UG Project
Report

Web development on mine machinery predictive maintenance

Report submitted in fulfillment of the requirements for the UG Project of

Third Year B.Tech.
in
Mining Engineering

by

Kuthadi Jaswanth Sai Prem

Under the guidance of

Dr. Suprakash Gupta

Department of Mining Engineering

INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India

Semester VI - May 2024

Dedicated to

Our beloved Parents, Teachers

and IIT-BHU.

Declaration

I, **Kuthadi Jaswnath Sai Prem**, certify that the work embodied in this UG Project Report is my bonafide work and carried out by me under the supervision of **Dr. Suprakash Gupta** in the Department of Mining Engineering, Indian Institute of Technology (BHU), Varanasi. The matter embodied in this report has not been submitted for the award of any other degree.

I declare that I have faithfully acknowledged and given credits to the research workers wherever their works have been cited in my work in this report. I further declare that I have not willfully copied any other's work, paragraphs, text, data, results, etc., reported in journals, books, magazines, reports dissertations, theses, etc., or available on websites, and have not included them in this report as my work.

Place: IIT (BHU) Varanasi
Date: May 1, 2024

Kuthadi Jaswanth Sai Prem
B.Tech. IIIrd Year Students
Department of Mining Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Department of Computer Science and Engineering
Indian Institute of Technology (BHU) Varanasi
Varanasi, INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Web development on mine machinery predictive maintenance**" being submitted by **Kuthadi Jaswanth Sai Prem (21155058)** , carried out in the Department of M Engineering, Indian Institute of Technology (BHU) Varanasi, is a bonafide work of our supervision.*

Dr. Suprakash Gupta
Supervisor
Department of Mining Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Dr. Suprakash Gupta
Head of the Department
Department of Mining Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to extend our heartfelt respect and gratitude to our supervisor, Dr. Suprakash Gupta, Professor in the Department of Mining Engineering at the Indian Institute of Technology (BHU) Varanasi. Dr. Suprakash Gupta's invaluable guidance, expertise, and enthusiastic involvement have been instrumental in the planning and development of this project. We are deeply indebted to his wisdom, vision, and persistent encouragement throughout our journey.

We also gratefully acknowledge his painstaking efforts in thoroughly going through and improving the manuscripts without which this work could not have been completed. We are also highly obliged to Prof. Pramod Kumar Jain, Director, Indian Institute of Technology (BHU) Varanasi, and Prof. Dr. Suprakash Gupta, Head of Department, Mining Engineering for providing all the facilities, help and encouragement for carrying out this exploratory project work. We also thank Dr. Rajesh Rai, Associate Prof., Mining Engineering, as a convener of this course and the other faculty members for their timely help and cooperation extended throughout the course of investigation. We are also obliged to our parents for their moral support, love, encouragement and blessings to complete this task. Finally, we are indebted and grateful to the Almighty for helping us in this endeavor.

Place: IIT (BHU) Varanasi **Kuthadi Jaswanth Sai Prem**

Date: May 1, 2024

Abstract

Weibull models, which are essential for assessing survivability and dependability, characterize the many kinds of failures observed in phenomena and components. There are numerous more Weibull-related distributions in addition to the well-known two- and three-parameter Weibull distributions that are found in the literature on dependability and statistics.

The purpose of this chapter is to provide a quick summary of these models, with a particular emphasis on those that may find use in future research. We begin with a summary of the traditional Weibull distribution and then explore its history, fundamental properties, and approaches to estimate and hypothesis testing, with a focus on graphical methods. We also go over a number of expansions and generalizations of the basic Weibull distributions. We also discuss some software for Weibull analysis and its uses in the reliability domain.

Contents

List of Figures

Abbreviations

Abbreviations	Description
TTF	Time To Failuer
α/η	Scale Parameter
β	Shape Parameter
T	Time

Chapter 1

Introduction

The Weibull reliability model is a very effective and adaptable instrument in the field of reliability engineering that is utilized to evaluate the dependability of various systems and components. Its use spans several industries, including telecommunications, healthcare, and manufacturing as well as aircraft. Engineers and analysts may use the Weibull model to obtain critical insights into the failure patterns of components over time. This information helps them make well-informed decisions about whether to schedule maintenance, improve product designs, and allocate resources.

The complexity of the Weibull reliability model is examined in this study, with an emphasis on how it may be used to forecast the time-to-failure values (TTF), or dependability of real-time data. In sectors like critical infrastructure, medical devices, and transportation systems where component failure can have serious consequences, it is crucial to comprehend the TTF distribution.

Additionally, this paper looks at how the Weibull dependability model is really implemented into a web application framework using Flask, a potent yet lightweight Python web framework. We hope to democratize access to reliability analysis tools by integrating the model into an intuitive online application, enabling engineers and other stakeholders to do reliable reliability evaluations with ease.

With this investigation, we hope to highlight the significance of the Weibull reliability model in improving systems' resilience and dependability, which will lead to safer, more effective, and more dependable operations in a variety of fields.

Chapter 2

Literature Review

2.1 What is Reliability?

Reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period.[1] Many fields, such as engineering, manufacturing, and psychology, rely heavily on the concept of reliability. According to engineering terminology, dependability is the probability that a system or component will function as intended for a particular amount of time under specific circumstances. The consistency and dependability of systems, goods, and procedures are evaluated using this metric, which eventually affects the safety, efficacy, and efficiency of those elements. Because they understand how important reliability is in preventing major repercussions like death, serious financial loss, or reputational harm, industries including aerospace, healthcare, automotive, and telecommunications place a special emphasis on it.

Metrics like Mean Time Between Failures (MTBF), Failure Rate (λ) , Probability of Failure (Pf), and Availability are frequently used to quantify dependability. These metrics guide decision-making processes about resource allocation, design, and maintenance by offering insights into the functionality and uptime of systems and components. There are several variables that affect reliability, including as design considerations, material quality, production methods, usage patterns, and maintenance procedures. Every one of these elements has the potential to affect a system's or component's overall dependability, thus in order to reduce risks and maximize performance, thorough analysis and management techniques are required.

The term "reliability engineering" refers to a variety of approaches and strategies used in the design, analysis, and optimization of systems and its constituent parts in order to attain specified reliability levels. To improve the dependability of systems and products, this interdisciplinary discipline uses techniques including reliability modeling, accelerated life testing, failure mode and effects analysis (FMEA), and reliability-centered maintenance (RCM). When evaluating the resilience and performance of systems and components under different stressors, such as load, vibration, temperature, and humidity, reliability testing is essential. Engineers may make well-informed design decisions and risk mitigation measures by identifying probable failure mechanisms and vulnerabilities in goods through rigorous testing regimens.

System and component reliability is estimated using mathematical models, historical data, or empirical evidence in reliability prediction and analysis. By using this technique, vulnerabilities may be found and designs can be optimized to increase dependability. Additionally, dependability is a dynamic quality that has to be continuously monitored, assessed, and improved upon throughout the course of a system's or product's lifetime. Organizations may minimize risks, increase overall performance, and improve customer happiness by adopting a continuous improvement culture. To put it simply, dependability is the foundation of quality and performance, encouraging innovation, safety, and consumer confidence in goods, services, and systems.

2.2 What is Weibull Reliability?

In reliability engineering, the Weibull reliability distribution is a fundamental statistical model that is widely used to assess the time-to-failure of systems or component parts. This distribution, named for its inventor Waloddi Weibull, is a mainstay in many different sectors because of its extraordinary versatility in capturing a broad range of failure patterns. Two fundamental factors, the shape parameter (β) and the scale parameter (η), constitute the foundation of the Weibull distribution. The failure rate may rise, fall, or stay constant over time depending on the shape parameter, which also controls the distribution curve's form. In the meantime, the scale parameter indicates the estimated period before a specific percentage of the population fails and reflects the typical life or scale of the distribution.

The Weibull distribution's probability density function (PDF) $f(x;\beta,\eta) = \frac{\beta}{\eta}\left(\frac{x}{\eta}\right)^{\beta-1} e^{-(x/\eta)^{\beta}}$ and reliability function characterize it mathematically. A component or system's likelihood of being operational after a predetermined amount of time is quantified by the reliability function, whereas the probability of detecting a particular time-to-failure value is described by the PDF.[2] Engineers and analysts may evaluate the failure characteristics and dependability of goods, equipment, and systems over time with the use of these mathematical formulas.

$R(t) = e^{-(t/\eta)^{\beta}}$ $where\ t = TTF \quad \eta = shapeparameterand\beta = scaleparameter$

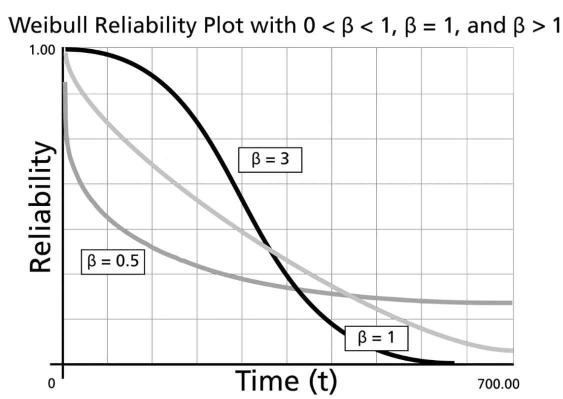

Figure 2.1: Weibull Distribution graphs on differnt $(\eta, \beta) values$

The Weibull reliability distribution finds extensive use in several domains, including lifespan data analysis, survival analysis, quality control, and reliability engineering. The Weibull distribution gives decision-makers information about the time-to-

failure behavior of systems and components, enabling them to make well-informed decisions about resource allocation, maintenance scheduling, and product design improvements.

2.3 Flask Frame Work

Flask is a Python web framework that has become quite popular due to its ease of use, flexibility, and simplicity. Flask was initially launched in 2010 and has since become a popular choice for many developers, especially those who value lightweight frameworks with minimalistic design principles.[3] Flask was developed by Armin Ronacher, the designer of prominent libraries like Jinja2 and Werkzeug.

Fundamentally, Flask represents the idea of a "micro-framework," meaning it offers only the necessities for web development—that is, none of the extraneous dependencies or complexity. Developers may construct architectures that closely match their unique needs and tastes and have more control over their projects with this minimalist approach.

The routing mechanism of Flask, which enables programmers to map URLs to Python functions called view functions, is one of its distinguishing characteristics. With Flask, routing may be accomplished by explicit route mapping or decorators, giving developers more freedom to specify how certain endpoints behave inside a web application. This facilitates the creation of RESTful APIs and the definition of the logic needed to handle different kinds of HTTP requests.

The integration of Flask with the robust and feature-rich Jinja2 template engine is another important feature. With Jinja2, developers can combine Python code, template expressions, and HTML markup to create dynamic HTML sites. With Jinja2's support for inheritance, macros, filters, and other sophisticated capabilities, creating intricate web interfaces with reusable components is simple.

Moreover, Flask comes with built-in support for managing HTTP requests and *replies, which includes functions for retrieving request data (like form data or request headers) and producing response data (like template rendering or JSON data return). Flask also has extensive error handling, request hooks, and session management, making it an ideal platform for developing web apps.*

Flask's extensibility is one of its advantages. An active community of extensions for the framework provides further features and capabilities. Numerous topics are covered by these extensions, such as cache, database connectivity, authentication, and authorization. Without having to start from scratch, developers may quickly expand Flask to match the unique needs of their projects by utilizing these extensions.

Flask is internally constructed using Werkzeug, a Python WSGI utility package. Flask uses Werkzeug's low-level HTTP request and response processing, routing, and other web-related features to create higher-level web development abstractions. Because of its underlying design, Flask is able to maintain its lightweight and fast nature while providing extensive functionality for creating web applications.

In conclusion, Flask is a strong and adaptable web framework that finds a happy medium between versatility and simplicity. Because of its straightforward architecture, copious documentation, and vibrant community support, it's become a preferred option for developers who want to create web apps fast and effectively. Flask is a compelling solution that may satisfy a broad range of demands and objectives, regardless of whether you're an experienced developer searching for a lightweight framework for your next project or a newbie trying to get started with web development.

2.4 Why should we use Flask?

There are several strong arguments in favor of using the Flask framework for web development projects. First off, Flask is well known for being straightforward. In contrast to more expansive frameworks that could have a challenging learning curve, Flask's minimalist design philosophy gives developers access to only the most necessary *capabilities for developing websites. Because of its simplicity, developers may easily understand the framework's fundamentals and begin creating apps without becoming mired down by needless complexity.*

Furthermore, Flask provides unmatched application architecture freedom. In contrast to frameworks that impose strict rules and patterns, Flask gives developers the freedom to create architectures that perfectly match their preferred coding styles and project needs. The versatility of Flask enables developers to customize their solutions for any kind of application, be it a large-scale enterprise application or a small-scale prototype.

Flask's extensibility is one of its best features. The framework has a thriving ecosystem of extensions that provide new features and functionalities to enhance its capabilities. Numerous topics are covered by these extensions, such as cache, database connectivity, authentication, and authorization. Without having to start from scratch, developers may quickly adapt Flask to the unique requirements of their projects by utilizing these extensions.

Scalability is also another important benefit of Flask. Despite being lightweight, Flask has all the necessary tools to effectively manage both small- and large-scale applications. Its simple design guarantees top performance, even with a lot of requests coming in. Furthermore, Flask apps may be set up on a variety of hosting platforms, such as cloud services and conventional web servers, giving you scalability choices to accommodate your application's expansion.

In addition, Flask has extensive documentation and a vibrant community. Flask has a vibrant community of developers that contribute to its continuous development via mailing lists, forums, and other means. The comprehensive and up-to-date documentation of the framework provides developers with a multitude of tools to enable them to study and handle issues with efficiency.

Finally, Flask easily interfaces with other Python frameworks and libraries, pro-

moting interoperability and letting programmers use already-existing tools and technologies in their work. Flask offers the versatility to work with a wide range of technologies, improving productivity and efficiency in development processes. These technologies include popular ORMs like SQLAlchemy, front-end frameworks like React or Angular, and authentication providers like OAuth.

To sum up, Flask is a strong and adaptable framework that shines in terms of community support, scalability, extensibility, flexibility, and integration possibilities. Whether you are an experienced developer looking for a lightweight yet reliable framework for your next project or a novice starting your first web development project, Flask provides the resources and tools you need to successfully realize your vision.

2.5 Applications of Flask

Flask is a versatile and lightweight web framework for Python that has a vast range of applications across several industries and use cases. Fundamentally, Flask is commonly used in variously difficult web development projects. Flask gives developers the resources and flexibility they need to realize their web-based ideas, whether they are for a large-scale social networking site, a blog, an e-commerce platform, or a content management system (CMS).

Moreover, creating RESTful APIs (Application Programming Interfaces) that enable communication between various software systems is a common use case for Flask. Flask's ability to create APIs for web services, mobile apps, Internet of Things (IoT) devices, and other software components makes it an important tool for facilitating smooth data integration and interchange across many platforms.

Apart from its usefulness in web development and API design, Flask is particularly impressive when it comes to microservices architecture. It is the perfect framework for developing microservices, which are systems made up of tiny, autonomous services that communicate with one another via APIs. This is because of its lightweight *architecture and minimalist approach. Software developers may create and implement microservices more quickly and easily because to Flask's flexibility and simplicity, which promotes software development's scalability, maintainability, and agility.*

Furthermore, Flask is often used in data visualization applications, especially in conjunction with Matplotlib, Plotly, and Bokeh packages. Through these interfaces, developers may produce visually stunning and interactive dashboards and data analysis tools that enable enterprises to effectively convey results through visualizations, get insights from data, and make choices.

Additionally, Flask is essential to efforts in data science and machine learning. Developers may implement recommendation engines, prediction models, and other data-driven applications by combining Flask with machine learning frameworks like TensorFlow, Scikit-learn, and PyTorch. This makes it possible to develop web-based applications that use AI and data analytics to address practical issues and provide insightful information.

Moreover, Flask is frequently used in fast development and prototyping settings, giving startups and small companies a quick and effective way to iterate on new concepts and get products to market quickly. While its versatility and extensibility accommodate more experienced learners and projects, its simple syntax and comprehensive documentation make it understandable to newcomers as well.

Beyond these uses, Flask is also used in educational contexts, such as online courses and academic institutions, to teach web development, Python programming, and related disciplines. Its features enable more complex projects and learning experiences, but its simplicity and adaptability make it a great tool for teaching students the foundations of web development and programming.

Furthermore, web-based interfaces for Internet of Things (IoT) devices may be constructed using Flask. Developers may construct IoT applications that improve convenience, efficiency, and productivity, such as smart home systems and industrial *automation solutions, by combining Flask with IoT platforms and protocols.*

All things considered, Flask's adaptability, ease of use, and extensibility make it a useful tool for developers working in a variety of fields and sectors, empowering them to quickly and effectively produce a wide range of web-based services and applications.

Chapter 3

Methodology

3.1 Predictive Maintenance

By using data analytics and machine learning, predictive maintenance is a proactive maintenance approach that foresees equipment faults before they happen.In contrast to conventional reactive or preventive maintenance methods, which depend on set timetables or pre-established criteria, predictive maintenance uses data to forecast the likelihood of equipment failure, enabling maintenance tasks to be carried out only when required.

Condition monitoring, which involves using sensors and other monitoring tools to continually gather information on the functionality and state of equipment, is the fundamental idea behind predictive maintenance. Numerous characteristics, including temperature, vibration, pressure, and lubrication levels, may be included in this data. This data is then subjected to advanced analytics methods, such as machine learning algorithms, to find patterns, trends, and anomalies that could point to upcoming equipment failures.

Predictive maintenance gives maintenance teams the ability to plan ahead and reduce downtime by studying past data and seeing early warning signals of impending breakdowns. This might include arranging for maintenance tasks to be completed *during scheduled downtime or swapping out components before they catastrophically fail. [4] Predictive maintenance also makes it possible to deploy maintenance resources more effectively, concentrating efforts where they are most required and cutting down on pointless maintenance chores.*

Predictive maintenance has a number of important advantages overall, such as higher equipment uptime, enhanced safety, and lower maintenance costs. Organizations may optimize their maintenance processes, improve operational efficiency, and ultimately achieve improved dependability and productivity by anticipating equipment breakdowns and adopting proactive efforts to rectify them.

3.2 Mine Machinery Predictive Maintenance

A fundamental shift in the way the mining sector maintains its equipment is represented by predictive maintenance in mine gear. Failures are no longer only to be responded to; instead, proactive steps must be taken to anticipate and prevent them from happening.[5] Predictive maintenance is revolutionary in a field where downtime can result in significant financial losses and safety issues.

Predictive maintenance fundamentally depends on using a large amount of data. Mine gear is equipped with sensors that are constantly gathering data on temperature, vibration, pressure, and other variables. Predictive analysis is based on this data, prior maintenance records, and environmental variables. The data is preprocessed to remove errors, deal with missing numbers, and get it ready for modeling before it can be examined.

Predictive maintenance relies heavily on feature engineering. From the raw data, engineers identify valuable traits that are used as inputs into prediction models. These characteristics might be time-series patterns, statistical measurements, or domain-specific indications of the health of the equipment. This data is then subjected to machine learning algorithms, which range from regression to more complex anomaly *detection methods, in order to create prediction models.*

These models are introduced into the mine's operating environment after they have been trained and verified. Here, they constantly keep an eye on the functionality and health of the gear in real time. The models warn maintenance workers when they notice deviations from typical behavior or probable failure indications. With this advance notice, maintenance staff may take preventative measures, such arranging repairs during scheduled downtime or swapping out components before they tragically fail.

There's more to predictive maintenance than that. By focusing resources where they are most required, it helps mining businesses improve their maintenance plans. Businesses may keep consistent production levels and steer clear of expensive disruptions by averting unplanned malfunctions. Furthermore, by lowering the possibility of mishaps brought on by malfunctioning equipment, the capacity to anticipate breakdowns in advance improves safety.

However, predictive maintenance is a continuous process of improvement rather than a one-time fix. Predictive models must be regularly updated and improved upon when new data becomes available and machinery is modified. By using an iterative process, the models are guaranteed to be accurate and functional throughout time, adjusting to changing equipment behavior and operating conditions.

In conclusion, predictive maintenance, which uses data and sophisticated analytics to forecast and avoid problems, is a proactive approach to equipment management in mining operations. By doing this, mining businesses may improve safety, reduce downtime, and increase operational efficiency, all of which will contribute to the industry's overall profitability and sustainability.

Chapter 4

Python Model and Dataset

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import minimize

# Load TTF data from an Excel file
excel_file = r'C:\Users\LENOVO\Downloads\Data_TTF (2).xlsx'  # Specify the path to your Excel file
df = pd.read_excel(excel_file)
ttf_data = df['TTF'].values  # Assuming 'TTF' is the column name containing the time-to-failure data

# Define the negative log-likelihood function for Weibull distribution
def neg_log_likelihood(params, data):
    alpha, beta = params
    return -np.sum(np.log((beta/alpha) * (data/alpha)**(beta - 1) * np.exp(-(data/alpha)**beta)))

# Initial guess for parameters (alpha, beta)
initial_guess = [np.mean(ttf_data), 0.5]
# Minimize the negative log-likelihood function to estimate parameters
result = minimize(neg_log_likelihood, initial_guess, args=(ttf_data,))
alpha_hat, beta_hat = result.x
#print("Estimated alpha (scale parameter):", alpha_hat)
#print("Estimated beta (shape parameter):", beta_hat)
# Calculate Weibull reliability distribution for TTF data
ttf_range = np.linspace(0, np.max(ttf_data), 100)  # Range of TTF values for plotting
reliability = np.exp(-(ttf_range / alpha_hat)**beta_hat)
# Plot the Weibull reliability distribution
plt.figure(figsize=(8, 6))
plt.plot(ttf_range, reliability, label=f'Weibull Reliability (alpha={alpha_hat:.2f}, beta={beta_hat:.2f})')
plt.xlabel('Time')
plt.ylabel('Reliability (R(TTF))')
plt.title('Weibull Reliability Distribution')
plt.legend()
plt.grid(True)
plt.show()
```

Figure 4.1: Python moel

Libraries used import numpy as np, import pandas as pd, import matplotlib.pyplot as plt, from scipy.optimize import minimize

4.0.1 NumPy (np)

The foundational library for numerical computation in Python is called NumPy, or Numerical Python. Large, multi-dimensional arrays and matrices may be handled with its assistance, and a vast array of mathematical functions is available for effective array operations. NumPy is a popular tool for tasks like array manipulation, statistical computations, and linear algebra operations because of its speed and adaptability. Applications in scientific computing, data analysis, and machine learning depend heavily on its capabilities.

4.0.2 Pandas(pd)

Pandas is a Python module that is essential for data analysis and manipulation. Building on the foundation NumPy gave, it includes strong data structures like Series and DataFrames that are intended to handle organized and labeled data. Users may quickly load, purify, convert, and examine datasets from a variety of sources, such as SQL databases, CSV files, and Excel spreadsheets. Data scientists, analysts, and researchers find it indispensable because of its user-friendly and expressive API, which makes it possible to do operations like data indexing, selection, filtering, aggregation, and visualization.

4.0.3 Matplotlib (plt)

With a wealth of features for producing static, interactive, and publication-quality visuals, Matplotlib is a standout Python charting toolkit. It offers an interface that is similar to MATLAB for creating a wide range of graphs, such as line, scatter, bar, and histogram plots, among others. Users may fine-tune every element of their *plots, such as the axes, labels, colors, markers, and styles, with Matplotlib. Because of its adaptability and flexibility, it is the go-to option for data visualization projects in industries like finance, engineering, science, and data analysis.*

4.0.4 SciPy.optimize.minimize

Known for its extensive array of scientific computing tools and techniques, SciPy is a companion library to NumPy. The 'optimize' module of the SciPy library provides a range of optimization and root-finding techniques to address various numerical optimization issues. The'minimize' function stands out among these algorithms in especially important ways. It is used to apply various optimization strategies to minimize a scalar function of one or more variables. The function "minimize," which is frequently used for activities like parameter estimation, curve fitting, and optimization in the scientific and engineering fields, is essential to the effective solution of challenging optimization issues.

4.1 Stroing Data

Time-to-failure (TTF) data is loaded into a Pandas DataFrame from an Excel file in this code section. The directory path to the file on the local system is included in the 'excel' variable, which is used to provide the Excel file path. The Excel file is then read using the Pandas library's 'pd.read()' method, which also creates a DataFrame ('df') with the data within.

The syntax 'df['TTF']' is used to extract the TTF values from the DataFrame's column labeled 'TTF' once the data has been put into it. Assumedly, the time-to-failure data is stored in an Excel file under a column labelled 'TTF'. Subsequently, the TTF column data is transformed into a NumPy array '(ttf)' using the '.values' property, therefore enabling additional processing and analysis.

In conclusion, this little piece of code effectively pulls TTF data from an Excel *file, converts it into a format that NumPy can handle, and gets it ready for further analysis like statistical modeling or visualization.*

4.2 Generating $(\alpha/\eta, \beta)$

In order to find the parameters that best suit the provided time-to-failure (TTF) data, this code snippet creates a negative log-likelihood function for the Weibull distribution, estimates its parameters, and minimizes the function.

The negative log-likelihood function is defined as follows:

$$\text{neg\_log\_likelihood}(\alpha, \beta, \text{data}) = -\sum_{i=1}^{n} \log\left(\frac{\beta}{\alpha}\left(\frac{\text{data}_i}{\alpha}\right)^{\beta-1} \exp\left(-\left(\frac{\text{data}_i}{\alpha}\right)^{\beta}\right)\right)$$

Figure 4.2: neg_log_likelihood

where:

- *and are the parameters of the Weibull distribution,*
- ***datai*** *represents each individual TTF observation*
- ***n*** *is the total number of observations*

The function takes a collection of arguments and TTF data and returns the negative log-likelihood value. Finding the ideal values for and that best suit the given TTF data is the aim of minimizing this function.

The 'initial_guess' variable yields a preliminary approximation for the parameters and . This is essential to get the optimization process started.

The negative log-likelihood function is then minimized using the'minimize' function from the SciPy library. As input inputs, it requires the TTF data ('ttf_data'), initial parameter estimates ('initial_guess'), and the negative log-likelihood function ('neg_log_likelihood'). The estimated parameters (hat) are provided via the output ('result.x').values of hat and hat) that reduce the negative log-likelihood function.

4.3 Calculating weibull

The Weibull reliability distribution for the provided time-to-failure (TTF) data is computed by the algorithm. The Weibull distribution is often used in reliability analysis to simulate the time before a system or component breaks. The likelihood that the system will continue to work after a predetermined amount of time is represented by the Weibull distribution's reliability function.

Initially, an array ('ttf_range') containing 100 equally spaced TTF values between 0 and the maximum TTF seen in the data is created using NumPy's np.linspace() method. Plotting the dependability distribution properly across a wide enough range of time durations requires this range of TTF values.

The Weibull distribution's reliability function is then computed for every value in the 'ttf$_r$ange$'.Thefollowingformulaisapplied:$

$$R(t) = e(t/(\alpha)hat) * *(\beta)hat$$

where

- ***R(t)*** *represents relibilty in time*

- **(**$\alpha)hatand(\beta)hatareestimatedperameter's.$

Weibull distribution derived from the process of optimization. The estimated parameters are used to generate the relevant reliability value for each TTF value in the 'ttf_range'. The'reliability' array holds the outcome.

Based on the fitted Weibull distribution, the computed reliability values indicate the likelihood that the system will continue to function after reaching each corresponding TTF value. The system's dependability over time may be shown by charting the reliability curve, which is one way to use these reliability metrics for additional analysis or visualization.

4.4 Plot graph

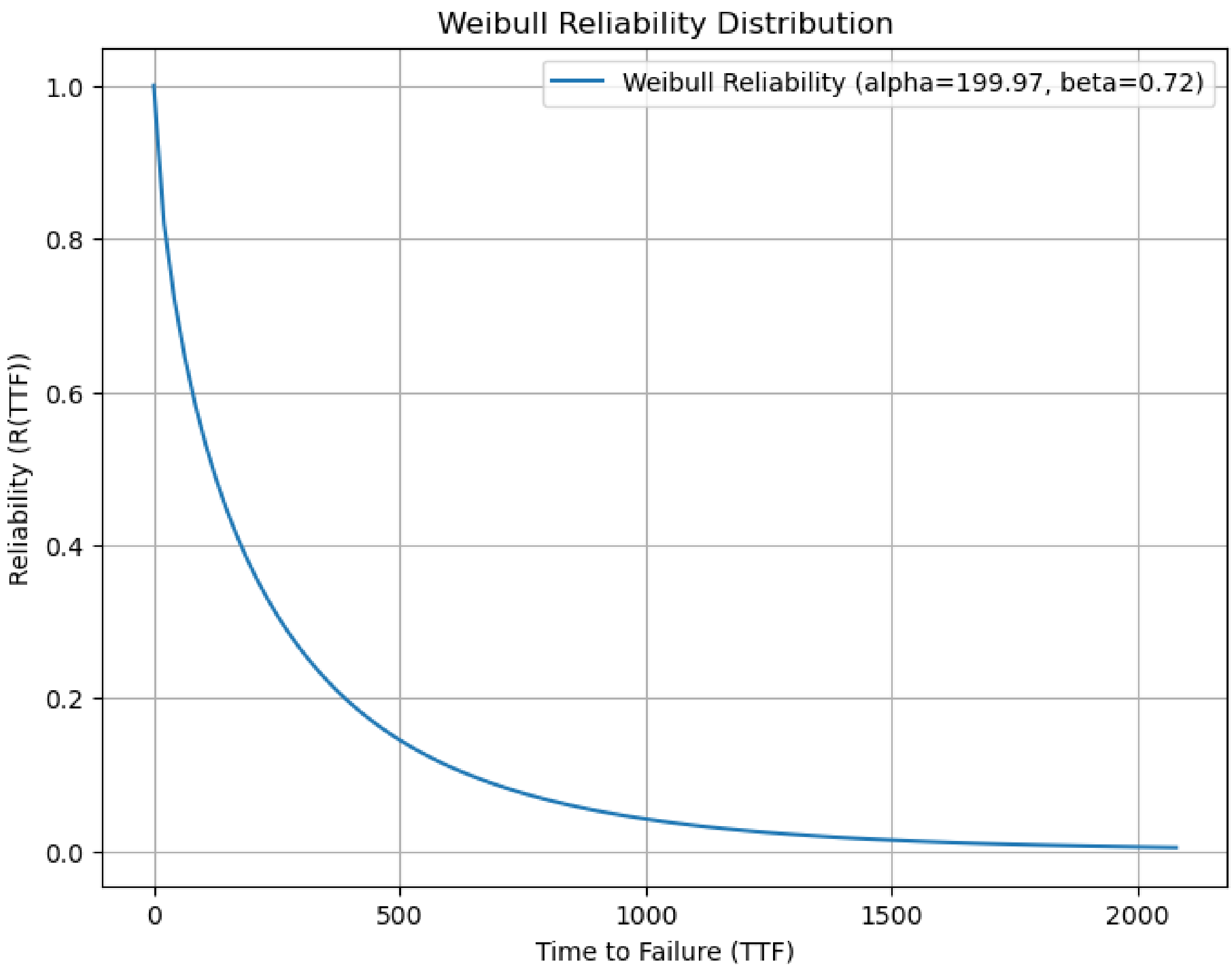

Figure 4.3: Required Output

This code segment uses the computed parameters "alpha_hat" and "beta_hat" to create a figure that visualizes the Weibull reliability distribution. The line 'plt.figure(figsize=(8, 6))' creates a new figure that is 8 inches by 6 inches in size, which works well as a plot canvas.

The real plot is then made using the 'plt.plot()' method. It requires two primary inputs: reliability, which holds the relevant "reliability" values determined using the Weibull distribution parameters, and "ttf_range," which indicates the range of time-to-failure values. The estimated parameters "alpha_hat" and "beta_hat" are included *in the label parameter, which is used to generate a meaningful name for the graphic.*

'plt.xlabel()' and 'plt.ylabel()' are then used to establish labels for the x- and y-axes, respectively, giving the plot's interpretation a clear context. The plot is given a title by the 'plt.title()' function, which indicates that it depicts the Weibull reliability distribution.

By adding a legend to the plot and displaying the label that was previously supplied in the 'plt.plot()' function, plt.legend() ensures clarity. Furthermore, grid lines are activated using 'plt.grid(True)' to aid in deciphering the numbers and connections within the graphic. The plot is finally shown to the user using the 'plt.show()' method. All things considered, this code segment makes it easier to see the Weibull reliability distribution and provides a graphical depiction of the examined system's dependability over time using the estimated Weibull parameters.

Chapter 5

Website Building

I developed a thorough website as part of my undergraduate thesis to help with predictive maintenance analysis for equipment and systems. The website provides customers with an interactive and user-friendly platform to access and utilize a predictive maintenance model that is based on the Weibull dependability distribution. This is achieved by leveraging a combination of front-end and back-end technology.

With the use of HTML, CSS, and JavaScript, the website's front end was created to have an aesthetically pleasing and intuitive user experience. The front-end offers users smooth navigation and interaction through thoughtful design and execution. Options for entering pertinent data, adjusting model parameters, and visualizing predictive maintenance analysis outcomes are shown to users. The focus was on developing an interface that is both intuitive and responsive, which improves user experience and makes data entry and interpretation more efficient.

Flask is a lightweight, flexible web framework for Python that powers the website's back-end functionality, which complements its front-end capabilities. Behind the scenes, Flask acts as the engine, processing user input, managing incoming requests from the front-end interface, and coordinating interactions with the predictive maintenance model. The website and the predictive model interact dynamically using

Flask, enabling seamless integration and dependability prediction execution.

The Flask environment is used to deploy the predictive maintenance model, which was created using Python modules including NumPy, Pandas, and SciPy. This model estimates parameters and makes predictions about the long-term dependability of equipment or systems using statistical analysis and optimization approaches. By entering pertinent data, such time-to-failure (TTF) values, users may get useful insights into the predicted reliability performance of the systems under analysis. The findings of the model are presented in an understandable and aesthetically pleasing way, making it easier for users to comprehend and make use of the data. The website offers a comprehensive predictive maintenance analysis solution by integrating JavaScript, HTML, CSS, and Flask. The website allows users to make educated decisions about resource allocation and maintenance planning by bridging the gap between front-end user interface design and back-end model execution. This multidisciplinary approach not only exhibits expertise with web development technologies but also shows how statistical modeling approaches can be applied in the real world to solve reliability engineering and maintenance management problems.

5.1 Interface of the website

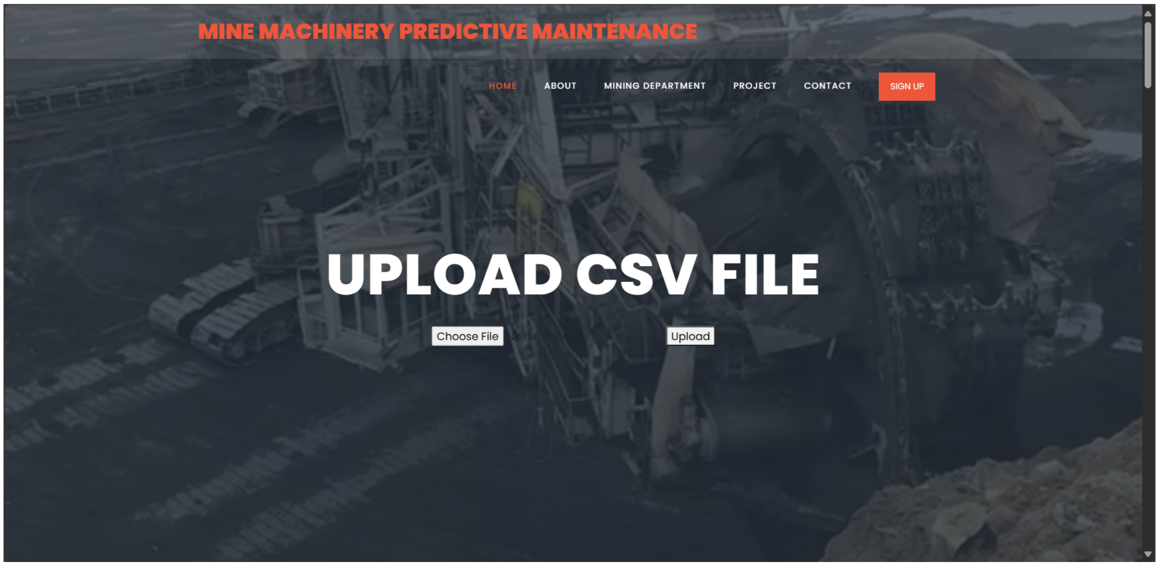

Figure 5.1: Home

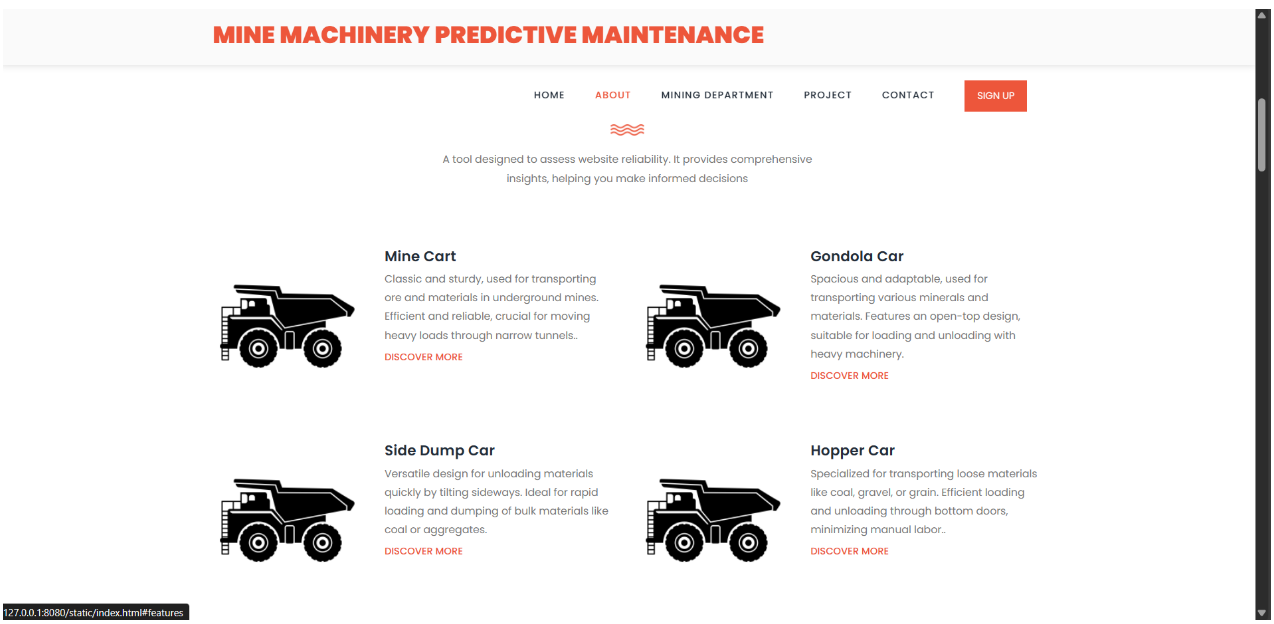

Figure 5.2: About

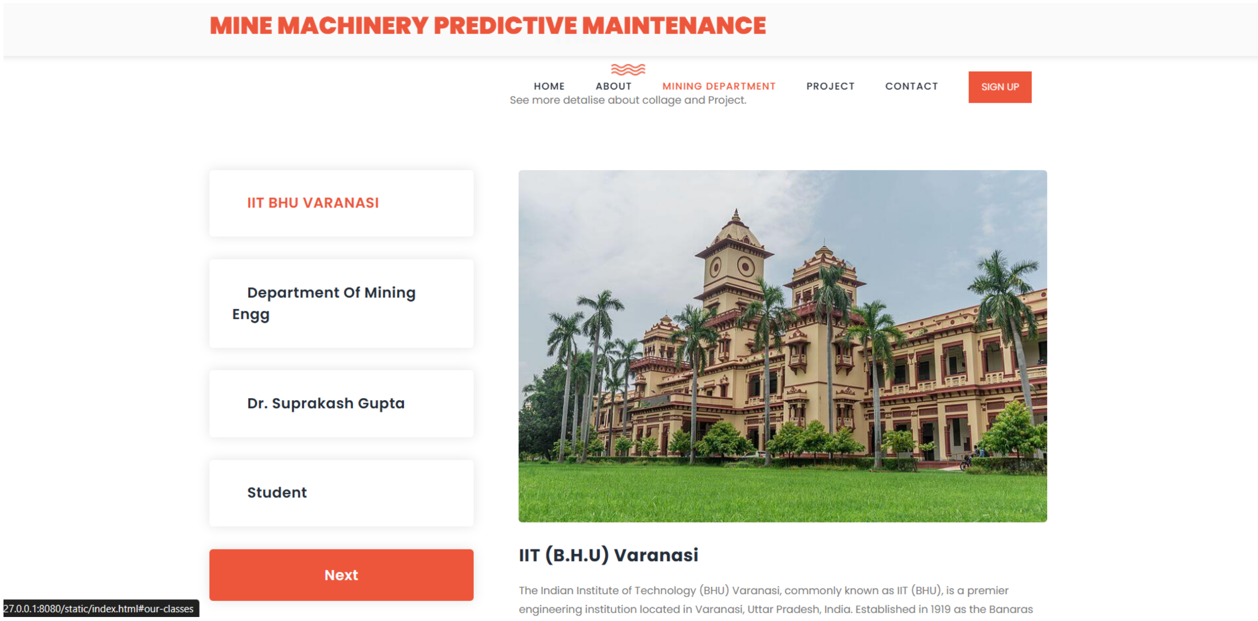

Figure 5.3: Details

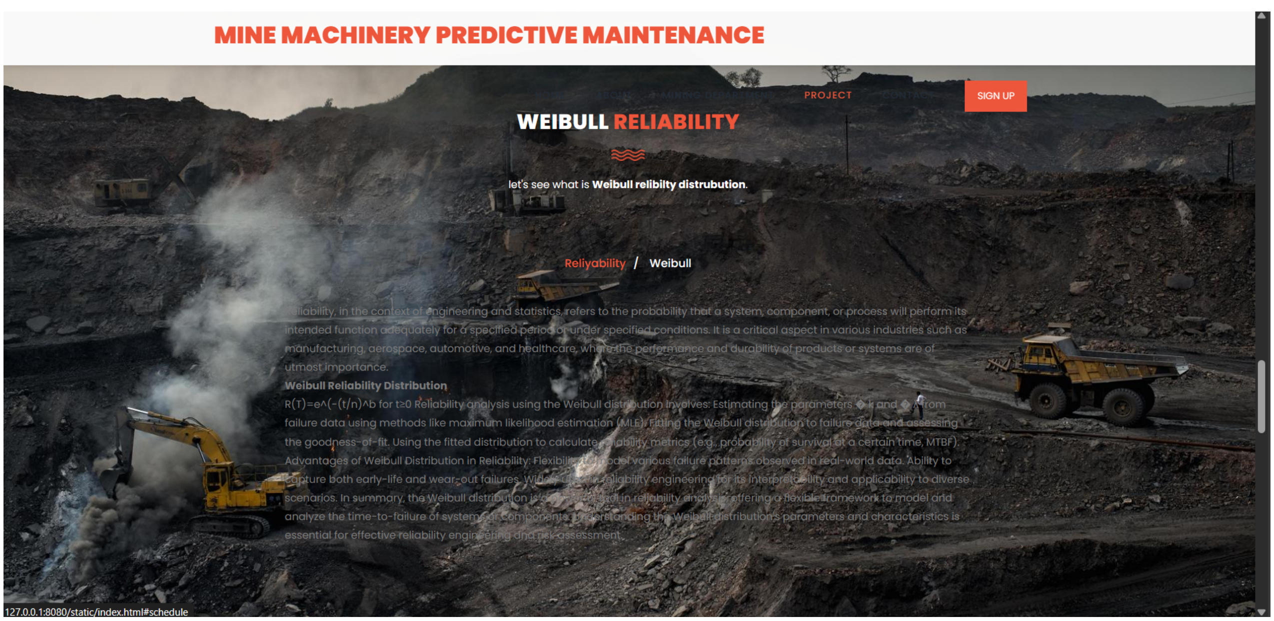

Figure 5.4: Weibull Distribution

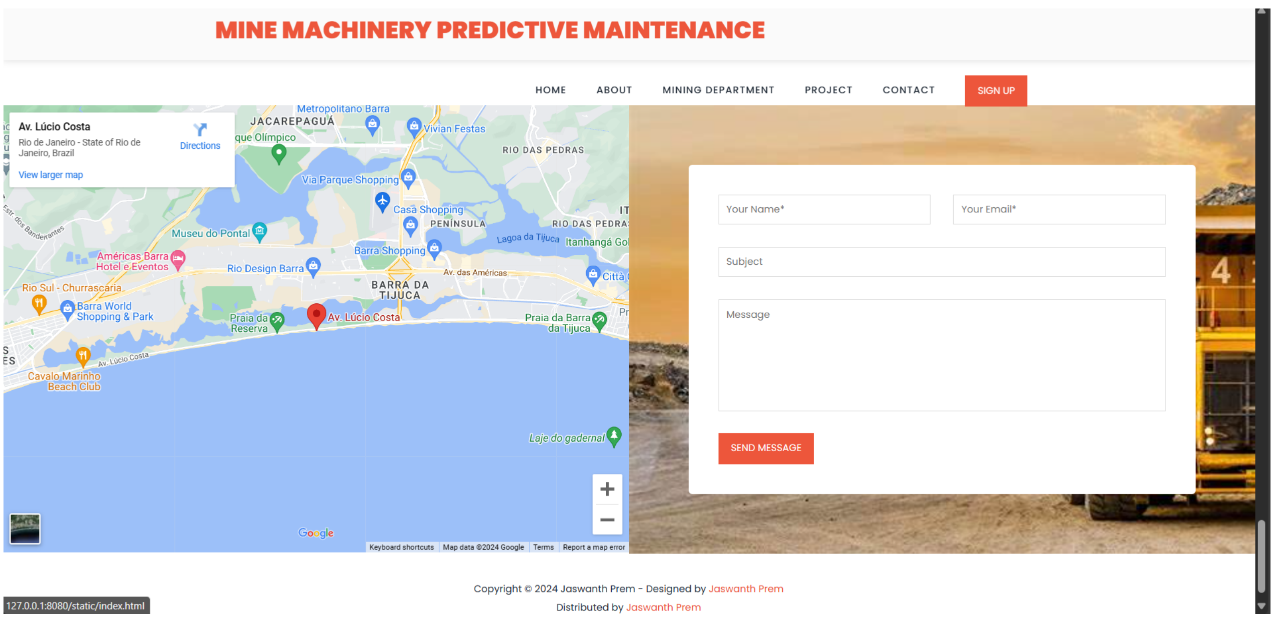

Figure 5.5: Contact us

Chapter 6

Conclusion

In conclusion, the discussion included a range of topics related to web development, predictive maintenance, and how these fields might be combined in real-world project settings. We looked at the idea of predictive maintenance, its significance in sectors like mining, and the methods and resources needed to put predictive maintenance models into practice. We also explored the Flask web framework and its uses in back-end programming, namely in the implementation of predictive models in online applications.

Additionally, we talked about front-end development with HTML, CSS, and JavaScript, emphasizing their function in designing user-friendly interfaces for predictive maintenance model interaction. It was stressed that integrating these front-end technologies with Flask's back-end features was an essential component of creating a thorough web application for predictive maintenance analysis.

The creation of a website that gives users access to and usage of a predictive maintenance model based on the Weibull reliability distribution serves as an example of how these ideas are put into practice. This project demonstrated how front-end and back-end technology might be seamlessly integrated to provide a unified platform for predictive maintenance analysis.

Overall, the discussion emphasized the multidisciplinary nature of web development and predictive maintenance, emphasizing the value of utilizing a variety of technology and skill sets to tackle challenging real-world problems. Organizations may optimize their maintenance procedures, boost operational effectiveness, and eventually attain increased productivity and dependability by incorporating predictive maintenance models into online applications.

Chapter 7

Future Development of this Project

Implementing functionality at the component level and incorporating a database for data storage and retrieval might be future project developments.

To improve the predictive maintenance analysis, more characteristics and functions might be added at the component level. This can entail improving the user interface to enable more user-friendly data input, adding sophisticated visualization tools to present analysis results in a more interactive way, and putting in place real-time monitoring features to update the predictive model in real-time as new data comes in. Additionally, to notify users of important maintenance events or abnormalities found by the system, the usage of automated alerts and notifications might be investigated. These improvements would increase the predictive maintenance solution's scalability and applicability to various industry applications in addition to its efficacy and usefulness. Simultaneously, the project design might benefit from incorporating a database to facilitate the effective storage and retrieval of data produced by the predictive maintenance analysis. This database might serve as a central repository for pertinent data by storing previous maintenance records, equipment specs, sensor data, and analytical findings. Through the utilization of a database management system (DBMS) like MySQL, PostgreSQL, or MongoDB, users may effortlessly retrieve and *examine the data, do trend analysis, and produce reports for additional examination or decision-making. Furthermore, the dependability and integrity of the stored data would be guaranteed by putting best practices for data management into effect, such as data normalization, indexing, and data integrity requirements. All things considered, the database integration and component-level implementation mark important turning points in the project's future evolution. These improvements would not only improve the predictive maintenance solution's functioning but also set the stage for the system's long-term upkeep, scalability, and extensibility. Through the integration of these functionalities, the project hopes to offer an extensive and resilient framework for predictive maintenance analysis, enabling establishments to maximize their maintenance procedures and attain increased operational effectiveness.*

Bibliography

[1] *A. Bruton, J. H. Conway, and S. T. Holgate, "Reliability: What is it, and how is it measured?"* Physiotherapy*, vol. 86, no. 2, pp. 94–99, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031940605612114*

[2] *C.-D. Lai, D. Murthy, and M. Xie, "Weibull distributions and their applications,"* Springer Handbook of Engineering Statistics*, vol. Chapter 3, pp. 63–78, 02 2006.*

[3] *F. Aslam, H. Mohammed, and P. Lokhande, "Efficient way of web development using python and flask."* International Journal of Advanced Research in Computer Science*, vol. 6, 01 2015.*

[4] *F. Raza, "Ai for predictive maintenance in industrial systems," 11 2023.*

[5] *O. Dayo-Olupona, B. Genc, T. Celik, and S. Bada, "Adoptable approaches to predictive maintenance in mining industry: An overview,"* Resources Policy*, vol. 86, p. 104291, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0301420723010024*