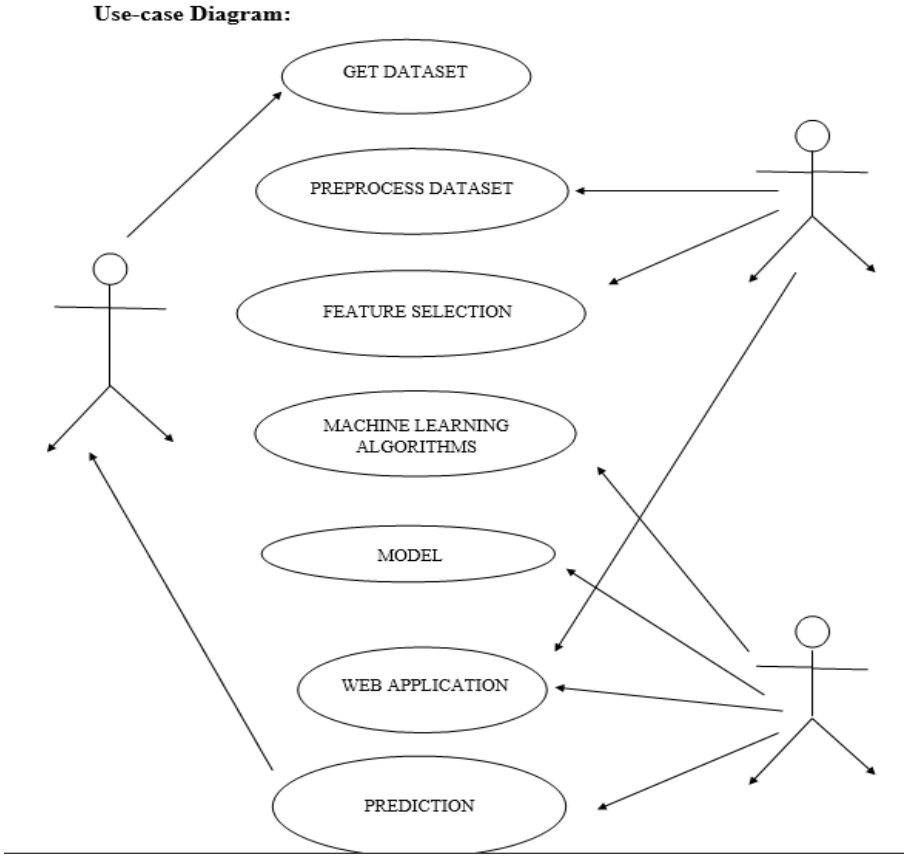

FIG 4.1 USECASE DIAGRAM

Sequence Diagram

A sequence diagram is a type of interaction diagram that illustrates how processes operate with one another and in what order. It displays the dynamic behaviour of a system, showcasing the flow of messages or interactions between objects or components over time.

- Illustrate interactions between objects or components over time.
- Show the order of messages exchanged between objects to accomplish specific tasks or scenarios.
- Include lifelines representing the lifespan of objects involved in the interactions.

Sequence Diagram:

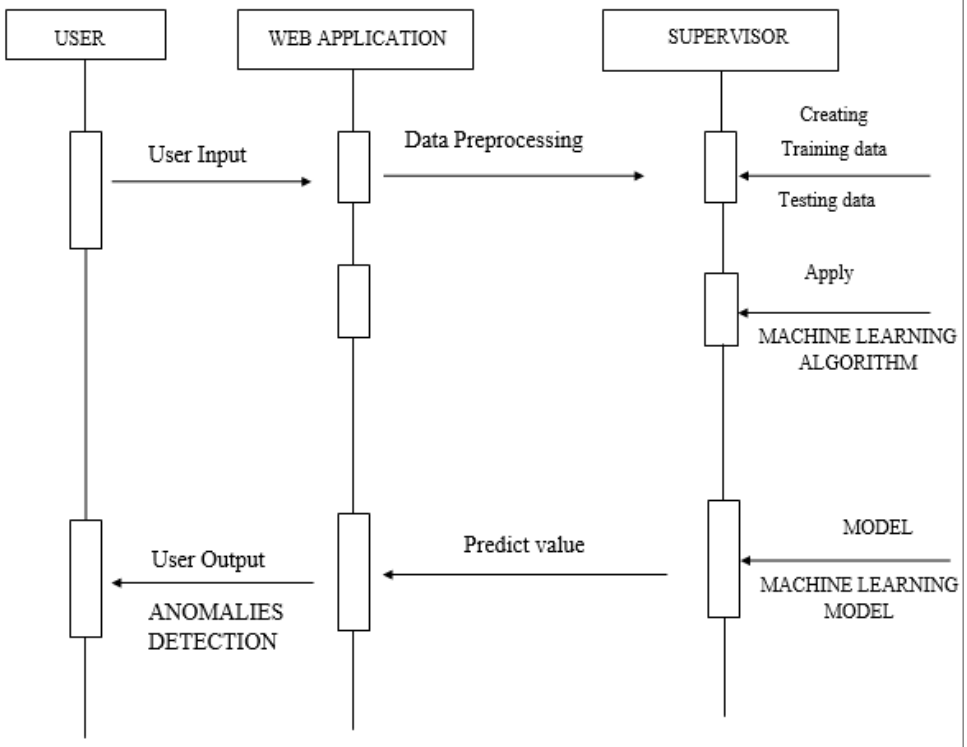

FIG 4.2 SEQUENCE DAIGRAM

5.1.3 Activity Diagram

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

The most important shape types:

- Rounded rectangles represent activities.

- Diamonds represent decisions.
- Bars represent the start or end of concurrent activities.
- A black circle represents the start of the workflow.
- An encircled circle represents the end of the workflow.

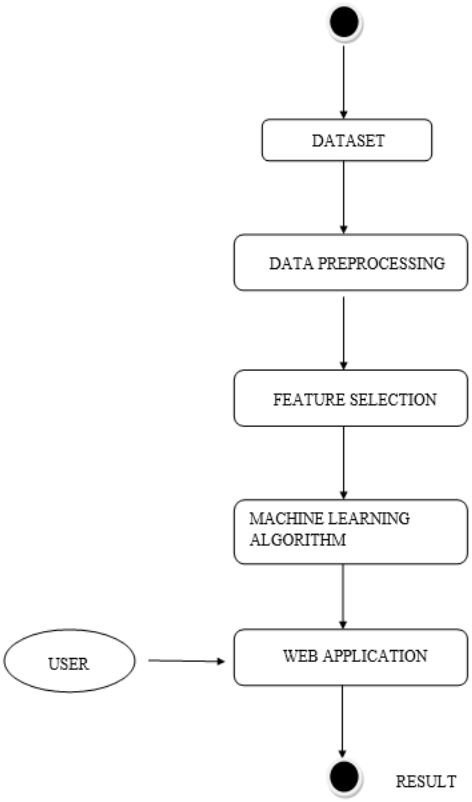

FIG 4.3 ACTIVITY DIAGRAM

5.1.4 Collaboration Diagram

UML Collaboration Diagrams illustrate the relationship and interaction between software objects. They require use cases, system operation contracts and

domain model to already exist. The collaboration diagram illustrates messages being sent between classes and objects.

Collobration Diagram:

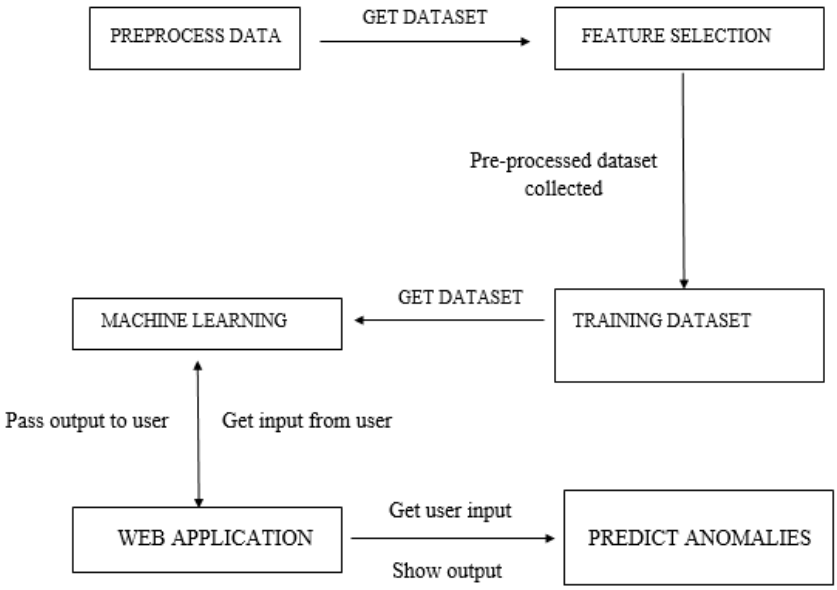

FIG 4.4 COLLABORATION DIAGRAM

4.2 HARDWARE AND SOFTWARE SPECIFICATION

HARDWARE REQUIREMENTS

- Hard Disk : 500GB and Above
- RAM : 4GB and Above

- Processor : I3 and Above

SOFTWARE REQUIREMENTS

- ✓ Operating System : Windows 10 (64 bit)
- ✓ Software : Python 3.7
- ✓ Tools : Anaconda (Jupyter Note Book IDE)

4.3 TECHNOLOGIES USED

- ➢ Python
- ➢ Machine Learning

Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- System scripting.

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.
- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.

- Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official End Of Life date of January 1, 2020 has been established for Python 2, after which time it will no longer be maintained.
- Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe Monty Python's Flying Circus, of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
- It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python is Interpreted

- Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer's processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.
- This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
- One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs.
- For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
- In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
- For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.
- Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.
- Python accomplishes what many programming languages don't: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

CHAPTER 5

CODING AND TESTING

5.1 CODING

Once the design aspect of the system is finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it easily screwed into the system.

5.2 CODING STANDARDS

Coding standards are guidelines to programming that focuses on the physical structure and appearance of the program. They make the code easier to read, understand and maintain. This phase of the system actually implements the blueprint developed during the design phase. The coding specification should be in such a way that any programmer must be able to understand the code and can bring about changes whenever felt necessary. Some of the standard needed to achieve the above-

Naming conventions of classes, data member, member functions, procedures etc., should be self-descriptive. One should even get the meaning and scope of the

variable by its name. The conventions are adopted for easy understanding of the intended message by the user. So it is customary to follow the conventions. These conventions are as follows:

VALUE CONVENTIONS

Value conventions ensure values for variable at any point of time. This involves the following:

- Proper default values for the variables.
- Proper validation of values in the field.
- Proper documentation of flag values.

5.3 TEST PROCEDURE

SYSTEM TESTING

Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example the design must not have any logic faults in the design is detected before coding commences, otherwise the cost of fixing the faults will be

considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough.

Testing is one of the important steps in the software development phase. Testing checks for the errors, as a whole of the project testing involves the following test cases:

- Static analysis is used to investigate the structural properties of the Source code.
- Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

UNIT TESTING

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

FUNCTIONAL TESTS

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files.

STRESS TEST

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

6.4.5 STRUCTURED TEST

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

- Exercise all logical decisions on their true or false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Checking attributes for their correctness.
- Handling end of file condition, I/O errors, buffer problems and textual errors in output information

6.4.6 INTEGRATION TESTING

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with

interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its intercommunications. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

TESTING

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is

important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers an yet undiscovered error. Any engineering product can be tested in one of the two ways:

WHITE BOX TESTING

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing.

Basis path testing:

- Flow graph notation
- Cyclometric complexity

BLACK BOX TESTING

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs

according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

The steps involved in black box test case design are:

- Graph based testing methods
- Equivalence partitioning
- Boundary value analysis

SOFTWARE TESTING STRATEGIES

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

- Testing begins at the module level and works "outward" toward the integration of the entire computer based system.
- Different testing techniques are appropriate at different points in time.
- The developer of the software and an independent test group conducts testing.

INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is

"putting them together"- interfacing. There may be the chances of data lost across on another's sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems.

PROGRAM TESTING

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses A relational operator or on arithmetic expression.

SECURITY TESTING

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for

invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

VALIDATION TESTING

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic.

USER ACCEPTANCE TESTING

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

CHAPTER 6

RESULT AND DISCUSSION

PREDICTION

The output analysis indicates that the Decision Tree algorithm exhibits higher accuracy compared to other models in the Modular Dataset on Sensor Anomalies Supporting Machine Learning Research in Manufacturing Systems. This finding suggests that Decision Trees are well-suited for detecting anomalies in sensor data, potentially due to their ability to capture complex decision boundaries and handle non-linear relationships between features.

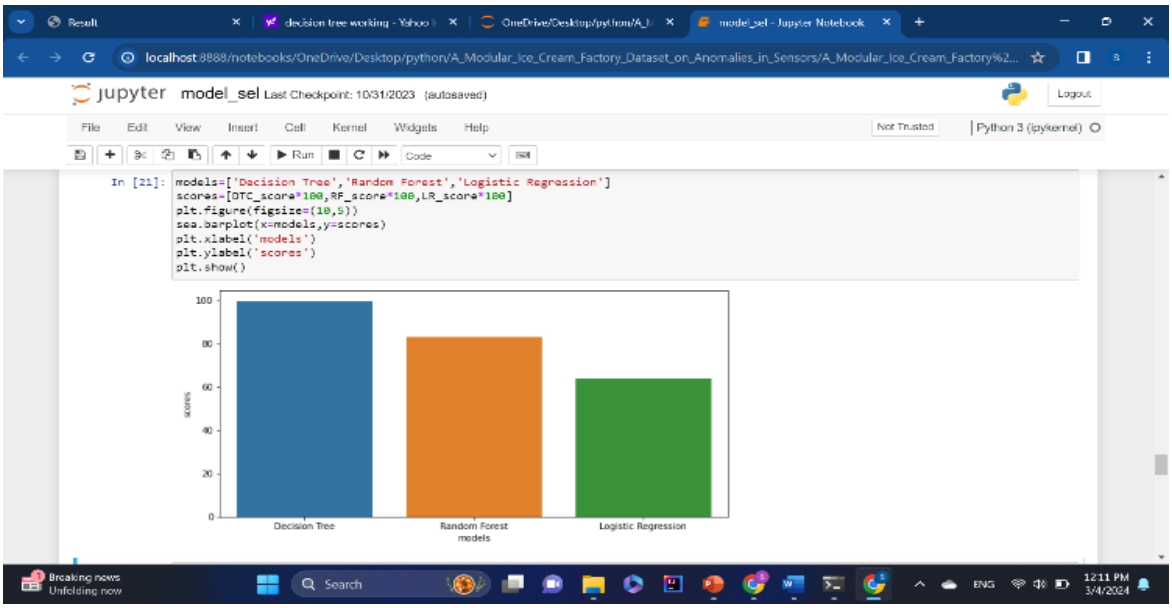

FIGURE 6.1: Decision tree gives more accuracy.

```
In [23]: s1=pd.DataFrame(data.loc[535:535,selected_Features])
         s1
```

Out[23]:

	number	Mixer_Level	Mixer_Temperature	Mixer_OpenOutlet	Mixer_Fill1On	Mixer
535	535	0.70799	277.37461	0	0	

1 rows × 30 columns

```
In [24]: DTC.predict(s1)
Out[24]: array([0], dtype=int64)
```

FIGURE 6.2: Output shows no anomaly found.

Fig(6.2) the output indicates that no anomalies were detected in the data by the anomaly detection model used in the Modular Dataset on Sensor Anomalies Supporting Machine Learning Research in Manufacturing Systems. This result could be interpreted in several ways: it may suggest that the system is functioning normally without any abnormal sensor readings during the observed period, or it could indicate potential limitations or inefficiencies in the anomaly detection algorithm employed. Further investigation and analysis are needed to understand the reasons behind the absence of detected anomalies and to refine the anomaly detection process for improved accuracy and reliability.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The development of the Modular Ice Cream Factory Dataset has provided a valuable resource for advancing research in machine learning applied to manufacturing systems. By simulating realistic sensor data from an ice cream production facility, researchers now have access to a diverse range of anomalies and normal operating conditions to train and evaluate their algorithms. The dataset's modular design allows for scalability and customization, accommodating various research needs and enabling the exploration of different machine learning techniques. Furthermore, the dataset facilitates the development of anomaly detection and predictive maintenance algorithms tailored specifically to the challenges of manufacturing environments. Through the analysis of sensor data, researchers can identify patterns indicative of equipment failures or process deviations, leading to more efficient maintenance scheduling and improved production efficiency. Ultimately, the Modular Ice Cream Factory Dataset has the potential to drive innovation in manufacturing systems by enabling the development of more intelligent and responsive production processes. One significant finding from this dataset is the effectiveness of machine learning techniques in identifying anomalies within manufacturing systems. By analyzing sensor data from

various points within the ice cream production process, researchers can train models to accurately detect deviations from normal operation. This capability is essential for preemptive maintenance, reducing downtime, and ensuring product quality.

Moreover, the Modular Ice Cream Factory Dataset highlights the importance of data standardization and preprocessing in machine learning research. Standardized data formats and preprocessing techniques ensure consistency and comparability across different studies, facilitating collaboration and benchmarking efforts within the research community.

7.2 FUTURE ENHANCEMENT

Looking ahead, several enhancements could be made to the Modular Ice Cream Factory Dataset to further its utility and impact in machine learning research in manufacturing systems. Firstly, expanding the dataset to include more diverse scenarios and anomalies would increase its applicability to a wider range of real-world manufacturing environments. This could involve incorporating data from different types of equipment, varying production volumes, and introducing additional sources of variability. Moreover, providing more detailed annotations and labels for the dataset would facilitate more comprehensive evaluations of machine learning algorithms.

Detailed information about the nature and severity of anomalies, as well as their root causes, would enable researchers to develop more targeted and effective anomaly detection techniques. Additionally, fostering collaboration

within the research community by establishing a platform for sharing insights, code, and benchmark results related to the dataset would accelerate progress in the field. This could involve organizing workshops or competitions centered around the dataset, encouraging the exchange of ideas and fostering innovation in machine learning applied to manufacturing systems. Overall, by addressing these future enhancements, the Modular Ice Cream Factory Dataset can continue to serve as a valuable resource for advancing research in this important area.

Overall, the Modular Ice Cream Factory Dataset serves as a foundation for advancing research in anomaly detection and machine learning applications within manufacturing systems. By addressing future enhancements and incorporating feedback from the research community, the dataset will continue to evolve as a valuable resource for academia and industry alike.

ANNEXURE

```
import pandas as pd
import numpy as np
d1=pd.read_csv("./Dataset/freeze/IceCreamFactory_105_Freeze.csv")
d1
d1=pd.read_csv("./Dataset/freeze/IceCreamFactory_105_Freeze.csv")
d1
d4=pd.read_csv("./Dataset/step/IceCreamFactory_102_Step.csv")
d4
d=[d1,d2,d3,d4]
for i in d:
    data=pd.concat([data,i],axis=0)
data
import matplotlib.pyplot as plt
import seaborn as sea
sea.countplot(x='Anomaly',data=data)
plt.show()
```

```
data['Anomaly'].unique()

data["Anomaly"]=data['Anomaly'].replace('-',0)

data["Anomaly"]=data['Anomaly'].replace('Freeze',1)

data["Anomaly"]=data['Anomaly'].replace('Ramp',2)

data["Anomaly"]=data['Anomaly'].replace('Step',3)

sea.countplot(x='Anomaly',data=data)

plt.show()

data

data=data.drop(['Timestamp','Parameter_for_Anomaly','Run_id','Actual_value'],axis=1)


data

data.isna().sum()

data.info()

data

data.to_csv('preprocesed.csv')

import pandas as pd

import warnings

warnings.filterwarnings('ignore')
```

```
import matplotlib.pyplot as plt

import seaborn as sea

data=pd.read_csv('./preprocesed.csv')

data

data=data.drop(['Unnamed: 0'],axis=1)

data

X=data.drop(['Anomaly'],axis=1)

Y=data['Anomaly']

X.info()

X=X.astype(float)

Y

pd.DataFrame(X).info()

from sklearn.linear_model import LogisticRegression

from sklearn.feature_selection import RFE

lr=LogisticRegression()

rfe=RFE(lr,n_features_to_select=30)

rfe.fit(X,Y)

rfe.get_support()

selected_Features=X.columns[rfe.get_support()]
```

```
x=X[selected_Features]

y=Y

selected_Features

x

from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.30,random_state=0)

X_train

X_test

from sklearn.metrics import classification_report

from sklearn.metrics import  accuracy_score

from sklearn.tree import DecisionTreeClassifier

DTC = DecisionTreeClassifier(random_state=0)

DTC.fit(X_train,Y_train)

DTC_score=DTC.score(X_test,Y_test)

y_pred=DTC.predict(X_test)

acc_score=accuracy_score(Y_test,y_pred)

print(acc_score)

print(classification_report(Y_test,y_pred))

DTC_score*100
```

```
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(max_depth=2,random_state=0)

RF.fit(X_train,Y_train)

RF_score=RF.score(X_test,Y_test)

RF_score*100

from sklearn.linear_model import LogisticRegression

LR=LogisticRegression(random_state=0)

LR.fit(X_train,Y_train)

LR_score=LR.score(X_test,Y_test)

LR_score*100

models=['Decision Tree','Random Forest','Logistic Regression']

scores=[DTC_score*100,RF_score*100,LR_score*100]

plt.figure(figsize=(10,5))

sea.barplot(x=models,y=scores)

plt.xlabel('models')

plt.ylabel('scores')

plt.show()

data[data['Anomaly']==0][530:540]

s1=pd.DataFrame(data.loc[535:535,selected_Features])
```

```
s1

pre=DTC.predict(s4)

pre

import joblib

joblib.dump(DTC,'model.pkl')
```

REFERENCES

1) Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).

2) Aggarwal, C. C. (2015). Outlier analysis (Vol. 12). Springer.

3) Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE transactions on pattern analysis and machine intelligence, 35(8), 1798-1828.

4) Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In Proceedings of the 23rd international conference on Machine learning (pp. 161-168).

5) Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM computing surveys (CSUR), 41(3), 1-58.

6) Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

7) Chollet, F. (2017). Deep learning with Python. Manning Publications Co.

8) Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 12(Jul), 2121-2159.

9) Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of statistics, 29(5), 1189-1232.

10) Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1). MIT press Cambridge.

11) Gunning, D. (2016). DARPA's explainable artificial intelligence (XAI) program. In Proceedings of the second conference on machine learning and systems (pp. 1-3).

12) Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

13) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

14) Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

15) Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

16) LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

17) Li, L., Han, W., & Wu, Q. J. (2018). Unsupervised anomaly detection for dynamic streaming data using LSTM-based predictive autoencoders. Neurocomputing, 324, 141-153.

18) Luo, Z., Feng, Y., Urtasun, R., & Gaidon, A. (2017). End-to-end deep learning for point cloud processing. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 4604-4612).

19) McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.

20) Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.

21) Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.

22) Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Berg, A. C. (2015). ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 115(3), 211-252.

23) Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

24) Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530.

25) Zhou, C., Yang, Z., Liu, Z., Niu, Y., Delgado-Battenfeld, C., & Zhao, T. (2018). Ensemble deep learning: A review. arXiv preprint arXiv:1611.03530.