NILKAMAL SCHOOL OF MATHEMATICS, APPLIED STATISTICS & ANALYTICS

A Study On Analysing Failure Rate Of Storage Devices Using New Weibull Log Logistic Grey Forecasting Model

A DISSERTATION SUBMITTED TO

SVKM'S NMIMS (DEEMED TO BE UNIVERSITY)

IN PARTIAL FULFILMENT FOR THE DEGREE OF

MASTERS OF SCIENCE

IN

STATISTICS AND DATA SCIENCE

BY

RITESH PATIL

DARSHANA PARLE

SEJAL MAHADIK

REETIK PRAJAPATI

NISHAD SANGALE

UNDER THE SUPERVISION OF

PROF. VAIBHAV VASUNDEKAR

NILKAMAL SCHOOL OF MATHEMATICS, APPLIED STATISTICS AND ANALYTICS

SVKM's Narsee Monjee Institute of Management Studies

(Deemed-To-Be-University)

V.L. Mehta Rd, Vile Parle (West), Mumbai – 400056

NOV- 2024

INDEX

Acknowledgement

We wholeheartedly thank our project mentor, Prof. Vaibhav Vasundekar for providing valuable guidance and support in all stages of our project work, his meticulous insights helped us to improve the quality of the project we have been making. Our project greatly benefited from his statistical knowledge and experience which were useful at the critical stages of the project.

We would like to show our gratitude towards the SDS course coordinator Dr. Pradnya Khandeparkar and Prof. Rushina Singhi for providing us with this opportunity to work on this project. We extend our heartfelt gratitude to all the faculty members for their support, suggestions, and encouragement.

Finally, we extend our heartfelt gratitude to all the people who directly or indirectly are involved with our project.

Motivation

The motivation behind this topic, “A Study of Analysing Failure Rate of Storage Devices Using New Weibull Log Logistic Grey Forecasting Model” is to improve the accuracy and stability of predicting hard disk drive (HDD) failures. Current HDD failure prediction methods often suffer from high false positive rates and significant resource consumption, which affects their overall efficiency. This paper introduces a novel forecasting model based on the Weibull Log-Logistic distribution, which is designed to capture the complex nonlinear interactions between factors that lead to HDD failure. By addressing the limitations of existing statistical and machine learning models, the proposed model incorporates a new accumulation generation operator and multiple interaction effects between variables, allowing for more precise failure prediction. This improvement can significantly reduce the economic impact of false alarms in early warning systems, enhancing the reliability of server operations that depend on the continuous functionality of hard disk drives.

Abstract

The paper presents a new forecasting model to predict hard disk drive (HDD) failures using a novel Weibull Log-Logistic Grey forecasting approach. HDD failure prediction is critical for the reliability of servers and storage systems. However, existing methods have limitations, such as high false positive rates and significant resource usage, which lead to inefficient early warning systems. This paper aims to address these issues by developing a more accurate and stable model that can effectively predict HDD failures and reduce economic losses caused by false alarms.

The new model is based on the Weibull Log-Logistic mixture distribution, which helps to better capture the failure trends in HDD data by reducing volatility and fitting complex patterns in the data. The model also introduces a new accumulation generation operator to improve the model's flexibility in handling nonlinear data. This operator helps in reducing the influence of erratic changes in the data, making the predictions more stable.

Another key feature of the model is the inclusion of multiple interaction effects, which allows it to consider the combined impact of various independent factors that influence HDD failure, such as temperature, disk activity levels, and SMART (Self-Monitoring, Analysis, and Reporting Technology) attributes. These attributes, collected by HDDs to monitor their own health, are critical in identifying potential failures. By considering the interaction between these factors, the model enhances its predictive power and accuracy.

The given model is applied to real-world data from four different HDD types provided by BackBlaze, a company that tracks HDD failures. The results demonstrated that the new model consistently outperformed existing models in terms of prediction accuracy. The model achieved lower Mean Absolute Percentage Error (MAPE) values, indicating that it was better at forecasting HDD failures with fewer false positives and improved reliability.

The paper concludes that the new Weibull Log-Logistic Grey forecasting model can be a valuable tool for companies that rely on large-scale data storage systems. It provides a more accurate and resource-efficient method for predicting HDD failures, which could lead to significant cost savings by preventing data loss and minimizing the need for costly, unnecessary HDD replacements. This paper also suggest that future research could further refine the model and explore its application to other types of mechanical systems with similar failure characteristics.

Keywords: Weibull Log-Logistic Grey forecasting model, Hard disk drive (HDD) failure prediction, Multiple interaction effects, Accumulation generation operator, SMART (Self-Monitoring, Analysis, and Reporting Technology) attributes.

Introduction

HDDs are a critical component of modern data infrastructure, from personal computers to large-scale cloud storage systems used by corporations and organizations. As the amount of data stored on these systems continues to grow, the need for reliable and efficient prediction methods to anticipate HDD failures has become increasingly important.

The failure of an HDD can lead to significant problems, including data loss, system downtime, and the high cost of replacement and recovery. Therefore, the ability to predict HDD failures before they occur is crucial in avoiding such disruptions. Effective early warning systems help companies minimize the risk of data loss and ensure smoother operations.

However, predicting HDD failures is a complex task. Current predictive models and techniques, while useful, have notable limitations. Many existing methods have high false positive rates, meaning they often mistakenly predict failure when there is none. This results in unnecessary replacements of HDDs, which leads to extra costs and wasted resources. Moreover, the accuracy of current models can be unreliable, especially when dealing with large and complex datasets. As a result, there is a need for new approaches that can provide more accurate and efficient predictions.

This paper aims to tackle these issues by proposing a new forecasting model, called the Weibull Log-Logistic Grey forecasting model, which is designed to improve the accuracy of HDD failure predictions while reducing false positives. The model builds on the strengths of two statistical distributions – Weibull and Log-Logistic – which are widely used in failure analysis and reliability studies. By combining these distributions, the proposed model captures the complex patterns and trends in HDD failure data more effectively.

In addition to this, the model introduces a new accumulation generation operator. This operator plays a crucial role in stabilizing the model by making it more flexible in handling nonlinear data. This is particularly important in the context of HDD failure prediction, where the data is often subject to fluctuations and erratic changes. The accumulation generation operator helps smooth out these fluctuations, enabling the model to provide more consistent and stable predictions.

Another important aspect of the proposed model is its ability to consider multiple interaction effects. HDD failures are rarely caused by a single factor. Instead, they are typically the result of a combination of various independent factors such as temperature, usage intensity, and the health status of the drive as monitored through SMART (Self-Monitoring, Analysis, and Reporting Technology) attributes. SMART attributes are a set of diagnostic features that HDDs use to assess their own health, and they provide valuable

information that can be used to predict potential failures. By incorporating the interaction between these factors, the model increases its predictive power and accuracy.

To test the effectiveness of the new model, we applied it to real-world data obtained from BackBlaze, a company that tracks HDD failures. BackBlaze's dataset includes information from four different types of HDDs, making it a valuable resource for validating the performance of the forecasting model. The results of the study showed that the Weibull Log-Logistic Grey forecasting model consistently outperformed existing methods in terms of accuracy. The model was able to predict failures with lower Mean Absolute Percentage Error (MAPE) values, which means it produced fewer errors and false positives compared to other models.

In conclusion, the introduction sets the stage for the rest of the paper by explaining the importance of HDD failure prediction and the limitations of current methods. The given research paper argue that their new Weibull Log-Logistic Grey forecasting model can help overcome these limitations by providing a more accurate, stable, and resource-efficient solution. The model's ability to capture complex patterns in HDD failure data and its consideration of multiple interaction effects make it a valuable tool for companies that rely on large-scale data storage systems. By reducing false positives and improving prediction accuracy, the model can help prevent costly data loss and minimize unnecessary HDD replacements.

Objectives

1. To test a new forecasting model to accurately predict hard disk drive (HDD) failures, improving the reliability of data storage systems.

2. To capture and fit the trend of the hard disk failure data flexibly and reduce the volatility.

3. To better handle the multiple interaction effects in hard disk drive failure data and reduce the economic losses caused by the false alarms in early warning system.

4. To study the grey model with non-linear and interactive features to forecast the failure trend of hard disk drives.

5. Implement the model to minimize incorrect failure predictions, reducing unnecessary HDD replacements and associated costs.

6. To apply the model to real-world HDD failure data for practical validation.

Methodology

The methodology follows several stages, beginning with the theoretical foundation from the Weibull and Log-Logistic distributions, proceeding with model development, parameter estimation, and evaluation on real-world data sets.

1. Data Preprocessing

Data Cleaning: The raw dataset from Backblaze is preprocessed to remove irrelevant and redundant data entries, ensuring only critical data attributes related to hard drive failures are retained.

SMART Attribute Selection: Based on prior studies and correlations identified within SMART data, specific attributes were selected for the model, including:

- **Reallocated Sectors Count** (SMART5),
- **Reported Uncorrectable Errors** (SMART187),
- **Command Timeout** (SMART188),
- **Current Pending Sector Count** (SMART197),
- **Uncorrectable Sector Count** (SMART198).

The selected SMART attributes are normalized to maintain consistency across different data scales, ensuring model stability and improving prediction accuracy.

2. Model Formulation: Weibull Log-Logistic Grey Forecasting Model (WLLGM)

- **Distribution Choice**: The Weibull Log-Logistic (WLL) distribution is chosen to handle failure data due to its flexibility in modeling time-to-failure data and accommodating non-linear behavior. This distribution combines the Weibull and Log-Logistic distributions to capture the variability and trend in the failure data more effectively than traditional distributions.

  1. Weibull Distribution-

  - **Weibull Distribution Function:**

$$f_1(t;\psi_1) = \frac{\beta_1}{\alpha_1}\left(\frac{t}{\alpha_1}\right)^{\beta_1-1} e^{-\left(\frac{t}{\alpha_1}\right)^{\beta_1}}$$

  2. Log-Logistic Distribution-

  - **Log-Logistic Distribution Function:**

$$f_2(t;\psi_2) = \frac{\beta_2}{\alpha_2}\left(\frac{t}{\alpha_2}\right)^{\beta_2-1}\left(1+\left(\frac{t}{\alpha_2}\right)^{\beta_2}\right)^{-2}$$

3. **Weibull Log-Logistic Mixture Distribution-**

- **Weibull Log-Logistic Mixture Distribution:**

$$f(t;\psi) = pf_1(t;\psi_1) + (1-p)f_2(t;\psi_2)$$

- **Accumulation Generation Operator**: To account for time dependencies and trends in failure data, an accumulation generation operator based on the WLL mixture distribution is introduced. This operator applies a weighted accumulation across data points, reducing volatility and smoothing the data to enhance the model's adaptability to the nonlinear characteristics of failure patterns in hard drives.

$$x^{(1)}(k) = \sum_{i=1}^{k} h(k-i+1,\psi)x^{(0)}(i)$$

- **Model Structure**: The proposed model, WLLGM(1, N), is based on the GM(1, N) grey model but extends it to include multiple interaction effects among SMART attributes. This model incorporates both direct effects and interaction effects between SMART attributes, capturing the compound influence of these variables on hard drive failure probabilities.

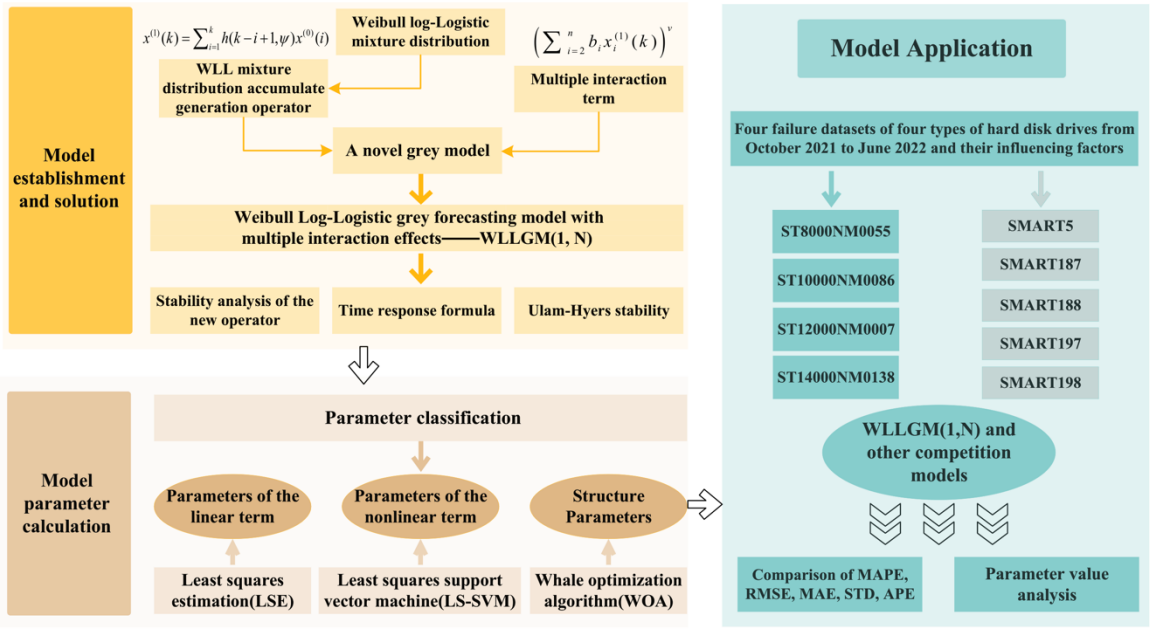


Fig. 1. Full-text framework diagram.

3. Parameter Estimation and Optimization

Parameter estimation for the WLLGM model is executed through a multi-step approach, applying both traditional and advanced optimization techniques to handle the linear, nonlinear, and complex interaction parameters within the model.

- **Linear Parameters**: Parameters representing linear relationships are estimated using the Least Squares Method.
- **Nonlinear Parameters**: Parameters that capture the nonlinear relationships among SMART attributes are estimated using the Least Squares Support Vector Machine (LS-SVM). LS-SVM offers advantages in handling small sample sizes and reducing sensitivity to outliers.
- **Structural Parameters**: Control parameters, such as the exponent for interaction terms and scaling factors, require meta-heuristic optimization due to the complex search space.
- **Optimization Using Whale Optimization Algorithm (WOA)**:
  The Whale Optimization Algorithm (WOA) is employed to optimize structure parameters like the power exponent and the control coefficient in the WLLGM model.
  WOA is chosen for its ability to efficiently handle complex, non-linear optimization problems by simulating the social behavior of humpback whales. This algorithm iteratively refines the parameter values to minimize the Mean Absolute Percentage Error (MAPE) of the model's predictions.

4. Multiple Interaction Effects Modeling

To capture the compound effect of SMART attributes on failure predictions, the WLLGM model introduces multiple interaction terms. This involves:

- **Defining Interaction Terms**: The model considers interaction terms across selected SMART attributes to capture the non-linear dependencies and compounding effects that influence hard drive failure. For instance, increased values in Reallocated Sectors (SMART5) and Reported Uncorrectable Errors (SMART187) are compounded, creating a heightened failure risk.

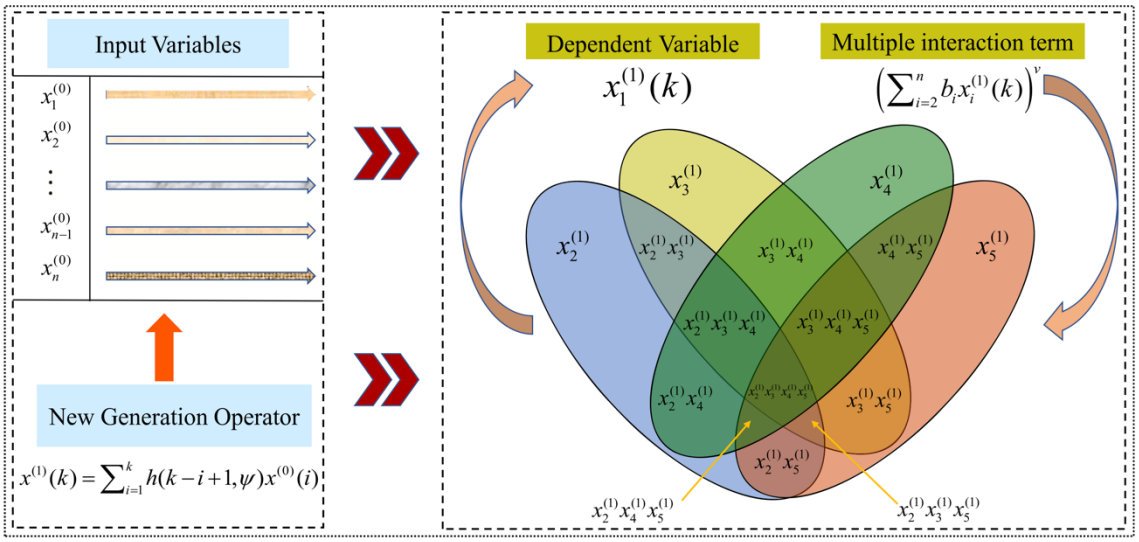


Fig. 2. The multiple interaction mechanism of the WLLGM(1, N) model.

5. Model Evaluation

Performance Metrics: These metrics are essential in assessing how closely the model's predictions align with the actual HDD failure rates. The model's effectiveness is assessed using several evaluation metrics:

- **Mean Absolute Percentage Error (MAPE)**- quantifying prediction accuracy
- **Root Mean Square Error (RMSE)** - assessing residuals' spread
- **Mean Absolute Error (MAE)** - assessing residuals' spread
- **Standard Deviation of Absolute Percentage Error (STD)**.
- **Comparative Analysis**: The WLLGM model's performance is compared with Multiple Linear Regression (MLR), The WLLGM model is found to outperform MLR, particularly in predictive accuracy and robustness.

$$\mathrm{MAPE} = \frac{1}{n}\sum_{s=1}^{n}\frac{\left|\hat{x}_1^{(0)}(s) - x_1^{(0)}(s)\right|}{x_1^{(0)}(s)} \times 100\%,$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{s=1}^{n}\left[\hat{x}_1^{(0)}(s) - x_1^{(0)}(s)\right]^2},$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{s=1}^{n}\left|\hat{x}_1^{(0)}(s) - x_1^{(0)}(s)\right|,$$

$$\mathrm{STD} = \sqrt{\frac{1}{n}\sum_{s=1}^{n}\left(\frac{\left|\hat{x}_1^{(0)}(s) - x_1^{(0)}(s)\right|}{x_1^{(0)}(s)} - \mathrm{MAPE}\right)^2},$$

$$\mathrm{APE} = \frac{\left|\hat{x}_1^{(0)}(s) - x_1^{(0)}(s)\right|}{x_1^{(0)}(s)} \times 100\%.$$

The methodology described above outlines the process of developing and validating a new Weibull Log-Logistic Grey Forecasting Model (WLLGM) to predict HDD failures. By incorporating both Weibull and Log-Logistic distributions, along with sophisticated parameter estimation techniques, the model effectively captures the nonlinear trends and interactions in HDD failure data, significantly improving prediction accuracy and reliability.

Results

Table 1

Experiment results of WLLGM and competition models for hard disk drive failure.

Model - ST4000DM000			
Month	Actual	WLLGM	MLR
Fitting			
February	22	20.1690	24.0485
March	26	25.3026	22.8306
April	20	20.7971	17.2586
May	26	26.1530	30.2511
Predicted			
June	22	21.8419	23.0721

Model - ST14000NM0138			
Month	Actual	WLLGM	MLR
Fitting			
February	112	110.3756	115.5808
March	93	92.3876	98.2647
April	86	88.2275	90.0256
May	89	91.6438	85.2450
Predicted			
June	113	111.5675	118.8821

Model - ST14000NM001G			
Month	Actual	WLLGM	MLR
Fitting			
February	12	12.7933	16.2408
March	13	12.5304	15.2672
April	16	18.3272	14.0256
May	19	20.3961	21.0256
Predicted			
June	13	12.7079	14.0256

Table 1 presents the results obtained by the WLLGM(1, N) model against the competition model MLR, where the actual data represents the total number of failures of hard disk drives per month. it can be found that that for the hard disk drives ST4000DM000 ,ST14000NM0138 ,ST14000NM001G , the fitting and prediction values of WLLGM are closest to the actual values, highlighting better fitting with less deviation.

Table 2

Evaluation Metrics of models for fitting and prediction.

Model - ST4000DM000		
Criterion	WLLGM	MLR
Fitting		
MAPE(%)	0.4474	1.0425
RMSE	0.1422	0.3439
STD%	0.1179	0.3734
MAE	0.082	0.0303
Prediction		
MAPE(%)	0.5567	2.3328
RMSE	0.7689	0.3728
STD%	0.5104	0.4212
MAE	0.6119	0.0312

Model - ST14000NM0138		
Criterion	WLLGM	MLR
Fitting		
MAPE(%)	0.5564	0.8745
RMSE	0.2887	0.3988
STD%	0.2373	0.4284
MAE	0.1958	0.0533
Prediction		
MAPE(%)	0.3425	4.615
RMSE	0.6275	0.4748
STD%	0.3699	0.5005
MAE	0.4308	0.0594

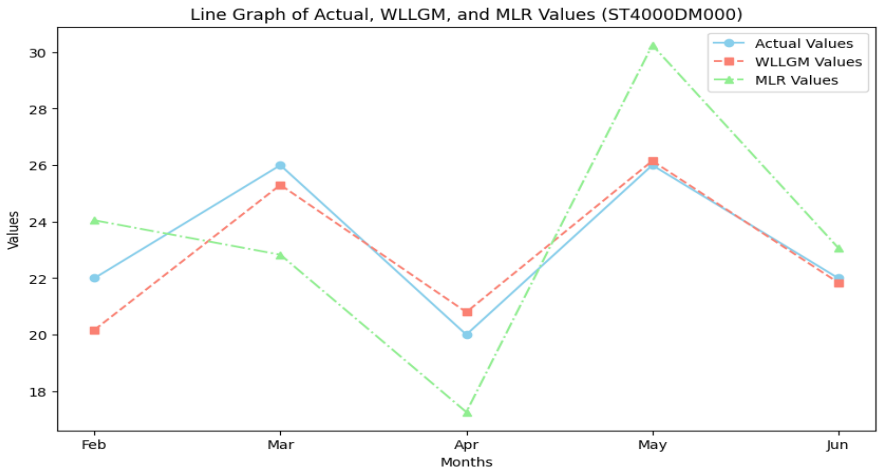
Model - ST14000NM001G		
Criterion	WLLGM	MLR
Fitting		
MAPE(%)	0.2002	1.2036
RMSE	0.1890	0.3624
STD%	0.0580	0.1254
MAE	0.1027	0.2145
Prediction		
MAPE(%)	0.6077	4.1824
RMSE	0.5737	0.3214
STD%	0.3664	0.2823
MAE	0.557	0.7523

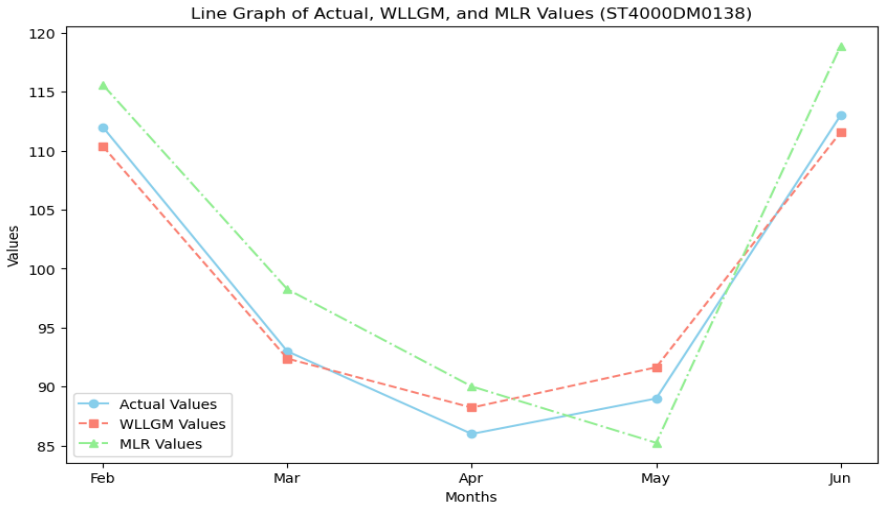
Table 2 presents the result of the model evaluation matrix, we can conclude that, for each dataset, the WLLGM consistently outperforms MLR across metrics:

WLLGM has a significantly lower MAPE, reflecting its accuracy in both fitted and predicted values. It's lower RMSE across datasets shows it more effectively minimizes error magnitude. It also shows lower variability in prediction errors, which enhances its robustness across datasets. WLLGM typically had a lower MAE, indicating that its errors are consistently smaller on average compared to MLR, aligning with WLLGM's focus on minimizing prediction errors.

Hence, the Weibull Log-Logistic Grey Forecasting Model (WLLGM) demonstrates superior accuracy and consistency over Multiple Linear Regression (MLR) for HDD failure prediction. The WLLGM's lower MAPE, RMSE, STD, and MAE values imply it can better adapt to the complex and non-linear interactions in HDD failure data.

The visualization results of each model.





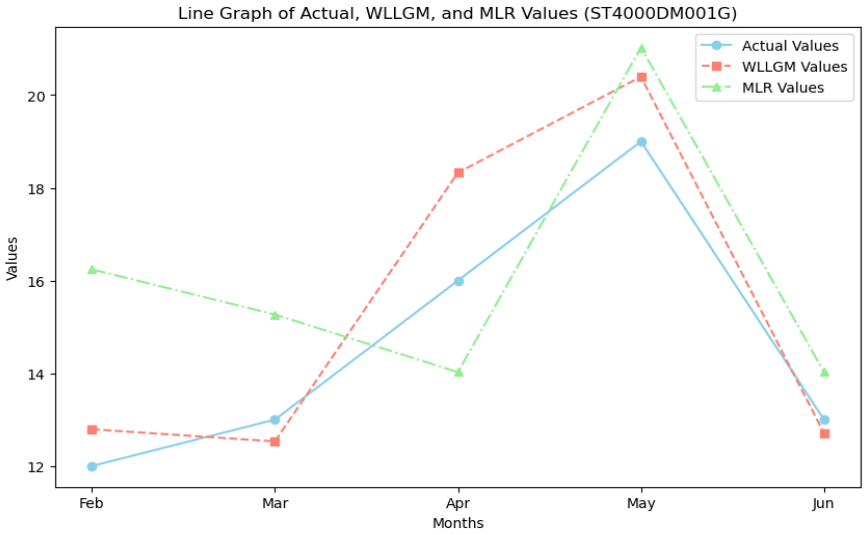


Table 3

The results of parameters of WLLGM.

Parameter	ST4000DM000	ST14000NM0138	ST14000NM001G
v	1.3953904	1.90212157	0.11779333
u	1.20138130	0.86652997	0.64586261
a	-0.00348812	-0.00049662	0.00600237
b2	4.04635767e-01	4.55979758e-01	2.84539493e-01
b3	1.23023154e-04	5.02074157e-06	1.27294058e-05
b4	-1.30001196e-01	3.43576572e-06	-1.30049882e-01
b5	-2.25882886e-06	-3.39541682e-06	2.18662073e-04
b6	-2.25882888e-06	-3.39541674e-06	2.18662073e-04
b7	0.50247033	0.49206755	0.49895195

Table 3 presents the values of the parameters of the proposed model for forecasting the failure of hard disk drives. Compared to ST4000DM000 and ST14000NM001G, which have stronger nonlinear relationships, ST14000NM0138's failure patterns are simpler due to its minimal nonlinearity, as indicated by the nonlinear degree (v) parameter. The consistency of the model is shown by the development coefficient (a), where a negative value suggests high model alignment with observed failure patterns and a positive but minor value represents dependable prediction performance. Lastly, the influence coefficients (bi) show that the most important feature for HDD failures is SMART5 (Reallocated Sectors Count), highlighting the significance of keeping an eye on this particular property.

Conclusions

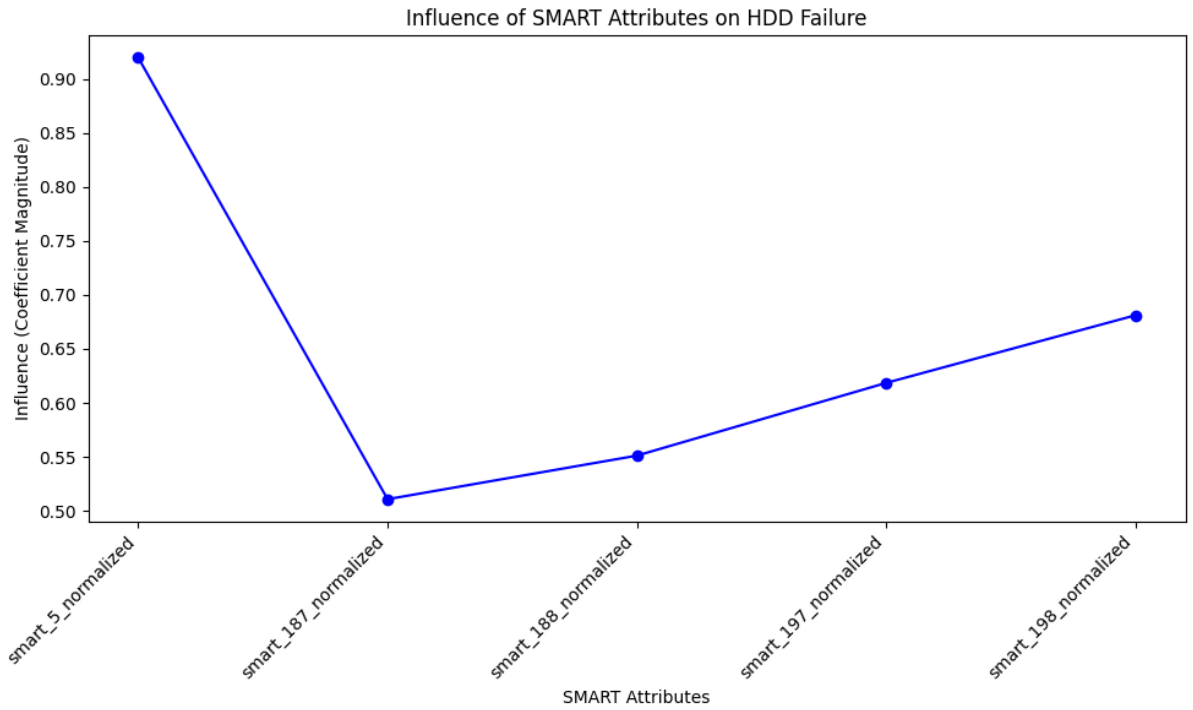
1.A novel Weibull Log-Logistic grey forecasting model with multiple interaction effects is proposed in this paper to accurately and stably predict hard disk drive failure.

2.The model's ability to account for multiple interaction effects between SMART attributes, enhances its predictive power. This multi-factor approach allows for a more comprehensive understanding of the causes behind HDD failures.

3.The new parameter estimation method is effective for the proposed model. The model was validated by taking three different varieties of hard disk drives as datasets, which shows superior prediction performance and robustness compared to the MLR model and can better characterize hard disk drive failure data.

4.The model effectively reduces false positive rates, which is a critical improvement over traditional HDD failure prediction models. This leads to fewer unnecessary HDD replacements, saving resources and costs.

5.Among the five SMART attributes that are most associated with hard disk drive failure, the reported uncorrectable errors, command timeout, current pending sector count, and uncorrectable sector count have the same influence on the failure of hard disk drives. The reallocated sectors count has the greatest impact.



Limitations

1. The model was validated using data from a specific source (BackBlaze) with particular HDD types. The performance of the model may vary when applied to different datasets or HDD brands, limiting its generalizability to all types of hard disk drives.

2. The model focuses on predicting failures based on historical data and certain factors (e.g., SMART attributes). It may not capture other relevant factors, such as manufacturing defects, power fluctuations, or external environmental factors, which can also lead to HDD failures.

3. The model's real-time prediction capability has not been fully explored. Implementing it in systems requiring live, continuous monitoring might pose challenges, such as data processing speed and resource allocation.

4. The reliance on SMART attributes assumes that these indicators are always accurate and sufficient for predicting failures. However, in some cases, SMART data may not capture all failure modes or might fail to provide early warning.

5. The current model is tailored for HDD failure prediction. Its application to other types of data storage technologies, such as solid-state drives (SSDs) or hybrid systems, is not explored, limiting its wider application.

6. While the model shows promising results, further refinement is necessary, particularly in addressing more complex nonlinearities and optimizing the balance between prediction accuracy and computational efficiency.

Scope

1. The current model is designed specifically for HDDs. Future research could extend the model to predict failures in other storage technologies such as solid-state drives (SSDs), hybrid storage systems, or cloud-based storage, where failure mechanisms and data characteristics differ.

2.Exploring the potential of integrating the Weibull Log-Logistic Grey forecasting model with machine learning algorithms could lead to hybrid models that combine statistical methods with advanced predictive analytics. This approach might further enhance prediction accuracy and adaptability to different datasets.

3. Since the Weibull and Log-Logistic distributions are commonly used in reliability analysis, future research could explore applying the proposed model to other mechanical systems with similar failure characteristics, such as turbines, vehicles, or medical devices, broadening its utility beyond HDDs.

References

[1] G.F. Hughes, J.F. Murray, K. Kreutz-Delgado, et al., Improved disk-drive failure warnings, Trans. Reliab. 51 (2002) 350–357, https://doi.org/10.1109/

tr.2002.802886.

[2] E. Pinheiro, W.D. Weber, L.A. Barroso, Usenix, Failure trends in a large disk drive population, in: Proceedings of the 5th USENIX Conference on File and Storage

Technologies, Usenix Assoc, San Jose, CA, 2007, p. 2. https://dl.acm.org/doi/10.5555/1267903.1267905.

[3] S. Sankar, M. Shaw, K. Vaid, et al., Datacenter scale evaluation of the impact of temperature on hard disk drive failures, ACM Trans. Storage 9 (2013) 1–24,

https://doi.org/10.1145/2491472.2491475.

[4] B.D. Strom, S. Lee, G.W. Tyndall, et al., Hard disk drive reliability modeling and failure prediction, Trans. Magn. 43 (2007) 3676–3684, https://doi.org/

10.1109/tmag.2007.902969.

[5] L.P. Queiroz, F.C.M. Rodrigues, J.P.P. Gomes, et al., A fault detection method for hard disk drives based on mixture of Gaussians and nonparametric statistics,

Trans. Ind. Inform. 13 (2017) 542–550, https://doi.org/10.1109/tii.2016.2619180.

[6] R. Chen, X. Xiao, M. Gao, et al., A novel mixed frequency sampling discrete grey model for forecasting hard disk drive failure, ISA Trans. (2024), https://doi.

org/10.1016/j.isatra.2024.02.023.

[7] J. Zhao, Y.Z. He, H.M. Liu, et al., Disk failure early warning based on the characteristics of customized SMART, in: Proceedings of the 19th IEEE Intersociety

Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, IEEE, Electr Network, 2020, pp. 1282–1288, https://doi.org/10.1109/

ITherm45881.2020.9190324.

[8] V. Tomer, V. Sharma, S. Gupta, et al., Hard disk drive failure prediction using SMART attribute, Mater. Today: Proc. 46 (2021) 11258–11262, https://doi.org/

10.1016/j.matpr.2021.03.229.

[9] R. Jiang, D. Murthy, Mixture of Weibull distributions—Parametric characterization of failure rate function, Appl. Stoch. Models Data. Anal. 14 (1998) 47–65,

https://doi.org/10.1002/(SICI)1099-0747(199803)14:1<47::AID-ASM306>3.0.CO;2-E.

Annexure/ Appendix

```
Load the data and split into training and testing sets where X are the independent
smart attributes i.e
smart_198_normalized,smart_197_normalized,smart_188_normalized,smart_187_normal
ized and the dependent variable is smart_5_normalized which affects the failure
rate.

"""

# Importing the required libraries

!pip install openpyxl

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from scipy.signal import convolve


# For Google Colab:

from google.colab import files

uploaded = files.upload()

# Get the filename

filename = list(uploaded.keys())[0]

# Read the Excel file using pandas

data = pd.read_excel(filename)


# Select the SMART attributes as features

X = data[['smart_5_normalized', 'smart_187_normalized', 'smart_188_normalized',

        'smart_197_normalized', 'smart_198_normalized']]

# Using 'smart_5_normalized' as target variable

y = data['smart_5_normalized']

# Split data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Output shapes of the training and test sets

print("Training set shape:", X_train.shape, y_train.shape)

print("Testing set shape:", X_test.shape, y_test.shape)


"""Defining the pdf of each distribution such as Weibull and Log Logistic and then
using it to define the Weibull Log Logistic model and then calculate the WLL
accumulation generation operator."""
```

```
from scipy.signal import convolve

def weibull_component(t, alpha1, beta1):


"""Weibull Probability Density Function (PDF) """


return (beta1 / alpha1) * (t / alpha1)**(beta1 - 1) * np.exp(-(t / alpha1)**beta1)

def log_logistic_component(t, alpha2, beta2):


 """Log-Logistic Probability Density Function (PDF) """


return (beta2 / alpha2) * (t / alpha2)**(beta2 - 1) / ((1 + (t / alpha2)**beta2)**2)

def wll_accumulation_operator_vectorized(data, p=0.3, alpha1=1.5, alpha2=0.5,
beta1=2.0, beta2=2.5):


"""Applies the Weibull Log-Logistic (WLL) accumulation generation operator on a data
sequence without using for loops. """


N = len(data)

t = np.arange(1, N + 1)  # Time steps from 1 to N

# Compute h(t) for all t using vectorized operations

h = p * weibull_component(t, alpha1, beta1) + (1 - p) * log_logistic_component(t,
alpha2, beta2)

# Perform convolution between data and h(t)

x1 = np.convolve(data, h, mode='full')[:N]

        return x1

# Ensure X_train contains only numeric data

X_train_numeric = X_train.apply(pd.to_numeric, errors='coerce').fillna(0)

# Apply WLL accumulation generation operator to each SMART attribute in the
training set

X_train_wll = X_train_numeric.apply(lambda col:
wll_accumulation_operator_vectorized(col.values), axis=0)

# Display the transformed training set

print("Transformed training set with WLL operator applied:\n", X_train_wll.head())


"""To calculate the hazard function i.e the value of failure rate of WLLGM model"""


p=0.3  #mixing proportion
```

```
alpha1=1.5

alpha2=0.5

beta1=2.0

beta2=2.5

N = len(data)

t = np.arange(1, N + 1)

h = p * weibull_component(t, alpha1, beta1) + (1 - p) * log_logistic_component(t,
alpha2, beta2)

mean_h=np.mean(h)

def reliability_function(t, alpha1 , beta1 ,alpha2, beta2 ,p):

    R1= np.exp(-(t / alpha1)**beta1)

    R2=1/ (1 + (t / alpha2)**beta2)

    return p*R1+(1-p)*R2

reliability=reliability_function(t, alpha1 , beta1 ,alpha2, beta2 ,p)

def failure_rate(t):

    f1 = weibull_component(t, alpha1, beta1)

    f2 = log_logistic_component(t, alpha2, beta2)

    return (p * f1 + (1 - p) * f2) / reliability

print(failure_rate(t))

# Calculate x(1)_i sequence using the accumulation operator

def compute_x1_sequence(x0_sequence):

    n = len(x0_sequence)

    x1_sequence = np.zeros(n)

    for k in range(n):

        x1_sequence[k] = sum(failure_rate(k - j + 1) * x0_sequence[j] for j in range(k + 1))

    return x1_sequence

# Generate x(1)_i for each original x(0)_i sequence

x_other_sequences = [compute_x1_sequence(x0) for x0 in x_original_sequences]

"""Using the whale optimization technique to calculate the meta-heuristic paramters
and calculating the best mape i.e the fitted mape for the model."""


import random

# Define the objective function (MAPE)

def calculate_mape(y_true, y_pred):


    """Calculate MAPE between actual and predicted values."""
```

```
    return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

# Function to calculate WLLGM predictions, considering SMART attribute coefficients
def wllgm_predict(X, mu, gamma, v, smart_coefficients):
    return np.dot(X, smart_coefficients) * mu + gamma - v

# Whale Optimization Algorithm
def whale_optimization_algorithm(X_train, y_train, search_space, iterations=20, whales_count=10):
    # Define parameters and initialize whales randomly within the search space
    whales = [np.array([random.uniform(low, high) for low, high in search_space]) for _ in range(whales_count)]
    best_whale = None
    best_mape = float("inf")
 # Initialize random SMART attribute coefficients
 smart_coefficients = np.random.rand(X_train.shape[1])  # One coefficient per SMART attribute
for iteration in range(iterations):
for whale in whales:
mu, gamma, v = whale
y_pred = wllgm_predict(X_train, mu, gamma, v, smart_coefficients)
mape = calculate_mape(y_train, y_pred)

# Update best whale
if mape < best_mape:
best_mape = mape
best_whale = whale
  # Whale position update based on encircling behavior and random exploration
        for i in range(whales_count):
            if random.random() < 0.5:
                whales[i] = best_whale + (random.random() - 0.5) * 2 * best_whale  # encircle
            else:
                whales[i] = np.array([random.uniform(low, high) for low, high in search_space])  # explore
 print(f"Iteration {iteration + 1}/{iterations}, Best MAPE: {best_mape}")
return best_whale, best_mape, smart_coefficients
# define search space for each parameter
```

```
search_space = [(0.01, 1),  # Range for mu
                (0.01, 5),  # Range for gamma
                (0.01, 2)]  # Range for v
# Run WOA to find the optimal parameters
best_params, best_mape, smart_coefficients =
whale_optimization_algorithm(X_train_wll, y_train, search_space)
print("Optimal parameters found by WOA:")
print("Mu:", best_params[0])
print("Gamma:", best_params[1])
print("V:", best_params[2])
model=wllgm_predict(X,best_params[0] ,best_params[1] , best_params[2],
smart_coefficients)
print(np.mean(model))


"""Using the Least Square Support Vector Machine (LS-SVM) to model the non linear
parameters and to calculate the test mape i.e for predicted values."""


from sklearn.svm import SVR
from sklearn.preprocessing import StandardScaler
# Parameters obtained from WOA
mu_opt, gamma_opt, v_opt = best_params
# Define the LS-SVM
scaler = StandardScaler()  # Normalize the data to improve SVM performance
X_train_scaled = scaler.fit_transform(X_train_wll)  # Scale transformed data
# Instantiate and train the Support Vector Regressor (SVR) as LS-SVM substitute
ls_svm = SVR(kernel='linear', C=1.0)  # Linear kernel for linear relationship
ls_svm.fit(X_train_scaled, y_train)
# Output the support vector coefficients (equivalent to the weights bi in the paper)
support_vector_coeffs = ls_svm.coef_
print("Estimated coefficients (b_i) for each feature:", support_vector_coeffs)
# Calculate predictions on the test set for evaluation
X_test_scaled = scaler.transform(X_test)  # Scale the test set with the training
scaler
y_pred = ls_svm.predict(X_test_scaled)
# Evaluate model performance on the test set
test_mape = calculate_mape(y_test, y_pred)
print("Test MAPE:", test_mape)
```

```
"""To compute the WLLGM value of the model"""


x1_sequence = np.array(X_train_wll)

x_other_sequences = [np.array([x1_sequence])]

# Define Z(1)(k) based on x1_sequence

def compute_Z1(k, x1_sequence):

    return 0.5 * (x1_sequence[k-1] + x1_sequence[k])

# Compute the WLLGM(1, N) value

def compute_WLLGM_1_N(k, x1_sequence, x_other_sequences, a, gamma, v, b_i,
b_n_plus_1):

    # Compute the sum term (sum_i=2^n b_i * x(1)_i(k))^v

    sum_term = sum(b_i[i] * x_other_sequences[i][k] for i in range(len(b_i)))

    interaction_term = (sum_term) ** v

# Compute Z(1)(k)

    Z1_k = compute_Z1(k, x1_sequence)

   # Apply Equation (14)

    left_term = x1_sequence[k] - x1_sequence[k-1] + a * Z1_k

    right_term = gamma * interaction_term + b_n_plus_1

# Calculate WLLGM(1, N) at index k

    WLLGM_1_N_value = left_term - right_term

    return WLLGM_1_N_value

k = 2

wllgm_value = compute_WLLGM_1_N(k, x1_sequence, x_other_sequences, a=0.5, gamma,
v, b_i, b_n_plus_1)

print(f"WLLGM(1, N) value at k={k}: {wllgm_value}")


"""Lasso regression is used when the mape is larger and we reduce it by taking the
cross validation terms."""


# Apply the WLL accumulation generation operator to the test set as we did with the
training set

X_test_wll = X_test.apply(lambda col:
wll_accumulation_operator_vectorized(col.values), axis=0)

# Now, scale the transformed test set with the same scaler used for the training set

X_test_scaled = scaler.transform(X_test_wll)

# Re-run LassoCV with the transformed test set
```

```
from sklearn.linear_model import LassoCV
from sklearn.model_selection import cross_val_score
# Apply LassoCV for regularization to minimize overfitting
lasso_model = LassoCV(alphas=np.logspace(-3, 3, 10), cv=5)  # Adjust alpha range if needed
lasso_model.fit(X_train_scaled, y_train)  # Train with scaled WLL-transformed data
# Evaluate with cross-validation
cv_mape_scores_lasso = -cross_val_score(lasso_model, X_train_scaled, y_train, cv=5, scoring='neg_mean_absolute_percentage_error')
print("Cross-validated MAPE with Lasso:", np.mean(cv_mape_scores_lasso))
# Predictions and evaluation on the test set with consistent scaling
y_pred_lasso = lasso_model.predict(X_test_scaled)
test_mape_lasso = calculate_mape(y_test, y_pred_lasso)
print("Test MAPE after regularization with Lasso:", test_mape_lasso)

"""Used to calculate the evaluation metrics in case of lasso regression"""

from sklearn.metrics import mean_squared_error, mean_absolute_error
import numpy as np
# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
# Calculate MAE
mae = mean_absolute_error(y_test, y_pred_lasso)
# Calculate Standard Deviation of Absolute Percentage Error
ape = np.abs((y_test - y_pred_lasso) / y_test) * 100
std_ape = np.std(ape)
# Display metrics
print("Evaluation Metrics on Test Set:")
print(f"MAPE: {test_mape_lasso}%")
print(f"RMSE: {rmse}")
print(f"MAE: {mae}")
print(f"STD of APE: {std_ape}")

"""Used to calculate the evaluation metrics such as rmse,mae,std for the model."""

from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
import numpy as np

# Calculate RMSE

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

# Calculate MAE

mae = mean_absolute_error(y_test, y_pred)

# Calculate Standard Deviation of Absolute Percentage Error

ape = np.abs((y_test - y_pred) / y_test) * 100

std_ape = np.std(ape)

# Display metrics

print("Evaluation Metrics on Test Set:")

print(f"MAPE: {test_mape}%")

print(f"RMSE: {rmse}")

print(f"MAE: {mae}")

print(f"STD of APE: {std_ape}")


"""To determine the smart attribute that affects the failure rate of the model the most by comparing all the smart attributes."""


# Determine the most influential SMART attribute

influence_strength = np.abs(smart_coefficients)  # Absolute values of the coefficients

# Get the index of the most influential SMART attribute

most_influential_index = np.argmax(influence_strength)

# Print the SMART attribute with the greatest influence

print(f"The most influential SMART attribute is SMART attribute {most_influential_index + 1}")

# Display the influence of all SMART attributes

print("SMART attribute influences:", influence_strength)

# Optional: Plot the influence of SMART attributes

import matplotlib.pyplot as plt

smart_attribute_names = ['smart_5_normalized', 'smart_187_normalized', 'smart_188_normalized', 'smart_197_normalized', 'smart_198_normalized']

plt.bar(smart_attribute_names, influence_strength)

plt.xlabel('SMART Attributes')

plt.ylabel('Influence (Coefficient Magnitude)')

plt.title('Influence of SMART Attributes on HDD Failure')

plt.show()
```

```
import pandas as pd

# Select relevant SMART attribute columns

smart_columns = ['smart_5_raw', 'smart_187_raw', 'smart_188_raw', 'smart_197_raw',
'smart_198_raw']

smart_data = filtered_data[smart_columns]

# Apply cumulative sum to each SMART attribute column

accumulated_data = smart_data.cumsum()

# Optionally, visualize to check the trend

import matplotlib.pyplot as plt

accumulated_data.plot(title=f'Accumulated Data for {data}', figsize=(10, 6))

plt.xlabel("Date")

plt.ylabel("Accumulated Values")

plt.show()


"""To compare the WLLGM model with other model we use the below MLR model"""


#Importing the libraries

import pandas as pd

from google.colab import files

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

import numpy as np


"""Here we check for null or missing values and drop if any. Then we set the
independent and dependent variables and split the data into test and train
datasets."""


# Remove rows with missing values if any

data = data.dropna()

# Select SMART attributes as independent variables (based on the research paper)

independent_vars = ['smart_198_normalized', 'smart_188_normalized',
'smart_187_normalized', 'smart_197_normalized']

dependent_var = 'smart_5_normalized'

X = data[independent_vars]

y = data[dependent_var]

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Apply Multiple Linear Regression to establish the linear relation
model = LinearRegression()
model.fit(X_train, y_train)
# Step 7: Retrieve the estimated parameters (coefficients and intercept)
coefficients = model.coef_
intercept = model.intercept_
print("Intercept:", intercept)
print("Coefficients:", coefficients)
# Display the results
print("\nEstimated Parameters (MLR):")
for var, coef in zip(independent_vars, coefficients):
    print(f"{var}: {coef:.4f}")

"""Calculating the evaluation metrics for both fitted and predicted values."""

from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
# Function to calculate APE (Absolute Percentage Error)
def absolute_percentage_error(y_true, y_pred):
    y_true, y_pred = np.array(y_true), np.array(y_pred)
    return np.abs((y_true - y_pred) / y_true) * 100
# Function to calculate RMSE
def root_mean_squared_error(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))
# Function to calculate STD of absolute percentage error
def std_ape(y_true, y_pred):
    ape = absolute_percentage_error(y_true, y_pred)
    return np.std(ape)
# Step 6: Get Fitted and Predicted Values
fitted_values = model.predict(X_train)  # Fitted values for the training set
predicted_values = model.predict(X_test)  # Predicted values for the test set
# Calculate metrics for fitted values (training set)
mape_fitted = np.mean(absolute_percentage_error(y_train, fitted_values))
rmse_fitted = root_mean_squared_error(y_train, fitted_values)
```

```
std_ape_fitted = std_ape(y_train, fitted_values)

mae_fitted = mean_absolute_error(y_train, fitted_values)

# Calculate metrics for predicted values (test set)

mape_predicted = np.mean(absolute_percentage_error(y_test, predicted_values))

rmse_predicted = root_mean_squared_error(y_test, predicted_values)

std_ape_predicted = std_ape(y_test, predicted_values)

mae_predicted = mean_absolute_error(y_test, predicted_values)

# Display results

print("Metrics for Fitted Values (Training Set):")

print(f"MAPE: {mape_fitted:.2f}%")

print(f"RMSE: {rmse_fitted:.2f}")

print(f"STD of APE: {std_ape_fitted:.2f}")

print(f"MAE: {mae_fitted:.2f}")

print("\nMetrics for Predicted Values (Test Set):")

print(f"MAPE: {mape_predicted:.2f}%")

print(f"RMSE: {rmse_predicted:.2f}")

print(f"STD of APE: {std_ape_predicted:.2f}")

print(f"MAE: {mae_predicted:.2f}")
```