**Novel Machine Learning Techniques for Classification of Rolling Bearings**

**ABSTRACT:**

Rolling bearing faults frequently cause rotating equipment failure, leading to costly downtime and maintenance expenses. As a result, researchers have focused on developing effective methods for diagnosing these faults. In this paper, we explore the potential of Machine Learning (ML) techniques for classifying the health status of bearings. Our approach involves decomposing the signal, extracting statistical features, and using feature selection employing Binary Grey Wolf Optimization. We propose an ensemble method using voting classifiers to diagnose faults based on the reduced set of features. To evaluate the performance of our methods, we utilize several performance indicators. Our results demonstrate that the proposed voting classifiers method achieves superior fault classification, highlighting its potential for use in predictive maintenance applications.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS** | i  v  vii |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. SCOPE OF THE PROJECT   3. OBJECTIVE   1.4 EXISTING SYSTEM  1.4.1 EXISTINGSYSTEM DISADVANTAGES  1.5 LITERATURE SURVEY  1.6 PROPOSED SYSTEM  1.6.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 :PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4 :SYSTEM DESIGN**  **4.1 general**  **4.2 uml diagrams**  4.2.1 USE CASE DIAGRAM  4.2.2 CLASS DIAGRAM  4.2.3 OBJECT DIAGRAM  4.2.4 STATE DIAGRAM  4.2.5 activity diagram  4.2.6 SEQUENCE DIAGRAM  4.2.7 COLLABORATION DIAGRAM  4.2.8 COMPONENT DIAGRAM  4.2.9 DATA FLOW DIAGRAM  4.2.10 DEPLOYMENT DIAGRAM  4.2.11 SYSTEM ARCHITECTURE |  |
| 5. | **CHAPTER 5 : DEVELOPMENT TOOLS**  5.1 general  5.2 History of Python  5.3 Importance of Python  5.4 Features of Python  5.5 Libraries used in python |  |
| 6. | **CHAPTER 6 :IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION |  |

|  |  |  |
| --- | --- | --- |
| 7. | **CHAPTER 7 :SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 :SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 :**  **FUTURE ENHANCEMENT**  9.1 FUTURE ENHANCEMENTS |  |
| **10** | **CHAPTER 10 :**  10.1CONCLUSION  10.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
|  |  |  |
| 4.1 | Use case Diagram |  |
| 4.2 | Class diagram |  |
| 4.3 | Object diagram |  |
| 4.4 | State Diagram |  |
| 4.5 | Activity Diagram |  |
| 4.6 | Sequence diagram |  |
| 4.7 | Collaboration diagram |  |
| 4.8 | Component Diagram |  |
| 4.9 | Data flow diagram |  |
| 4.10 | Deployment Diagram |  |
| 4.11 | Architecture Diagram |  |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processes. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**CHAPTER-1**

**INTRODUCTION**

1

Rolling bearings are critical components in rotating machinery and are widely used in various industrial applications such as automotive, aerospace, and manufacturing. These components play a crucial role in the smooth operation of machines by reducing friction between moving parts. However, the reliability and performance of rolling bearings can significantly affect the overall health of machinery. Faults in bearings, such as surface damage, misalignment, and wear, can lead to severe consequences, including machinery breakdowns, costly repairs, unexpected downtime, and potential safety hazards. Therefore, early and accurate detection of bearing faults is essential for maintaining system efficiency, minimizing maintenance costs, and preventing catastrophic failures.

In recent years, researchers have concentrated on developing advanced techniques to diagnose bearing faults effectively. Traditional diagnostic methods, such as vibration analysis and signal processing techniques, have been widely used to monitor bearing conditions. These methods involve analyzing the vibration signals generated by bearings to detect anomalies. However, traditional approaches often face challenges due to the complexity of vibration signals, the presence of noise, and the requirement for expert knowledge to interpret results. With the advent of Machine Learning (ML) techniques, new opportunities have emerged for automating the fault diagnosis process, improving accuracy, and reducing dependency on manual expertise. ML algorithms can analyze large datasets, learn patterns, and make predictions, making them suitable for fault diagnosis applications.

This paper explores the potential of ML techniques for classifying the health status of rolling bearings. The proposed approach begins with signal decomposition, where raw vibration signals are processed to extract useful information. Following this, a set of statistical features is derived from the decomposed signals, providing a comprehensive representation of the bearing’s condition. To enhance the diagnostic performance, feature selection is performed using Binary Grey Wolf Optimization, which helps in selecting the most relevant features, thereby reducing computational complexity and improving classification accuracy.

**1.2 SCOPE OF THE PROJECT**

The scope of this project is centered around developing an efficient and automated system for diagnosing faults in rolling bearings, which are critical components in rotating machinery. The primary aim is to leverage Machine Learning (ML) techniques to improve the reliability and accuracy of fault detection, thereby reducing downtime and maintenance costs. The project involves a comprehensive approach that starts with signal decomposition and statistical feature extraction, followed by feature selection using Binary Grey Wolf Optimization. By focusing on optimizing feature sets, we can enhance the performance of classifiers, ensuring the model is both efficient and scalable. The proposed solution incorporates an ensemble learning method using voting classifiers, specifically Random Forest, XGBoost, and Support Vector Classifier (SVC), to deliver superior diagnostic accuracy. This ensemble approach capitalizes on the strengths of each individual classifier, providing a robust and resilient fault detection framework. Additionally, the system's capabilities extend to predictive maintenance, offering early detection of bearing issues, which can significantly prolong equipment lifespan and prevent unexpected failures. This project is scalable and adaptable, making it applicable to various types of rotating equipment across different industries, ultimately contributing to smarter maintenance strategies and operational efficiency.

**1.3 OBJECTIVE**

The primary objective of this project is to develop a robust and efficient diagnostic system for detecting faults in rolling bearings, which are critical components in many industrial machines. The system aims to leverage advanced Machine Learning (ML) techniques to improve the accuracy and reliability of fault diagnosis, thus minimizing the risks of unexpected equipment failures. By decomposing vibration signals, extracting key statistical features, and optimizing feature selection using Binary Grey Wolf Optimization, the project focuses on enhancing the classification accuracy for identifying bearing health conditions. Additionally, it proposes an ensemble approach using voting classifiers—specifically Random Forest, XGBoost, and Support Vector Classifier (SVC)—to ensure a high-performing, generalized model capable of distinguishing between various fault types with precision. The overarching goal is to integrate this system into predictive maintenance frameworks, enabling industries to proactively manage equipment health, optimize maintenance schedules, and ultimately reduce operational costs while maximizing machinery uptime. Through comprehensive performance evaluations using standard indicators, the system aims to set a new benchmark in bearing fault diagnostics, surpassing traditional methods and contributing to the field of smart maintenance solutions.

**1.4 EXISTING SYSTEM:**

Traditional bearing fault diagnosis, researchers have relied on signal processing techniques to extract features from vibration signals. These techniques involve time-domain, frequency-domain, or time-frequency domain analyses. The extracted features are then used in machine learning classifiers like Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Decision Trees (DT), and Random Forest (RF) for fault classification.

Moreover, many existing systems use individual classifiers for fault diagnosis, which can lead to suboptimal results due to the limitations of single-model approaches. These methods often suffer from overfitting, poor generalization, or sensitivity to noisy data. The accuracy of these standalone classifiers is often insufficient for real-time predictive maintenance applications, especially when dealing with complex fault patterns in bearing systems.

**1.4.1 EXISTING SYSTEM DISADVANTAGES:**

* Limited Generalization
* Noise Sensitivity
* Overfitting and Underfitting
* Feature Dependence
* Model Complexity

**1.5 LITERATURE SURVEY**

**Title:** A Fault Diagnosis Method of Rolling Bearing Based on Improved Recurrence Plot and Convolutional Neural Network

**Author:** Xiaoping Liu, Lijian Xia, Jian Shi, Lijie Zhang, Linying Bai, Shao-Ping Wang

**Year:** 2023

**Description:** The recurrence plot (RP) method has been introduced into bearing fault diagnosis due to its capability of effectively analyzing nonlinear and nonstationary waveform signals in dynamic systems. However, the interference of noise increases the difficulty of RP-based fault diagnosis. To solve this problem, this article proposed a novel antinoise bearing fault diagnosis method based on improved RP and a convolutional neural network (CNN). First, different scales of approximation coefficients and detail coefficients were obtained and constructed for RP based on wavelet packet decomposition (WPD) on the vibrational signal. Meanwhile, redundant parts of each RP were removed according to its symmetry characteristics, and the remaining parts of these RPs were spliced into multiscale asymmetric RP (MARP) containing all coefficients. Then, a fault diagnosis model for rolling bearing was established with MARP as the input of the pretrained ResNet-34. Finally, the validity of the proposed fault diagnosis method was validated on the Paderborn bearing dataset. Experimental results showed that the proposed fault diagnosis method achieved an accuracy of 90% under Gaussian white noise with a signal-to-noise ratio (SNR) of above −6 dB.

**Title:** A Two-Stage Feature Selection Approach for Fruit Recognition Using Camera Images With Various Machine Learning Classifiers

**Author:** Tri Tran Minh Huynh; Tuan Minh Le; Long Ton That; Ly Van Tran; Son Vu Truong Dao

**Year:** 2022.

**Description**: Fruit and vegetable identification and classification system is always necessary and advantageous for the agriculture business, the food processing sector, as well as the convenience shops and hypermarkets where these products are sold. Therefore, it is necessary to build an effective automated tool to meet the needs of the market by boosting the outcome, in order to improve economic efficiency. In this paper, a two-stage model is proposed to recognize fruits using camera images. We employed a Densnet121 to get the features from the fruits dataset in the first module. In the second stage, we utilize a feature subset selection method to choose the most significant features for recognizing fruits from the images of the fruits. In this study, Adaptive particle - Grey Wolf Optimization (APGWO) has been applied for choosing the most pertinent features. The final subset feature has been used for recognizing fruits using several machine learning classifiers, namely K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and Multilayer Perceptron (MLP). The proposed research’s experimental results are highly effective; the training time of proposed models is reduced to over 50%, and the classification accuracy reaches 99%.

**Title:** Application of Machine Learning in Epileptic Seizure Detection

**Author:** Ly V Tran, Hieu M Tran, Tuan M Le, Tri T M Huynh, Hung T Tran, Son V T Dao

**Year:** 2022.

**Description:** Epileptic seizure is a neurological condition caused by short and unexpectedly occurring electrical disruptions in the brain. It is estimated that roughly 60 million individuals worldwide have had an epileptic seizure. Experiencing an epileptic seizure can have serious consequences for the patient. Automatic seizure detection on electroencephalogram (EEG) recordings is essential due to the irregular and unpredictable nature of seizures. By thoroughly analyzing EEG records, neurophysiologists can discover important information and patterns, and proper and timely treatments can be provided for the patients. This research presents a novel machine learning-based approach for detecting epileptic seizures in EEG signals. A public EEG dataset from the University of Bonn was used to validate the approach. Meaningful statistical features were extracted from the original data using discrete wavelet transform analysis, then the relevant features were selected using feature selection based on the binary particle swarm optimizer. This facilitated the reduction of 75% data dimensionality and 47% computational time, which eventually sped up the classification process. After having been selected, relevant features were used to train different machine learning models, then hyperparameter optimization was utilized to further enhance the models' performance. The results achieved up to 98.4% accuracy and showed that the proposed method was very effective and practical in detecting seizure presence in EEG signals. In clinical applications, this method could help relieve the suffering of epilepsy patients and alleviate the workload of neurologists.

**Title:**  Signal-to-Image: Rolling Bearing Fault Diagnosis Using ResNet Family Deep-Learning Models

**Author:**  Guoguo Wu ,Xuerong Ji ,Guolai Yang 1,Ye Jia and Chuanchuan Cao

**Year:** 2023

**Description**: Rolling element bearings (REBs) are the most frequent cause of machine breakdowns. Traditional methods for fault diagnosis in rolling bearings rely on feature extraction and signal processing techniques. However, these methods can be affected by the complexity of the underlying patterns and the need for expert knowledge during signal analysis. This paper proposes a novel signal-to-image method in which the raw signal data are transformed into 2D images using continuous wavelet transform (CWT). This transformation enhances the features extracted from the raw data, allowing for further analysis and interpretation. Transformed images of both normal and faulty rolling bearings from the Case Western Reserve University (CWRU) dataset were used with deep-learning models from the ResNet family. They can automatically learn and identify patterns in raw vibration signals after continuous wavelet transform is used, eliminating the need for manual feature extraction. To further improve the training results, squeeze-and-excitation networks (SENets) were added to improve the process. By comparing results obtained from several models, we found that SE-ResNet152 has the best performance for REB fault diagnosis.

**Title:** Triboelectric nanogenerator-embedded intelligent bearing with rolling ball defect diagnosis via signal decomposition and automated machine learning

**Author**:Fangyang Dong , Hengyi Yang, Hengxu Du, Meixian Zhu, Ziyue Xi, Yulian Wang, Taili Du , Minyi Xu

**Year:** 2024**.**

**Description:** Smart fault diagnosis of bearings is of great significance due to their extensive applications on various occasions. Recently, self-powered sensing technology based on triboelectric nanogenerators promotes the development of intelligent bearings. However, the effective detection and recognition of the rolling element defects of bearings need to be investigated further. This study proposes a triboelectric sensor-embedded rolling bearing (T-bearing) to monitor the working conditions and conduct the defect diagnosis of rolling balls. The interdigitated copper electrode covered by polytetrafluoroethylene film is attached to the inner surface of the outer ring of a commercial bearing. Such a design not only directly forms the TENG with rolling balls to obtain the contact-sensing signals, but also successfully achieves the diagnosis of rolling ball defects with similar triboelectric signals through a novel analysis and prediction paradigm combining signal decomposition and automated machine learning. Finally, a recognition accuracy of 99.48% with five different conditions of bearing balls is reached, which is extremely superior to the highest accuracy of 78.34% without signal decomposition. Thus, this study provides a new strategy for the defect diagnosis and the intelligent application of tribo electricbearings.

**1.6 PROPOSED SYSTEM**

To address the limitations of traditional bearing fault diagnosis methods, we propose an advanced ensemble-based approach utilizing a voting classifier. Our system begins with signal decomposition and statistical feature extraction, followed by feature selection using Binary Grey Wolf Optimization (BGWO). Unlike conventional methods, we employ an ensemble of three powerful classifiers: Random Forest, XGBoost, and Support Vector Classifier (SVC). These models are combined using a voting strategy, which significantly enhances classification accuracy and robustness.

The use of a voting ensemble leverages the strengths of each classifier—Random Forest's ability to handle noisy data, XGBoost's gradient boosting efficiency, and SVC's precision in class boundaries. By integrating these models, our approach achieves superior fault classification performance.The results demonstrate that our ensemble method outperforms standalone classifiers, providing a reliable and efficient solution for early detection of bearing faults in rotating machinery.

**1.6.1 PROPOSED SYSTEM ADVANTAGES:**

* Improved Accuracy
* Robustness to Noise
* Enhanced Generalization
* Better Feature Selection
* Reduced Overfitting

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Rolling bearings are critical components in rotating machinery and are widely used in various industrial applications such as automotive, aerospace, and manufacturing. These components play a crucial role in the smooth operation of machines by reducing friction between moving parts. However, the reliability and performance of rolling bearings can significantly affect the overall health of machinery. Faults in bearings, such as surface damage, misalignment, and wear, can lead to severe consequences, including machinery breakdowns, costly repairs, unexpected downtime, and potential safety hazards. Therefore, early and accurate detection of bearing faults is essential for maintaining system efficiency, minimizing maintenance costs, and preventing catastrophic failures.

In recent years, researchers have concentrated on developing advanced techniques to diagnose bearing faults effectively. Traditional diagnostic methods, such as vibration analysis and signal processing techniques, have been widely used to monitor bearing conditions. These methods involve analyzing the vibration signals generated by bearings to detect anomalies. However, traditional approaches often face challenges due to the complexity of vibration signals, the presence of noise, and the requirement for expert knowledge to interpret results. With the advent of Machine Learning (ML) techniques, new opportunities have emerged for automating the fault diagnosis process, improving accuracy, and reducing dependency on manual expertise.

This paper explores the potential of ML techniques for classifying the health status of rolling bearings. The proposed approach begins with signal decomposition, where raw vibration signals are processed to extract useful information. Following this, a set of statistical features is derived from the decomposed signals, providing a comprehensive representation of the bearing’s condition. To enhance the diagnostic performance, feature selection is performed using Binary Grey Wolf Optimization, which helps in selecting the most relevant features, thereby reducing computational complexity and improving classification accuracy.

**2.2 METHODOLOGIES**

**2.2.1MODULES NAME:**

**Modules Name:**

* Acquiring the Dataset
* Data Inspection
* Data Preparation
* Model Deployment
* Model Development
* Model Testing
* Outcome Prediction
  + 1. **MODULES EXPLANATION:**

1. **Acquiring The Dataset:**

The first step in our approach involves acquiring a high-quality dataset that includes vibration signals from rolling bearings under various operational conditions. These datasets are typically collected from real-world industrial setups or specialized test rigs designed to simulate bearing faults. The dataset serves as the foundation for our model, ensuring that it can accurately differentiate between normal and faulty bearing states. We focus on gathering data that captures a range of fault types, including inner race, outer race, and ball defects, as well as different levels of fault severity.

1. **Data Inspection:**

Once the dataset is acquired, a thorough inspection is conducted to understand its structure, quality, and relevance. This step involves checking for missing values, inconsistencies, or anomalies that could affect model performance. We perform exploratory data analysis (EDA) to identify patterns, trends, and correlations among features, which can provide insights into the bearing conditions. By visualizing the data, we can assess its distribution and identify the most influential features, setting the stage for effective feature engineering.

**3) Data Preparation:**

In the data preparation phase, we clean and preprocess the data to ensure it is suitable for machine learning algorithms. This includes handling missing values, scaling features to a uniform range, and encoding categorical variables if present. Signal decomposition techniques, such as Wavelet Transform or Empirical Mode Decomposition (EMD), are applied to extract time-frequency domain features from the raw vibration signals. Feature extraction is critical for capturing the characteristics of bearing faults, enabling the model to make more accurate predictions.

1. **Model Deployment:**

Once the machine learning model achieves satisfactory performance in the testing phase, the next step is model deployment. This process involves integrating the trained ensemble model (voting classifier using Random Forest, XGBoost, and SVC) into a real-time system where it can be used for predicting the health status of bearings on live data. The deployment environment may include an industrial setup where vibration signals are continuously monitored and processed. The deployed model is designed to handle incoming data streams, perform feature extraction, and apply the trained classifiers to detect faults accurately. This setup enables real-time fault diagnosis, which is crucial for implementing effective predictive maintenance strategies.

1. **Model Development:**

For model development, we implement an ensemble learning approach using voting classifiers, combining Random Forest, XGBoost, and Support Vector Classifier (SVC). The goal is to leverage the strengths of these algorithms: Random Forest for its robustness to overfitting, XGBoost for its superior handling of complex patterns, and SVC for its ability to perform well with high-dimensional data. Feature selection is optimized using Binary Grey Wolf Optimization, which reduces dimensionality and enhances model performance. The ensemble model aims to deliver high accuracy in classifying bearing health conditions, providing a reliable diagnostic tool for predictive maintenance.

1. **Model Testing:**

After developing the model, it undergoes rigorous testing to evaluate its effectiveness. We split the dataset into training and testing sets, using techniques like cross-validation to ensure the model’s robustness. Various performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix analysis are used to assess how well the model classifies different fault types. This step ensures that the ensemble voting classifiers can generalize well to new, unseen data, which is crucial for real-world applications.

1. **Outcome Prediction:**

The final step in our process is outcome prediction, where the trained model is deployed to predict the health status of rolling bearings in real-time scenarios. The system processes incoming vibration signals, extracts the relevant features, and uses the ensemble voting classifier to determine whether the bearing is healthy or faulty. This predictive capability supports proactive maintenance strategies, enabling industries to detect potential bearing failures early, thereby minimizing downtime and reducing maintenance costs. The outcome prediction module is designed to be integrated seamlessly into existing maintenance systems, providing actionable insights for machinery health management.

**2.3 TECHNIQUE USED OR ALGORITHM USED**

**2.3.1** **EXISTING TECHNIQUE:**

Traditional bearing fault diagnosis methods typically rely on individual Machine Learning (ML) classifiers to detect anomalies. Commonly used algorithms include Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Tree (DT), and Random Forest (RF). These models analyze statistical features extracted from vibration signals to classify bearing health conditions. However, the performance of these standalone classifiers can be limited by their sensitivity to noise, feature selection, and especially in complex fault scenarios.

Additionally, existing approaches may lack robustness due to their reliance on a single classifier's decision boundary, which could result in lower accuracy when dealing with diverse datasets. While techniques like feature extraction and optimization methods can enhance classifier performance, they still might not achieve optimal results due to overfitting or underfitting issues. Therefore, there is a need for more advanced methods that can integrate multiple classifiers' strengths to provide more accurate and reliable fault diagnoses.

**2.3.2 PROPOSED TECHNIQUE USED OR ALGORITHM USED:**

To overcome the limitations of individual classifiers, we propose a novel ensemble-based approach using a voting classifier mechanism. This method combines the capabilities of three powerful algorithms: Random Forest, XGBoost, and Support Vector Classifier (SVC). By leveraging the complementary strengths of these classifiers, our voting ensemble provides a robust solution for bearing fault diagnosis. The ensemble model integrates Random Forest's efficiency in handling large datasets, XGBoost's gradient boosting technique for reducing bias and variance, and SVC's precision in drawing decision boundaries.

The proposed voting classifier algorithm begins with signal preprocessing and feature extraction, followed by optimized feature selection using Binary Grey Wolf Optimization (BGWO). The selected features are then classified using the ensemble model, where each classifier contributes to the final decision based on a majority vote. This hybrid approach enhances the diagnostic accuracy and stability of the system, achieving superior fault detection results compared to traditional methods, making it suitable for predictive maintenance applications in industrial settings.

.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

We can see from the results that on each database, the error rates are very low due to the discriminatory power of features and the regression capabilities of classifiers. Comparing the highest accuracies (corresponding to the lowest error rates) to those of previous works, our results are very competitive.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 500 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* Operating System : Windows 10
* Platform : Spyder3
* Programming Language : Python
* Front End : Spyder3

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**The major non-functional Requirements of the system are as follows**

**Usability**

The system is designed with completely automated process hence there is no or less user intervention.

**Reliability**

The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

**Performance**

This system is developing in the high level languages and using the advanced back-end technologies it will give response to the end user on client system with in very less time.

**Supportability**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is built into the system.

**Implementation**

The system is implemented in web environment using Jupyter notebook software. The server is used as the intellignce server and windows 10 professional is used as the platform. Interface the user interface is based on Jupyter notebook provides server system.

**CHAPTER 4**

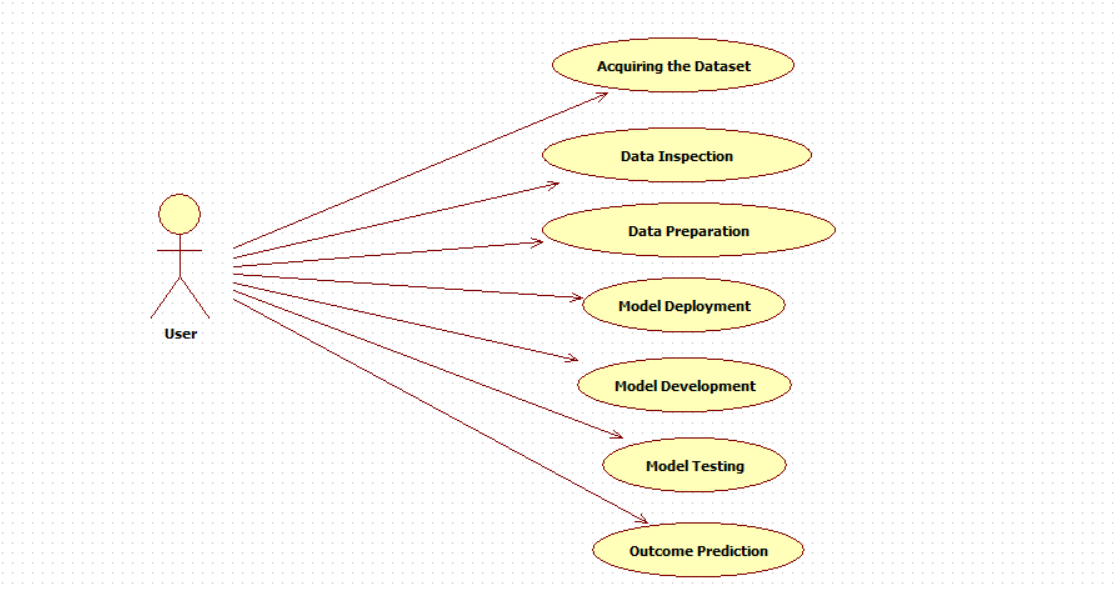
**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering.

**4.2 UML DIAGRAMS**

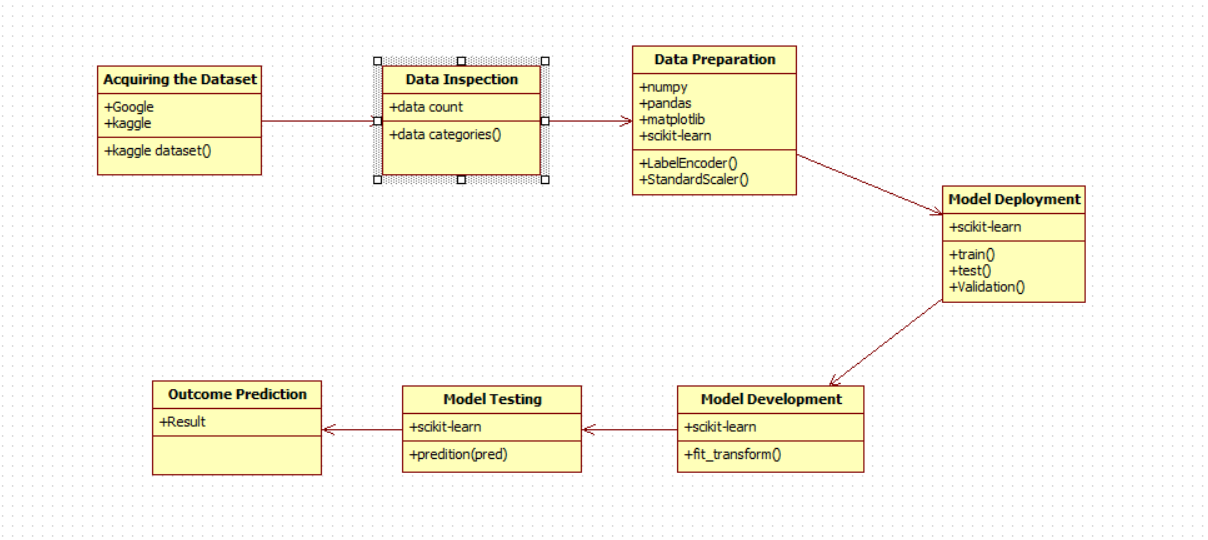
**4.2.1 USE CASE DIAGRAM**



**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

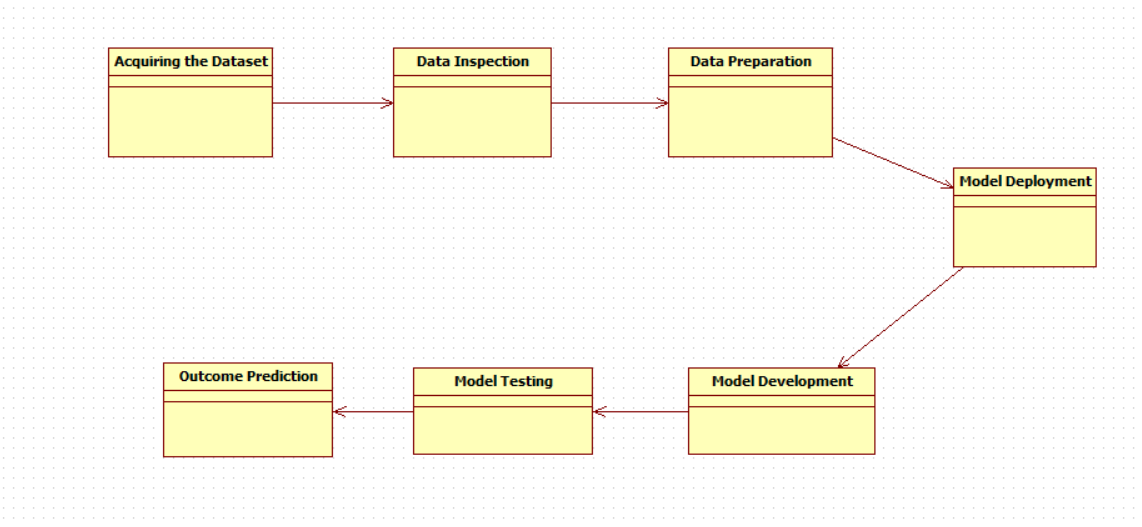
**4.2.2 CLASS DIAGRAM**

****

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

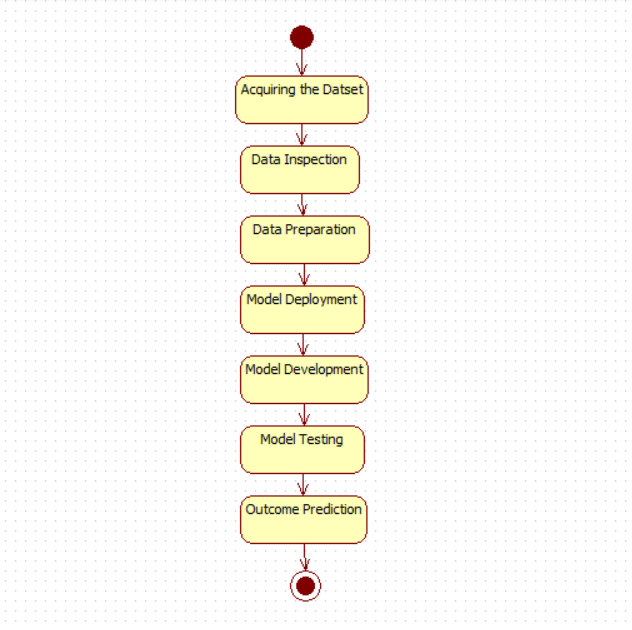
**4.2.3 OBJECT DIAGRAM**



**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

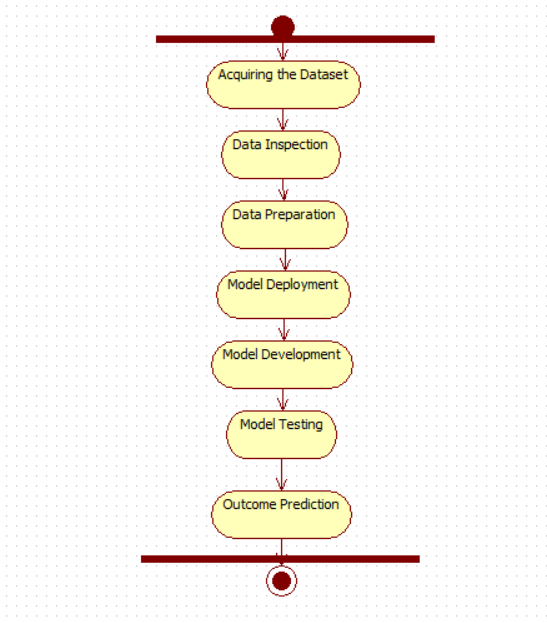
**4.2.4 STATE DIAGRAM**

****

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

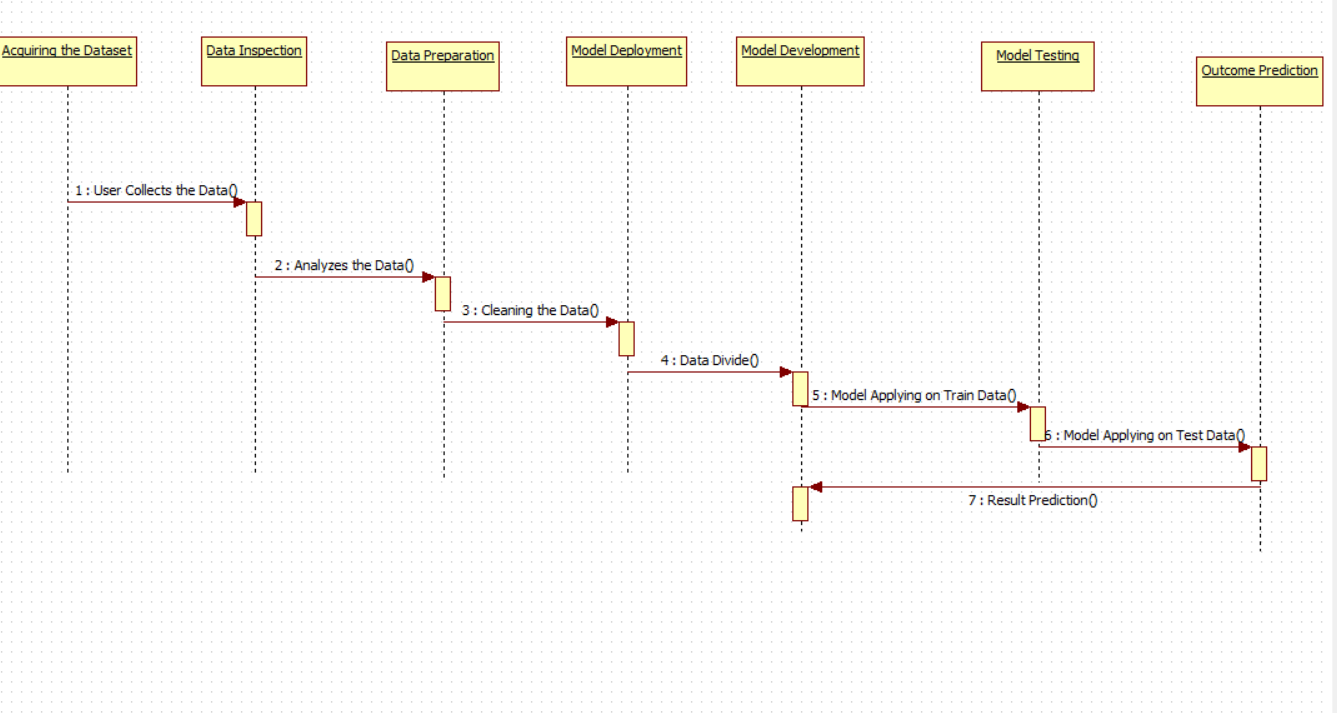
**4.2.5 ACTIVITY DIAGRAM**

****

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

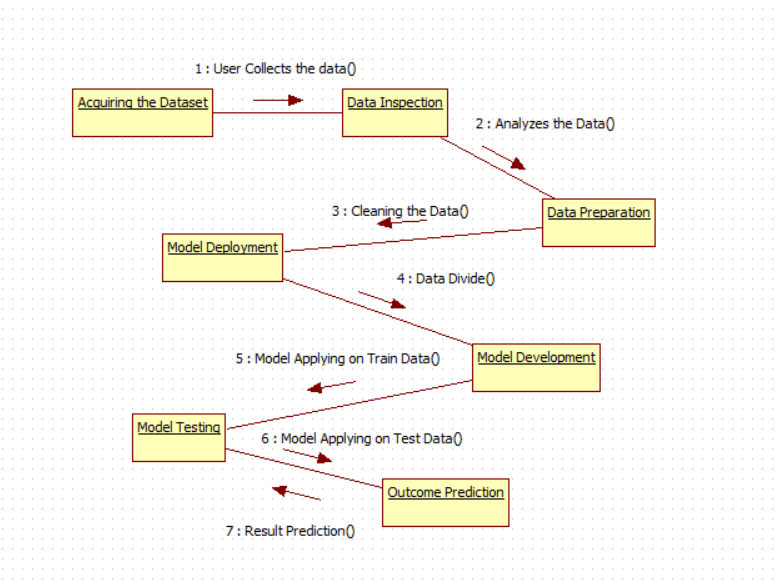
**4.2.6 SEQUENCE DIAGRAM**

****

**EXPLANATION:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

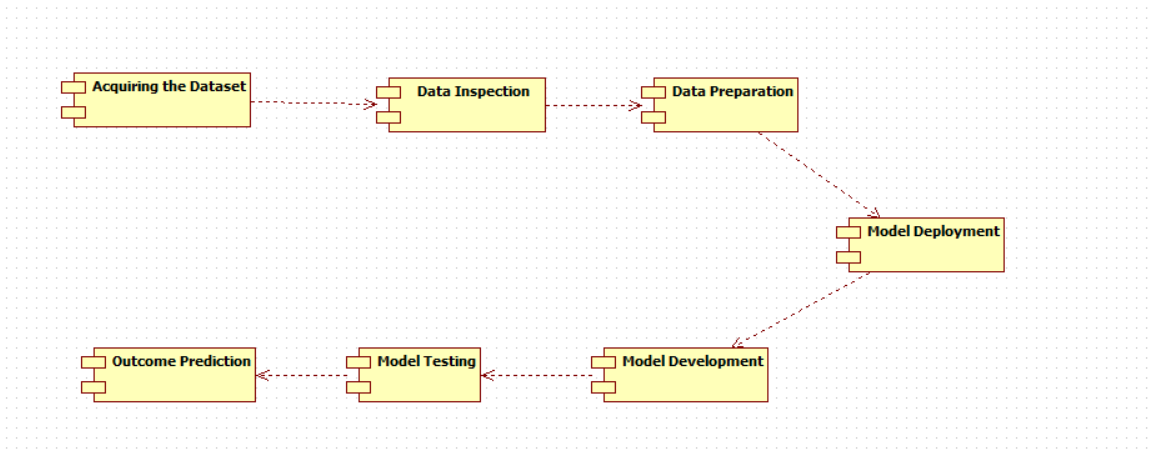
**4.2.7 COLLABORATION DIAGRAM**



**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

**4.2.8 COMPONENT DIAGRAM**



**EXPLANATION**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

**4.2.9 DATA FLOW DIAGRAM**

**Level 0**

Data Preparation

User

Acquiring the Dataset

Data Inspection

**Level 1**

Outcome Prediction

Model Deployment

Model Development

Model Testing

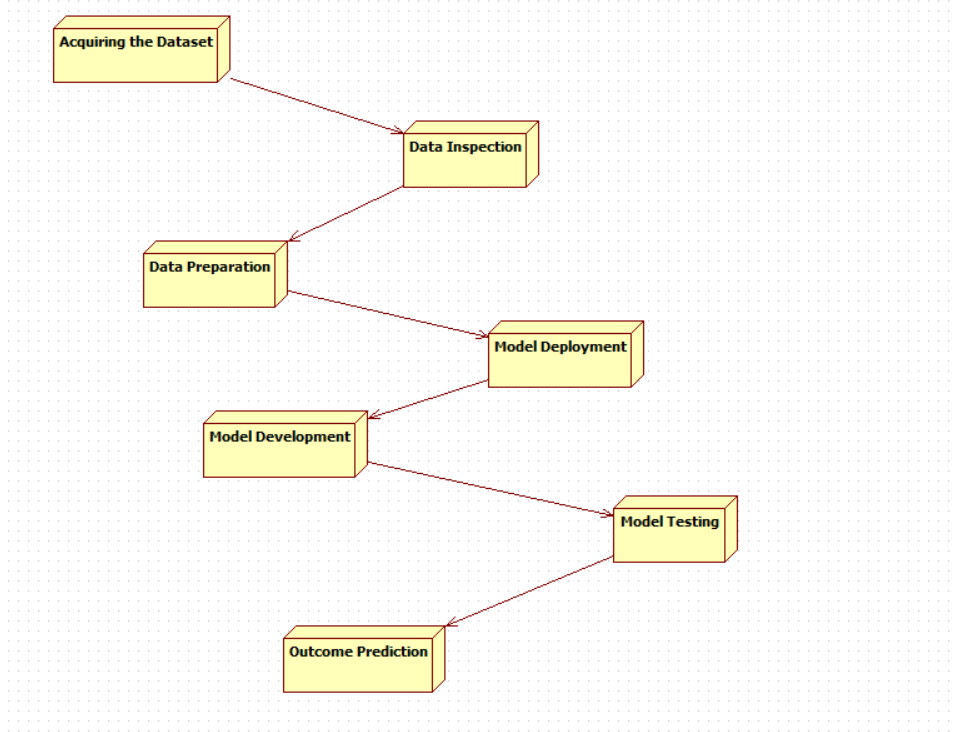
Fig 4.9: Data Flow Diagrams

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**4.2.10 DEPLOYMENT DIAGRAM**

****

**EXPLANATION:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

**-**

**SYSTEM ARCHITECTURE:**

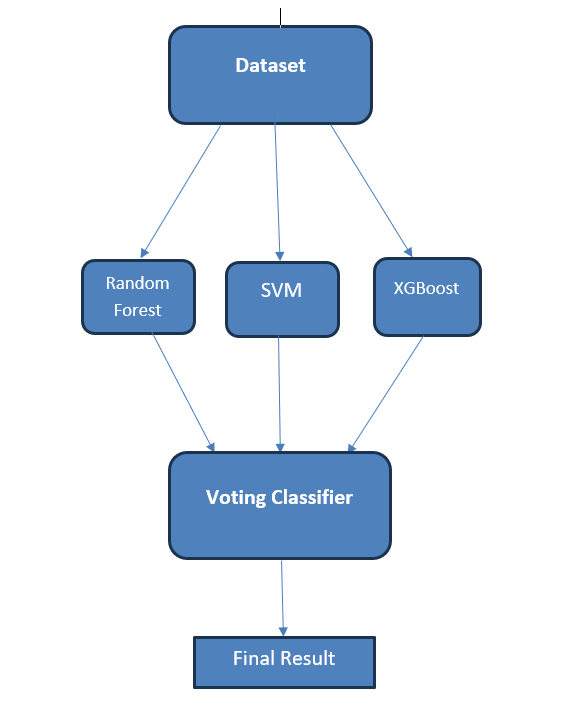


Fig 4.11: System Architecture

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## 5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### 5.3 Importance of Python

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### 5.4 Features of Python

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.5 Libraries used in python**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as dataframes.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure : NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding:**

**CHAPTER 7**

**SNAPSHOTS**

**General:**

This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

**SNAPSHOTS**

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3Types of Tests**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**FUTURE ENHANCEMENT**

**9.1 FUTURE ENHANCEMENTS:**

Future enhancements for the proposed system aim to expand its capabilities and improve fault detection and prediction in rolling bearings. One potential enhancement is the integration of deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to automatically extract features from raw sensor data, eliminating the need for manual feature extraction. This could result in more accurate and efficient fault classification.

Furthermore, implementing real-time monitoring and Internet of Things (IoT) integration could allow for continuous data collection and immediate diagnosis, enabling a more proactive approach to maintenance. By employing a feedback loop for periodic model retraining, the system could stay updated with new data, adapting to changing operational conditions.

Additionally, incorporating advanced visualization tools for easy interpretation of results would provide end-users with actionable insights and improve decision-making. The system's scope can also be extended to detect a wider range of faults beyond rolling bearings, making it applicable to other critical machinery. Combining these advancements will enhance the system’s utility in industrial predictive maintenance, ultimately reducing downtime and maintenance costs.

**CHAPTER 10**

**CONCLUSIONAND REFERENCES**

**10.1** **CONCLUSION**

In conclusion, this paper presents an effective machine learning-based approach for diagnosing rolling bearing faults. By leveraging Binary Grey Wolf Optimization for feature selection and employing an ensemble method using voting classifiers, we achieved significant improvements in fault classification accuracy. The proposed system demonstrates its potential for real-time predictive maintenance applications, reducing downtime and maintenance costs. Future enhancements, such as integrating deep learning models and real-time data processing, promise to further elevate the system’s performance and applicability in industrial settings.

**10.2 REFERENCES**

[1] M. Syafrudin, G. Alfian, N. L. Fitriyani, and J. Rhee, “Performance Analysis of IoT-BasedSensor, Big Data Processing, and Machine Learning Model for Real-Time Monitoring System in Automotive Manufacturing,” Sensors, vol. 18, no. 9, Art. no. 9, Sep. 2018, doi: 10.3390/s18092946.

[2] F. Arellano-Espitia, M. Delgado-Prieto, V. Martinez-Viol, J. J. Saucedo-Dorantes, and R. A.

Osornio-Rios, “Deep-Learning-Based Methodology for Fault Diagnosis in Electromechanical Systems,” Sensors, vol. 20, no. 14, Art. no. 14, Jan. 2020, doi: 10.3390/s20143949.

[3]R. Marino, C. Wisultschew, A. Otero, J. M. Lanza Gutierrez, J. Portilla, and E. de la Torre, “A

Machine-Learning-Based Distributed System for Fault Diagnosis With Scalable Detection Quality in Industrial IoT,” IEEE Internet Things J., vol. 8, no. 6, pp. 4339–4352, Mar. 10.1109/JIOT.2020.3026211. 2021,

[4] X. Liu, L. Xia, J. Shi, L. Zhang, L. Bai, and S. Wang, “A Fault Diagnosis Method of Rolling Bearing Based on Improved Recurrence Plot and Convolutional Neural Network,” IEEE Sens. J., vol. 23, no. 10, pp. 10767–10775, May 2023, doi: 10.1109/JSEN.2023.3265409.

[5] H. Tao, J. Qiu, Y. Chen, V. Stojanovic, and L. Cheng, “Unsupervised cross-domain rolling bearing fault diagnosis based on time-frequency information fusion,” J. Frankl. Inst., vol. 360, no. 2, pp. 14541477, Jan. 2023, doi10.1016/j.jfranklin.2022.11.004

[6] A. Alkaya and İ. Eker, “Variance sensitive adaptive 2, threshold-based PCA method for fault detection with experimental application,” ISA Trans., vol. 50, no. pp. 287–302, Apr. 2011, doi: 10.1016/j.isatra.2010.12.004.

[7] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, “KNN Model-Based Approach in Classification,” in On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, R. Meersman, Z. Tari, and D. C. Schmidt, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2003, pp. 986–996. doi: 10.1007/978-3-540-399643\_62.

[8] G. Wu, X. Ji, G. Yang, Y. Jia, and C. Cao, “Signal to-Image: Rolling Bearing Fault Diagnosis Using ResNet Family Deep-Learning Models,” Processes, vol. 11, no. 5, Art. no. 5, May 2023, doi: 10.3390/pr11051527.

[9] F. Dong et al., “Triboelectric nanogenerator embedded intelligent bearing with rolling ball defect diagnosis via signal decomposition and automated machine learning,” Nano Energy, vol. 119, p. 109072, Jan. 10.1016/j.nanoen.2023.109072. 2024,

[10] S. Afrasiabi, M. Afrasiabi, B. Parang, and M. Mohammadi, “Real-Time Bearing Fault Diagnosis of Induction Motors with Accelerated Deep Learning Approach,” in 2019 10th International Power Electronics, Drive Systems and Technologies Conference (PEDSTC), Feb. 2019, pp. 155–159. doi: 10.1109/PEDSTC.2019.8697244.

[11] X. Li, W. Zhang, and Q. Ding, “Understanding and improving deep learning-based rolling bearing fault diagnosis with attention mechanism,” Signal Process., vol. 161, pp. 136–154, Aug. 2019, doi: 10.1016/j.sigpro.2019.03.019.

[12] G. Xu, M. Liu, Z. Jiang, D. Söffker, and W. Shen, “Bearing Fault Diagnosis Method Based on Deep Convolutional Neural Network and Random Forest Ensemble Learning,” Sensors, vol. 19, no. 5, Art. no. 5, Jan. 2019, doi: 10.3390/s19051088.

[13] L. Yuan, D. Lian, X. Kang, Y. Chen, and K. Zhai, “Rolling Bearing Fault Diagnosis Based on Convolutional Neural Network and Support Vector Machine,” IEEE Access, vol. 8, pp. 137395–137406, 2020, doi: 10.1109/ACCESS.2020.3012053.

[14] X. Li, Y. Yang, H. Pan, J. Cheng, and J. Cheng, “A novel deep stacking least squares support vector machine for rolling bearing fault diagnosis,” Comput. Ind., vol. 110, pp. 36–47, Sep. 2019, doi: 10.1016/j.compind.2019.05.005.

[15] A. Ucar, M. Karakose, and N. Kırımça, “Artificial Intelligence for Predictive Maintenance

Applications: Key Components, Trustworthiness, and Future Trends,” Appl. Sci., vol. 14, no. 2, Art. no. 2, Jan. 2024, doi: 10.3390/app14020898.

[16] S. Kaparthi and D. Bumblauskas, “Designing predictive maintenance systems using decision tree based machine learning techniques,” Int. J. Qual. Reliab. Manag., vol. 37, no. 4, pp. 659–686, Jan. 2020, doi: 10.1108/IJQRM-04-2019-0131.

[17] B. Qi, Y. Li, W. Yao, and Z. Li, “Application of EMD Combined with Deep Learning and

Knowledge Graph in Bearing Fault,” J. Signal Process. Syst., vol. 95, no. 8, pp. 935–954, Aug.

2023, doi: 10.1007/s11265-023-01845-z.

[18] F. Dao, Y. Zeng, and J. Qian, “A novel denoising method of the hydro-turbine runner for fault signal based on WT-EEMD,” Measurement, vol. 219, p. 113306, Sep. 2023, doi: 10.1016/j.measurement.2023.113306.

[19] Y. Hu, Y. Ouyang, Z. Wang, H. Yu, and L. Liu, “Vibration signal denoising method based on CEEMDAN and its application in brake disc unbalance detection,” Mech. Syst. Signal Process.,

vol. 187, p. 109972, Mar. 2023, doi: 10.1016/j.ymssp.2022.109972.

[20] R. Abdelkader, A. Kaddour, A. Bendiabdellah, and Z. Derouiche, “Rolling Bearing Fault Diagnosis Based on an Improved Denoising Method Using the Complete Ensemble Empirical Mode Decomposition and the Optimized Thresholding Operation,” IEEE Sens. J., vol. 18, no. 17, pp. 71667172, Sep. 2018, doi: 10.1109/JSEN.2018.2853136.

[21] Z. Wu and N. E. Huang, “Ensemble empirical mode decomposition: a noise-assisted data analysis method,” Adv. Adapt. Data Anal., vol. 01, no. 01, pp. 1–41, Jan. 2009, doi: 10.1142/S1793536909000047.

[22] Y. Li, Y. Li, X. Chen, J. Yu, H. Yang, and L. Wang, “A New Underwater Acoustic Signal Denoising Technique Based on CEEMDAN, Mutual Information, Permutation Entropy, and Wavelet Threshold Denoising,” Entropy, vol. 20, no. 8, Art. no. 8, Aug. 2018, doi: 10.3390/e20080563.

[23] Y. Yang, P. Fu, and Y. He, “Bearing Fault Automatic Classification Based on Deep Learning,” IEEE Access, vol. 6, pp. 71540–71554, 2018, doi: 10.1109/ACCESS.2018.2880990.

[24] J. R. Quinlan, “Induction of decision trees,” Mach. Learn., vol. 1, no. 1, pp. 81–106, Mar. 1986, doi: 10.1007/BF00116251.

[25] L. Breiman, “Random Forests,” Mach. Learn., vol. 45, no. 1, pp. 5–32, Oct. 2001, doi:

10.1023/A:1010933404324.

[26] T. T. M. Huynh, T. M. Le, L. T. That, L. V. Tran, and S. V. T. Dao, “A Two-Stage Feature Selection Approach for Fruit Recognition Using Camera Images With Various Machine Learning

Classifiers,” IEEE Access, vol. 10, pp. 132260132270, 2022, doi: 10.1109/ACCESS.2022.3227712.

[27] L. V. Tran, H. M. Tran, T. M. Le, T. T. M. Huynh, H. T. Tran, and S. V. T. Dao, “Application of Machine Learning in Epileptic Seizure Detection,” Diagnostics, vol. 12, no. 11, Art. no. 11, Nov. 2022, doi: 10.3390/diagnostics12112879.

[28] S. V. T. Dao, Z. Yu, L. V. Tran, P. N. K. Phan, T. T. M. Huynh, and T. M. Le, “An Analysis of Vocal Features for Parkinson’s Disease Classification Using Evolutionary Algorithms,” Diagnostics, vol. 12, no. 8, Art. no. 8, Aug. 2022, doi: 10.3390/diagnostics12081980.

[29] T. Le, T. Pham, and S. Dao, “Using Machine Learning to Predict the Defaults of Credit Card

Clients,” in Fintech with Artificial Intelligence, Big Data, and Blockchain, P. M. S. Choi and S. H. Huang, Eds., in Blockchain Technologies. , Singapore: Springer, 2021, pp. 133–152. doi:

10.1007/978-981-33-6137-9\_4.

[30] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” Adv. Eng. Softw., vol. 69, pp. 4661, Mar. 2014, doi:10.1016/j.advengsoft.2013.12.007.