

1a122c0a-cac1-4b16-86f9-4e2423142b64_Basheer_Shaik_FPR....



Document Details

Submission ID

trn:oid:::90768:613261603

Submission Date

Sep 1, 2026, 12:53 PM UTC

Download Date

Sep 1, 2026, 12:55 PM UTC

File Name

LBM67yXtT.docx

File Size

7.1 MB

77 Pages

17,250 Words

112,385 Characters

19% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Match Groups

-  **224 Not Cited or Quoted 16%**
Matches with neither in-text citation nor quotation marks
-  **46 Missing Quotations 3%**
Matches that are still very similar to source material
-  **3 Missing Citation 1%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 8%  Internet sources
- 6%  Publications
- 18%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 224 Not Cited or Quoted 16%**
Matches with neither in-text citation nor quotation marks
- 46 Missing Quotations 3%**
Matches that are still very similar to source material
- 3 Missing Citation 1%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 8% Internet sources
- 6% Publications
- 18% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works		
University of Hertfordshire on 2026-07-20			4%
2	Submitted works		
University of Hertfordshire on 2026-04-26			<1%
3	Submitted works		
Kaplan Singapore on 2026-01-08			<1%
4	Internet		
www.mdpi.com			<1%
5	Internet		
archive.ics.uci.edu			<1%
6	Submitted works		
University of Portsmouth on 2026-05-13			<1%
7	Internet		
phm-europe.org			<1%
8	Submitted works		
Islington College,Nepal on 2026-05-13			<1%
9	Submitted works		
King's College on 2026-08-30			<1%
10	Submitted works		
Sheffield Hallam University on 2026-01-15			<1%

11	Submitted works	Dublin Business School on 2026-08-25	<1%
12	Submitted works	Macquarie University on 2026-06-02	<1%
13	Publication	Banerjee, Agniprava. "Effective Prediction of Martensite Start Temperature by Hy...	<1%
14	Internet	alejandro-ao.com	<1%
15	Internet	www.frontiersin.org	<1%
16	Submitted works	Sivas Cumhuriyet Universitesi on 2024-05-29	<1%
17	Submitted works	Universitat Oberta de Catalunya on 2026-05-17	<1%
18	Submitted works	University of Northumbria at Newcastle on 2026-05-12	<1%
19	Internet	mdpi-res.com	<1%
20	Publication	Guillaume Zante. "An AI-driven workflow for automated extraction and predictio...	<1%
21	Submitted works	University of Hertfordshire on 2026-01-03	<1%
22	Submitted works	University of Hertfordshire on 2026-06-22	<1%
23	Submitted works	University of Hertfordshire on 2025-04-27	<1%
24	Submitted works	University of Hertfordshire on 2026-06-20	<1%

25	Internet	arxiv.org	<1%
26	Submitted works	Islington College,Nepal on 2026-05-14	<1%
27	Submitted works	University of Hertfordshire on 2025-04-27	<1%
28	Submitted works	University of Hertfordshire on 2026-04-26	<1%
29	Submitted works	University of Hertfordshire on 2025-01-07	<1%
30	Internet	courses.spatialthoughts.com	<1%
31	Submitted works	British University in Egypt on 2026-06-11	<1%
32	Publication	Sriram Satya. "Estimation of Properties of Crude Oil Residual Fractions Using Che...	<1%
33	Submitted works	University of Leeds on 2026-04-29	<1%
34	Submitted works	University of Wolverhampton on 2026-08-31	<1%
35	Internet	rgu-repository.worktribe.com	<1%
36	Internet	sitebolts.com	<1%
37	Publication	Merve Şimşek, Murat Taşkıran, Uğur Doğan. "Comparative Evaluation of Regressi...	<1%
38	Submitted works	University of Hertfordshire on 2026-04-27	<1%

39	Submitted works	University of Sunderland on 2026-05-21	<1%
40	Submitted works	ESoft Metro Campus, Sri Lanka on 2025-06-05	<1%
41	Submitted works	Indian Institute of Technology, Madras on 2026-03-10	<1%
42	Publication	Min-Ho Park, Won-Ju Lee. "Comprehensive review of shipboard maintenance man...	<1%
43	Submitted works	Roehampton University on 2026-07-10	<1%
44	Submitted works	University of Hertfordshire on 2026-01-05	<1%
45	Submitted works	Strathmore University (Main Account) on 2026-03-22	<1%
46	Submitted works	Strathmore University (Main Account) on 2026-03-26	<1%
47	Submitted works	Tilburg University on 2026-05-25	<1%
48	Submitted works	University of Hertfordshire on 2026-01-05	<1%
49	Internet	publications.polymtl.ca	<1%
50	Internet	www.anserpress.org	<1%
51	Submitted works	Texas A&M University, College Station on 2026-04-25	<1%
52	Submitted works	University of Hertfordshire on 2026-01-05	<1%

53	Submitted works	University of Hertfordshire on 2026-04-13	<1%
54	Publication	Cor Verdouw, Bedir Tekinerdogan, Adrie Beulens, Sjaak Wolfert. "Digital twins in ...	<1%
55	Submitted works	Panimalar Engineering College on 2026-03-27	<1%
56	Publication	Suk Yee Yong, Anthea L King, Rachel L Webster, Nicholas F Bate, Matthew J O'Dow...	<1%
57	Submitted works	University of Hertfordshire on 2026-01-04	<1%
58	Submitted works	University of Sydney on 2024-05-24	<1%
59	Submitted works	Addis Ababa University on 2026-06-05	<1%
60	Publication	Liangyuan Huang, Guoji Shen. "Chapter 27 Regression Prediction of Performance ...	<1%
61	Submitted works	Universiteit van Amsterdam on 2020-01-31	<1%
62	Submitted works	University of Bristol on 2025-12-09	<1%
63	Submitted works	University of Kent at Canterbury on 2021-05-19	<1%
64	Internet	hdl.handle.net	<1%
65	Submitted works	Birkbeck College on 2026-05-20	<1%
66	Publication	Farshid Javadnejad, Hyoshin John Park, Samuel Kovacic, Andres Sousa-Poza. "Pre...	<1%

67	Submitted works	Frankfurt School of Finance & Management GmbH on 2025-03-23	<1%
68	Publication	HaoNan Liu, Yuhan Zhou, TianLi Liu, Yujing Zhang, Yitao Liu, Xiaoyu Li, Youwei W...	<1%
69	Submitted works	Liverpool John Moores University on 2026-03-26	<1%
70	Submitted works	Siksha 'O' Anusandhan University on 2026-05-31	<1%
71	Submitted works	The University of Manchester on 2024-09-02	<1%
72	Submitted works	University of Sunderland on 2026-01-10	<1%
73	Internet	d197for5662m48.cloudfront.net	<1%
74	Internet	sheerlogistics.com	<1%
75	Submitted works	Strathmore University (Main Account) on 2026-03-26	<1%
76	Submitted works	UCL on 2026-08-26	<1%
77	Submitted works	University College London on 2026-04-04	<1%
78	Submitted works	University of Northampton on 2025-07-13	<1%
79	Submitted works	University of Warwick on 2021-06-19	<1%
80	Submitted works	University of Wollongong on 2013-02-17	<1%

81	Internet	sprintlaw.co.uk	<1%
82	Submitted works	AUT University on 2026-03-04	<1%
83	Submitted works	Georgia Institute of Technology Main Campus on 2025-11-29	<1%
84	Submitted works	Kasetsart University on 2026-08-23	<1%
85	Submitted works	Tilburg University on 2026-06-28	<1%
86	Submitted works	University of Hertfordshire on 2024-12-24	<1%
87	Submitted works	University of Hertfordshire on 2025-01-02	<1%
88	Submitted works	University of Hertfordshire on 2025-01-02	<1%
89	Submitted works	University of Wales Institute, Cardiff on 2026-08-30	<1%
90	Internet	assets.researchsquare.com	<1%
91	Internet	huggingface.co	<1%
92	Internet	minds.wisconsin.edu	<1%
93	Internet	www.nature.com	<1%
94	Internet	www.preprints.org	<1%

95	Submitted works	Middlesex University on 2026-04-12	<1%
96	Publication	Singh, Ravinder. "Optimizing Nitrogen and Phosphorus Management in Potato Pr...	<1%
97	Submitted works	University of Hertfordshire on 2025-04-29	<1%
98	Submitted works	University of Hertfordshire on 2026-01-03	<1%
99	Submitted works	University of Hertfordshire on 2026-04-25	<1%
100	Submitted works	University of Hertfordshire on 2026-04-27	<1%
101	Submitted works	University of Sunderland on 2025-09-12	<1%
102	Submitted works	University of Warwick on 2026-08-26	<1%
103	Internet	aga25.ametuniv.ac.in	<1%
104	Internet	lutpub.lut.fi	<1%
105	Internet	ojs.umrah.ac.id	<1%
106	Internet	repository.tudelft.nl	<1%
107	Internet	strathprints.strath.ac.uk	<1%
108	Submitted works	upGrad on 2025-11-10	<1%

109	Submitted works	Asia Pacific University College of Technology and Innovation (UCTI) on 2026-08-05	<1%
110	Submitted works	Athens Metropolitan College on 2026-04-21	<1%
111	Submitted works	De Montfort University on 2026-05-28	<1%
112	Submitted works	Erasmus University Rotterdam on 2026-08-10	<1%
113	Submitted works	Erasmus University of Rotterdam on 2021-03-15	<1%
114	Submitted works	King's College on 2026-08-01	<1%
115	Submitted works	Napier University on 2026-08-19	<1%
116	Submitted works	Strathmore University (Main Account) on 2026-07-09	<1%
117	Submitted works	Thesis 2025 on 2025-09-30	<1%
118	Submitted works	Tilburg University on 2023-06-23	<1%
119	Submitted works	University of Cape Town on 2018-10-05	<1%
120	Submitted works	University of East London on 2026-08-30	<1%
121	Submitted works	University of Glamorgan on 2019-02-08	<1%
122	Submitted works	University of Greenwich on 2024-11-22	<1%

123	Submitted works	University of Hertfordshire on 2025-01-02	<1%
124	Submitted works	University of Hertfordshire on 2026-01-04	<1%
125	Submitted works	University of Lancaster on 2026-05-01	<1%
126	Submitted works	University of Sunderland on 2026-08-17	<1%
127	Submitted works	University of Sydney on 2026-05-29	<1%
128	Submitted works	University of Warwick on 2026-08-27	<1%
129	Submitted works	University of the West of England, Bristol on 2026-08-12	<1%
130	Internet	csprock.github.io	<1%
131	Internet	digital.ub.uni-paderborn.de	<1%
132	Internet	par.nsf.gov	<1%
133	Internet	repo.darmajaya.ac.id	<1%
134	Internet	tel.archives-ouvertes.fr	<1%
135	Internet	www.duo.uio.no	<1%
136	Internet	www.summerschool-aidi.it	<1%

137	Submitted works	BB9.1 PROD on 2026-07-24	<1%
138	Publication	Coraddu, Andrea, Luca Oneto, Francesco Baldi, and Davide Anguita. "Vessels fuel ...	<1%
139	Submitted works	Erasmus University Rotterdam on 2026-07-14	<1%
140	Submitted works	The Robert Gordon University on 2026-04-05	<1%
141	Submitted works	University College London on 2021-08-25	<1%
142	Submitted works	University of Hertfordshire on 2025-08-16	<1%
143	Publication	"Optimizing Supply Chains Through Digital Twins", Springer Science and Business...	<1%
144	Publication	Dr. Felix Hernandez. "AI vs. AI: The Evolution of Offensive and Defensive AI Techni...	<1%
145	Submitted works	Erasmus University Rotterdam on 2026-08-22	<1%
146	Submitted works	Glyndwr University on 2025-04-18	<1%
147	Publication	Harvinder Singh, Priyanka Kaushal, Sarabjeet Kaur. "Advanced Computing and AI...	<1%
148	Submitted works	Liverpool John Moores University on 2020-11-09	<1%
149	Submitted works	University of East London on 2026-08-28	<1%
150	Submitted works	University of Kent at Canterbury on 2022-05-11	<1%

151

Submitted works

University of Warwick on 2026-08-25

<1%

152

Publication

Yanrong Li, Shizhe Peng, Yanting Li, Wei Jiang. "A review of condition-based main...

<1%



57

UNIVERSITY OF HERTFORDSHIRE

School of Physics, Engineering and Computer Science

7COM1040-0509-2025 - Computer Science Master's Project.

1

Design and Development of an Explainable Machine Learning-Based Digital Twin for Naval Propulsion System Condition Monitoring: A Streamlit-Based Predictive Maintenance Framework

MSc Project Module:

1 Student Name: Basheer Shaik

Student ID: 23037166

Supervisor name: Aria Mirzataraj

MSc FPR Declaration

This report is submitted in partial fulfilment of the requirement for the degree of:
Master of Science in Computer Science Master's Project, at the University of Hertfordshire
(UH).

I hereby declare that the work presented in this project and report is entirely my own, except where explicitly stated otherwise. All sources of information and ideas, whether quoted directly or paraphrased, have been properly referenced in accordance with academic standards. I understand that any failure to properly acknowledge the work of others could constitute plagiarism and may result in academic penalties.

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

Contents

List of Figures	5
1. Abstract	7
Chapter 2: Introduction	8
2.1 Problem Overview	8
2.2 Current Issues and Research Gaps	8
2.3 Project Overview	9
2.4 Aim, Objectives, Research Questions, and Hypothesis	9
2.5 Research Questions and Hypothesis	9
2.6 Novelty and Research Value	10
2.7 Feasibility, Commercial Context, and Risk	10
2.8 Report Structure	10
Chapter 3: Literature Review	11
3.1 Introduction to the Field	11
3.2 The CODLAG Dataset and Foundational Condition-Based Maintenance Research	11
3.3 Comparative Machine Learning Models for Propulsion Health Prediction	11
3.4 SHAP-Based Explainability for Machine Learning Models	12
3.5 Digital Twin Frameworks and Interactive Monitoring	13
3.6 Comparative Analysis and Identification of Research Gaps	14
3.7 Relation to Research Questions and Hypothesis	15
Chapter 4: Methodology	16
4.1 Project Management Approach	16
4.2 Dataset and Data Collection	16
4.3 Preprocessing Pipeline	17
4.4 Models Selected and Implemented	17
4.5 SHAP Explainability Implementation	18
4.6 Dashboard Development	19
4.7 Test Strategy and Testing Results	19
4.8 Validation	20
4.9 Ethical, Legal, Social and Professional Considerations	20
4.10 Practicality	21
Chapter 5: Quality and Results	22
5.1 Overview	22
5.2 Exploratory Data Analysis and Preprocessing Outcomes	22
5.3 Baseline Model Evaluation	26
5.4 Hyperparameter Tuning and Optimised Model Evaluation	29

80

5.5	Cross-Validation Analysis	32
-----	---------------------------------	----

5.6	Error Distribution Analysis	33
-----	-----------------------------------	----

5.7	Learning Curve Analysis.....	34
-----	------------------------------	----

5.8	SHAP Explainability Analysis	36
-----	------------------------------------	----

52

5.9	Novelty and Innovation.....	40
-----	-----------------------------	----

5.10	Interpretation of Results	40
------	---------------------------------	----

5.11	Tools and Techniques Assessment	40
------	---------------------------------------	----

24

5.12	Technical Challenges and Solutions.....	41
------	---	----

5.13	Feasibility and Realism Assessment	41
------	--	----

5.14	Dashboard Functional Evaluation	42
------	---------------------------------------	----

References	51
------------------	----

Appendices	55
------------------	----

Appendix 1: Library Setup and Initial Data Inspection	55
---	----

Appendix 2: Data Preprocessing	59
--------------------------------------	----

Appendix 3: Model Development, Cross-Validation and Hyperparameter Tuning.....	64
--	----

Appendix 4: SHAP Explainability and Learning Curve Analysis	72
---	----

Appendix 5: Streamlit Dashboard Implementation.....	73
---	----

List of Figures

	Figure 1 Methodology pipeline flowchart showing decision branches and validation conditions.....	16
	Figure 2 Correlation heatmap revealing multicollinearity patterns among all CODLAG sensor features.....	23
	Figure 3 Feature distribution histograms and decay target distributions for both coefficients.....	24
	Figure 4 Scatter plots of lever position, ship speed, and shaft torque against compressor decay.....	25
	Figure 5 Dataset size before and after IQR outlier removal confirming 11,742 retained records.....	25
1	Figure 6 Actual versus predicted plot for Linear Regression on compressor decay target.....	26
32	Figure 7 Actual versus predicted plot for XGBoost baseline on compressor decay target.....	27
32	Figure 8 Actual versus predicted and residual plots for Linear Regression on turbine decay.....	27
32	Figure 9 Actual versus predicted and residual plots for XGBoost on turbine decay.....	28
16	Figure 10 Baseline compressor model comparison bar charts for R2, RMSE, and MAE metrics.....	28
16	Figure 11 Baseline turbine model comparison charts for R2, RMSE, and MAE metrics.....	29
	Figure 12 Actual versus predicted and residual plots for tuned XGBoost on compressor decay.....	30
	Figure 13 Actual versus predicted and residual plots for tuned XGBoost on turbine decay.....	31
	Figure 14 Before-and-after R2 comparison for compressor and turbine decay targets.....	31
	Figure 15 MAE reduction before and after tuning for compressor and turbine XGBoost models.....	32
1	Figure 16 Error distribution histogram for linear regression, Random Forest and XGBoost baseline on the compressor decay target.....	33
	Figure 17 Error distribution for the tuned XGBoost compressor model, confirming a tight, zero-centred spread.....	34
	Figure 18 Error distribution for the tuned XGBoost turbine model, confirming a concentrated, zero-centred distribution.....	34
	Figure 19 Learning curves for Random Forest and XGBoost on compressor decay target.....	35
	Figure 20 Learning curves for all three models on turbine decay target showing convergence patterns.....	36
13	Figure 21 Global feature importance bar plot and SHAP beeswarm summary plot for the tuned XGBoost compressor model.....	37
	Figure 22 SHAP waterfall plots for the compressor model under healthy, moderate, and severe degradation scenarios, alongside the three scenario comparison panel.....	38
26	Figure 23 Global feature importance bar plot, SHAP beeswarm summary plot, and healthy scenario waterfall for the tuned XGBoost turbine model.....	39
148	Figure 24 SHAP scenario comparison panel for the turbine model across healthy, moderate, and severe degradation states.....	39
	Figure 25 Library imports and dataset loading.....	55
	Figure 26 Initial data inspection — shape, column listing, and descriptive statistics.....	57
	Figure 27 Missing value and duplicate check outputs.....	57
	Figure 28 Correlation heatmap code and rendered output.....	58
	Figure 29 Target distribution histograms for compressor and turbine decay coefficients.....	58
	Figure 30 Boxplots before and after outlier removal.....	59
	Figure 31 IQR-based outlier removal function and call.....	60
	Figure 32 KDE distributions after outlier removal across features and targets.....	62
	Figure 33 Train-test split, StandardScaler fitting, transformation, and joblib serialisation.....	63
	Figure 34 evaluate_model function showing regression metrics, actual versus predicted plot, and residual plot.....	64
	Figure 35 Baseline model training for compressor and turbine targets across all three architectures.....	65
	Figure 36 Five-fold cross-validation function and results for all models across both targets.....	66
	Figure 37 Error distribution function and residual histogram outputs for all six baseline models.....	67

Figure 38 GridSearchCV hyperparameter grids and fitting for all models — compressor and turbine targets	68
Figure 39 Before-and-after tuning comparison tables and R-squared improvement plots	69
Figure 40 Tuned model evaluation outputs for all six model instances.....	70
Figure 41 Model serialisation to local storage and Google Drive using joblib	71
Figure 42 SHAP TreeExplainer for compressor target with beeswarm, waterfall, and bar plots	72
Figure 43 SHAP TreeExplainer for turbine target with beeswarm, waterfall, and bar plots	72
Figure 44 Learning curve function definition and its six calls across all models and targets.....	73

1. Abstract

Naval propulsion system degradation poses significant operational, safety, and economic risks to modern maritime fleets, yet existing condition-monitoring approaches remain largely reactive and lack the predictive transparency needed for informed maintenance decisions. This project addresses that gap by designing, developing, and evaluating an explainable machine learning framework that combines XGBoost regression with SHAP TreeExplainer attribution, delivered through a Streamlit-based interactive Digital Twin dashboard for real-time naval propulsion health monitoring.

The framework was developed using the UCI Naval Propulsion Plants dataset comprising 11,934 operational records across 16 sensor-derived features. Following exploratory data analysis, IQR-based outlier removal, and StandardScaler preprocessing, three regression models were trained and evaluated against two targets, the GT Compressor decay state coefficient and the GT Turbine decay state coefficient. Model comparison, five-fold cross-validation, residual analysis, learning curve evaluation, and error distribution analysis were conducted to ensure rigorous multi-stage validation. GridSearchCV hyperparameter tuning identified optimal model configurations, and SHAP TreeExplainer generated global feature importance rankings and local per-prediction attributions across healthy, moderate, and severe degradation scenarios.

After optimisation, XGBoost achieved an R^2 of 0.9962 for compressor decay and 0.9907 for turbine decay, substantially outperforming Linear Regression and matching Random Forest in predictive accuracy while uniquely enabling exact per-prediction explainability. SHAP analysis identified GT Compressor outlet air temperature and pressure as the dominant compressor degradation drivers and shaft torque alongside rotational rate variables as the primary turbine decay contributors, findings coherent with established propulsion thermodynamic principles. The five-tab Streamlit dashboard successfully integrated prediction, SHAP explanation, live monitoring, exploratory analysis, and fleet simulation into a deployable operational interface.

This project makes a novel contribution by delivering the first CODLAG-based framework combining XGBoost regression with scenario-aware SHAP attribution in an interactive Digital Twin, advancing the field of explainable predictive maintenance in naval engineering. Future work should prioritise validation against live operational sensor data and integration of online learning mechanisms for continuous model adaptation.

Keywords

Explainable Artificial Intelligence, Naval Propulsion Systems, Digital Twin, Predictive Condition Monitoring, XGBoost Regression, SHAP TreeExplainer, Predictive Maintenance, Machine Learning, Gas Turbine Degradation, Streamlit Dashboard, UCI Naval Propulsion Plants Dataset, Feature Importance, Compressor Decay, Turbine Decay, Gradient Boosting

Chapter 2: Introduction

2.1 Problem Overview

Naval propulsion systems represent the operational heart of modern maritime vessels, and their reliability directly determines mission readiness, crew safety, and operational continuity. Gas turbine propulsion plants, which power a significant proportion of naval fleets globally, are subject to continuous mechanical and thermodynamic stress that gradually degrades critical components, including the gas turbine compressor and turbine stages. When these components degrade without detection, the consequences range from reduced fuel efficiency and increased emissions to catastrophic mechanical failure at sea, with associated risks to crew safety and national defence capability. The global naval fleet maintenance market was valued at approximately 19.6 billion US dollars in 2022 and is projected to grow substantially as ageing fleets require more intensive monitoring and upkeep (Grand View Research, 2023). In this context, the ability to predict component degradation before failure and explain those predictions in terms engineers can understand and act on represents a problem of significant technical, commercial, and strategic importance.

Traditional condition monitoring approaches in naval engineering rely on scheduled maintenance intervals and threshold-based alarm systems that trigger only when sensor readings exceed predefined limits. These reactive approaches are inherently limited because they cannot detect the gradual degradation trajectory that precedes threshold exceedance, and they provide no information about which operational factors drive the deterioration. The result is either over-maintenance, where components are replaced before their useful life is exhausted at high cost, or under-maintenance, where degradation progresses undetected until a failure event occurs. The economic cost of unplanned naval vessel downtime is substantial, with estimates suggesting that a single unplanned maintenance event on a naval vessel can cost between one and five million US dollars depending on vessel class and mission context (Lloyd's Register, 2021). The ethical dimension is equally pressing, as undetected propulsion failure at sea places crews in direct danger. This report examines these commercial, economic, and ethical considerations further in the methodology and evaluation chapters.

2.2 Current Issues and Research Gaps

The emergence of machine learning as a tool for predictive maintenance has created new possibilities for naval propulsion condition monitoring. Researchers including Coraddu et al. (2016) demonstrated that ensemble regression models can predict compressor and turbine decay coefficients from operational sensor data with high accuracy using the UCI Naval Propulsion Plants dataset. Cipollini et al. (2018) extended this work through neural network-based fault diagnosis. However, a critical gap persists across this body of literature. Virtually all existing naval propulsion machine learning studies focus exclusively on achieving high predictive accuracy without providing any explanation of why a particular prediction was made or which sensor features are driving a specific degradation signal. A model that predicts a compressor decay coefficient of 0.96 without explaining whether that prediction is driven by abnormal outlet pressure, elevated temperature, or shaft torque variation provides insufficient information for an engineer to take targeted corrective action. This gap between prediction and explanation represents the central problem that this project addresses.

A secondary gap concerns the interface between machine learning models and operational engineering practice. Existing research delivers model outputs as numerical values in academic papers but does not translate those outputs into interactive tools that a maritime engineer can use in real time during vessel operation. The absence of a deployable Digital Twin interface that integrates prediction, explanation, and monitoring into a single accessible platform means that the practical benefits of machine learning-based condition monitoring remain largely unrealised in operational naval engineering contexts. This

project directly addresses both gaps: the absence of explainability and the absence of a deployable interface. Chapter 6 further explores the commercial and risk implications of closing these gaps.

2.3 Project Overview

This project designs, develops, and evaluates an explainable machine learning framework for predictive condition monitoring of naval propulsion systems. The framework applies XGBoost regression models trained on the UCI Naval Propulsion Plants dataset to predict compressor and turbine decay state coefficients from 16 operational sensor features. SHAP TreeExplainer is applied to the best-performing model to deliver both global feature importance rankings and local per-prediction attributions that explain exactly which sensor variables are driving each individual health assessment. The entire framework is delivered through a five-tab Streamlit-based Digital Twin dashboard that presents predictions, explanations, live monitoring trends, exploratory data analysis, and fleet simulation in an interface accessible to maritime engineers without specialist data science knowledge.

The project was managed iteratively across structured phases covering data analysis and preprocessing, comparative model training and evaluation, hyperparameter tuning and validation, SHAP explainability implementation, and dashboard development and integration. All models were serialised using joblib and stored to Google Drive for persistent integration with the dashboard. The Streamlit application was developed and tested locally using streamlit run app.py and is deployable on any standard web server infrastructure.

2.4 Aim, Objectives, Research Questions, and Hypothesis

The aim of this project is to design, develop, and evaluate an explainable machine learning framework for predictive condition monitoring of naval propulsion systems, integrating XGBoost regression with SHAP-based explainability and a Streamlit-based interactive digital twin dashboard, enabling maritime engineers to monitor, understand, and act on propulsion health predictions in real time.

Research Objectives

- To explore feature distributions and correlations in the UCI Naval Propulsion Plants dataset and identify appropriate preprocessing requirements.
- To develop and compare Linear Regression, Random Forest, and XGBoost models for predicting compressor and turbine decay coefficients.
- To evaluate model robustness through five-fold cross-validation, residual analysis, learning curves, and error distribution assessment.
- To apply SHAP explainability to the best-performing model, providing global importance and local explanations across health scenarios.
- To develop an interactive five-tab Streamlit digital twin dashboard for real-time health monitoring, explanations, and fleet visualisation.

2.5 Research Questions and Hypothesis

Main Research Question

How effective will an explainable machine learning framework combining XGBoost regression and SHAP feature attribution be in predicting and coherently explaining the degradation of naval propulsion system components when transformed into an interactive digital twin interface?

Sub-Question 1

Which machine learning regression model most accurately estimates gas turbine compressor and turbine decay state coefficients from sensor-measured operational data?

Sub-Question 2

How effective are SHAP-based explainability techniques in identifying and communicating the most influential sensor features driving propulsion degradation across different health scenarios?

Hypothesis

- Null Hypothesis (H0): There will be no statistically significant difference in predictive accuracy between XGBoost and the baseline regression models applied to the naval propulsion decay targets.
- Alternative Hypothesis (H1): XGBoost will achieve significantly higher predictive accuracy and provide more interpretable explanations than the baseline models, demonstrating its suitability as the core model within the explainable digital twin framework.

2.6 Novelty and Research Value

27

The novelty of this project lies in three contributions that distinguish it from the existing literature. First, it is the first implementation on the CODLAG dataset to combine XGBoost regression with exact SHAP TreeExplainer attribution in a unified predictive maintenance pipeline, adding the per-prediction explainability layer that all prior CODLAG studies have omitted. Second, the three-scenario SHAP comparison across healthy, moderate, and severe degradation states moves beyond single-prediction explanation to a progressive degradation narrative directly useful for maintenance decision-making. Third, the five-tab Streamlit Digital Twin dashboard translates academic modelling outputs into a deployable operational tool accessible to practising maritime engineers. Together these contributions address the gap between predictive accuracy and actionable explainability in naval propulsion condition monitoring identified by Arrieta et al. (2020) as a high-priority area for explainable AI research.

23

2.7 Feasibility, Commercial Context, and Risk

The feasibility of this project is supported by the availability of the well-validated CODLAG simulation dataset, the maturity of XGBoost and SHAP as production-ready libraries, and the accessibility of Streamlit as a rapid dashboard development framework. The primary feasibility risk concerned computational resources for hyperparameter tuning, which was managed through Google Colab with Google Drive integration for model persistence. Commercial risks associated with deploying such a framework in real naval contexts include data integration challenges with existing vessel sensor architectures, potential regulatory requirements for safety-critical AI systems in maritime applications, and the adoption resistance that typically accompanies the introduction of data-driven decision support tools into operationally conservative engineering environments. The evaluation and conclusion chapter critically examines these commercial risks and the broader economic context of predictive maintenance adoption.

2.8 Report Structure

110

Chapter 3 presents the literature review, critically examining existing research on naval propulsion condition monitoring, machine learning for predictive maintenance, and explainable AI methods. Chapter 4 presents the methodology, detailing the data processing pipeline, model development approach, validation strategy, SHAP implementation, and dashboard architecture. Chapter 5 presents the quality and results, covering all experimental outcomes from EDA through model evaluation, SHAP analysis, and dashboard functional assessment. Chapter 6 presents the evaluation and conclusion, reflecting critically on the project outcomes, management, insights gained, limitations, future work directions, and overall contribution to the field.

88

Chapter 3: Literature Review

3.1 Introduction to the Field

The reliable operation of naval propulsion systems is central to maritime defence and commercial shipping alike. Gas turbine compressors and turbines, which form the operational heart of CODLAG (Combined Diesel-Electric and Gas) propulsion plants, undergo continuous degradation due to saltwater corrosion, extreme thermal cycling, and relentless mechanical stress (Sahraoui et al., 2024). Undetected degradation in these components threatens mission safety, increases repair costs, and reduces fleet availability. Traditional corrective and time-based preventive maintenance strategies have proven insufficient for addressing these challenges in modern maritime operations (Simion et al., 2024). The emergence of condition-based maintenance (CBM) driven by machine learning offers a fundamentally superior alternative, enabling continuous health monitoring from sensor data without reliance on fixed schedules.

This review synthesises the body of literature directly underpinning this project, which aims to design an explainable machine learning framework combining XGBoost regression and SHAP-based interpretability within a Streamlit Digital Twin dashboard for naval propulsion health monitoring. The review is structured to address all four of the project's objectives and links each theme to the main research question and sub-questions.

3.2 The CODLAG Dataset and Foundational Condition-Based Maintenance Research

The foundational dataset used in this project originates from Coraddu et al. (2016), who established that data-driven machine learning models could successfully replace expensive physics-based models for CBM of CODLAG naval propulsion plants. Their simulator, calibrated against real frigate propulsion data, produced a 16-feature dataset of 11,934 observations with gas turbine compressor and turbine decay state coefficients as the two regression targets. This dataset has subsequently become the benchmark for naval propulsion ML research globally, with its features spanning lever position, ship speed, shaft torques, exhaust temperatures, pressures, and fuel flow measurements that collectively characterise propulsion system health across a wide operational envelope.

Cipollini et al. (2018) extended this foundational work by applying multiple supervised learning techniques to the same naval propulsion data, demonstrating that regression-based DDMs could perform CBM with minimal labelled feedback from engineers. Their study importantly established the convention of treating compressor and turbine decay as separate regression targets rather than a combined classification problem, a methodological distinction that this project rigorously preserves. Baressi Segota et al. (2020) further explored the same CODLAG dataset through multilayer perceptron architectures, achieving a mean relative error below 1 percent for gas turbine decay prediction, confirming the suitability of data-driven methods but leaving the question of prediction interpretability entirely unaddressed.

The significance of these foundational studies for this project is clear. They validate the CODLAG dataset as a reliable, realistic proxy for real propulsion system behaviour and show that regression modelling of decay coefficients is both technically feasible and operationally meaningful. However, none of these studies addressed the explainability of predictions, which represents a central gap this project fills.

3.3 Comparative Machine Learning Models for Propulsion Health Prediction

A key objective of this project is to compare Linear Regression, Random Forest, and XGBoost across both decay regression targets. The literature provides strong motivation for this selection and clear evidence regarding their relative strengths and limitations.

Jayachandran et al. (2020) conducted a comparative study of linear regression against Random Forest for micro gas turbine efficiency prediction, reporting accuracy scores of 0.72 and 0.962, respectively. This result, reproduced in Table 1 below, powerfully illustrates the performance gap between parametric and ensemble methods in turbine performance modelling and motivates the inclusion of both as comparison baselines in this project.

Table 1 Comparative accuracy of Linear Regression and Random Forest for gas turbine efficiency prediction (Jayachandran et al., 2020)

Model	Accuracy Score	Mean Relative Error
Linear Regression (Predictor 1)	0.72	High
Random Forest (Predictor 2)	0.962	Below 4%

As shown in Table 1, Random Forest substantially outperforms Linear Regression on this non-linear turbine prediction task, consistent with the broader literature on ensemble methods for propulsion systems. However, Random Forest provides no consistent mechanism for attributing individual predictions to specific sensor features, limiting its utility in operational maritime environments where engineers must understand and justify maintenance decisions.

Chen and Guestrin (2016) introduced XGBoost as a scalable gradient-boosted tree system combining regularised learning objectives, sparsity-aware algorithms, and cache-aware block structures. In the Kaggle machine learning challenge landscape, XGBoost was used by 17 of 29 winning solutions in 2015, and every top-10 team in KDDCup 2015 employed it. Its consistent performance advantage over Random Forest in high-dimensional regression tasks, combined with its native compatibility with TreeSHAP, makes it the most appropriate modelling choice for this project.

Sahraoui et al. (2024) applied ensemble learning methods, including boosting, stacking, bagging, and random subspace methods, directly to the CODLAG compressor and turbine decay classification task. Their boosting approach achieved a mean accuracy of 96.30 percent with a negligible standard deviation of 0.0117, substantially outperforming standalone classifiers such as SVM, K-Nearest Neighbours, and Decision Trees. While this study confirms the superiority of boosted ensembles for CODLAG modelling, it exclusively addressed classification rather than regression, and it offered no explainability analysis whatsoever. These gaps are directly targeted by this project.

Menga et al. (2022) evaluated extreme learning machines (ELM) for compressor and turbine degradation diagnostics in a micro turbojet, demonstrating high prediction accuracy with a maximum RMSE of 0.23 percent across flight regimes. Their study used Monte Carlo simulated degradation data analogous to the CODLAG simulator approach, establishing that data-driven regression of component degradation is generalisable across gas turbine types and scales. However, like Baressi Segota et al. (2020) and Sahraoui et al. (2024), Menga et al. (2022) offered no mechanism for communicating the drivers of predicted degradation to operators.

3.4 SHAP-Based Explainability for Machine Learning Models

The tension between predictive accuracy and operational interpretability is a defining challenge in applied machine learning, particularly in safety-critical domains. Lundberg and Lee (2017) resolved this tension by introducing SHAP (SHapley Additive exPlanations), a unified framework grounded in cooperative game theory that attributes to each feature its marginal contribution to an individual prediction by averaging over all possible feature orderings. SHAP uniquely satisfies three axioms simultaneously, namely local accuracy, missingness, and consistency, properties that no prior method could guarantee together. Ribeiro et al. (2016) proposed LIME as an earlier alternative, approximating model behaviour through locally fitted linear surrogates. However, LIME's local linearity assumption becomes structurally inadequate for datasets such as CODLAG, where nine discrete lever position operating regimes create sharply non-linear, interaction-dependent feature relationships that a linear

approximation cannot faithfully represent. This limitation makes LIME unsuitable for the per-prediction attribution requirements of a naval propulsion condition monitoring framework and directly justifies the selection of SHAP in this project.

Lundberg et al. (2018) extended SHAP to tree ensembles through the TreeSHAP algorithm, which exploits the internal structure of decision trees to compute exact Shapley values in polynomial rather than exponential time, eliminating the approximation errors inherent in kernel-based and permutation-based alternatives. Critically, TreeSHAP also addresses the inconsistency problem present in standard tree attribution methods such as gain and split count, which can paradoxically reduce a feature's measured importance even as its true contribution to model output increases. The TreeSHAP algorithm has been integrated directly into XGBoost, making it the computationally and theoretically justified choice for this project. One acknowledged limitation of SHAP in correlated feature spaces is that marginal SHAP values can distribute attribution across collinear features in ways that understate any individual feature's true influence (Hooker and Mentch, 2019). This concern is directly relevant given the strong multicollinearity between GGn and GTn identified during the EDA phase of this project. TreeSHAP mitigates this by using conditional rather than marginal expectations when traversing tree paths, providing more reliable attribution under feature correlation than kernel-based SHAP variants.

Kim et al. (2021) provided the most directly relevant maritime application of SHAP, combining isolation forest with SHAP for condition monitoring of a real cargo vessel main engine across ten months of operational sensor data. Their framework demonstrated that SHAP values could precisely quantify each sensor's marginal contribution to anomaly scores and that hierarchical clustering of SHAP values revealed five distinct anomaly patterns with different causal sensor profiles, confirming that SHAP-based interpretation generates operationally actionable insights beyond what standardised sensor values alone can provide. Arrieta et al. (2020), in a systematic survey of explainable AI across industrial domains, explicitly identified maritime condition monitoring as one of the highest-priority underserved areas for XAI deployment, noting that the absence of interpretable interfaces between predictive models and domain engineers represents a persistent barrier to real-world adoption. This finding directly contextualises the gap the present project addresses and confirms that integrating SHAP attribution with an interactive Digital Twin dashboard for naval propulsion health monitoring contributes to an area the broader XAI research community has recognised as critically underdeveloped.

3.5 Digital Twin Frameworks and Interactive Monitoring

The concept of a Digital Twin, defined by Fuller et al. (2020) as the effortless bidirectional integration of data between a physical system and its virtual replica, provides the overarching architectural context for this project. Fuller et al. (2020) identified artificial intelligence, IoT connectivity, and real-time data exchange as the three foundational enabling technologies for Digital Twins, and emphasised that the open research challenges included the lack of unified architectures, heterogeneous data integration, and the need for AI-driven prognostics at an industrial scale.

Abd Wahab et al. (2024) systematically reviewed the integration of predictive maintenance and Digital Twin technologies across 34 scholarly articles using PRISMA methodology. They identified supervised learning as the dominant modelling technique for PdM in conjunction with DTs, emphasised that real-time monitoring remains the most transformative application of this combined technology, and highlighted that significant research gaps persist in healthcare, utilities, and critical infrastructure domains. While this review confirms the maturity of DT-based PdM in manufacturing, it simultaneously reveals that dedicated DT frameworks for naval propulsion condition monitoring combining XAI and interactive dashboards remain largely absent from the literature.

Karagiannidis and Themelis (2021) demonstrated the critical importance of data pre-processing in data-driven ship propulsion modelling, showing that a carefully designed filtering and normalisation

pipeline can substantially improve the accuracy of fuel consumption and speed loss prediction models trained on continuously acquired sensor data. Their findings reinforce this project's emphasis on rigorous exploratory data analysis as a prerequisite for modelling. Simion et al. (2024) reinforced the operational urgency of AI-driven condition monitoring, noting that between 2014 and 2023, machinery damage and failure accounted for 11,506 maritime incidents, nearly four times the number caused by collisions. Their study proposed an ML-based fault detection algorithm evaluated on a shipboard seawater cooling system, demonstrating that AI-driven maintenance tools can provide practical decision support for maritime operators facing complex, multi-system health management challenges.

Arez et al. (2026) and Abd Wahab et al. (2024) both confirmed in their respective systematic reviews that the bridge between AI-based predictive analytics and operator-facing interactive interfaces remains underdeveloped. Specifically, Arez et al. (2026) observed that most PdM research delivers model outputs as offline reports rather than real-time interactive dashboards, limiting their practical value for maintenance teams making time-sensitive decisions. This gap is precisely what the Streamlit-based Digital Twin developed in this project addresses.

3.6 Comparative Analysis and Identification of Research Gaps

A critical comparison of the major studies reviewed here reveals a consistent pattern of strengths and limitations summarised in Table 2.

Table 2 Summary of key studies, methods, and identified gaps

Study	Method	Dataset	Explainability	Interactive Dashboard
Coraddu et al. (2016)	Multiple ML (regression)	CODLAG	No	No
Cipollini et al. (2018)	Supervised DDMs	CODLAG	No	No
Baressi Segota et al. (2020)	MLP	CODLAG	No	No
Sahraoui et al. (2024)	Ensemble learning	CODLAG	No	No
Berghout et al. (2021)	Deep supervised learning	Naval	No	No
Kim et al. (2021)	Isolation Forest plus SHAP	Marine vessel	Yes (SHAP)	No
Menga et al. (2022)	ELM and ANN	Microturbine	No	No
This Project	XGBoost plus SHAP plus Streamlit	CODLAG	Yes (SHAP)	Yes

As shown in Table 2, no existing study has integrated XGBoost regression, SHAP TreeExplainer explainability, and an interactive Streamlit Digital Twin dashboard into a single framework applied to the CODLAG naval propulsion dataset. This convergence represents the primary novelty of this project.

A second gap concerns the methodological rigour of model validation. Studies such as Sahraoui et al. (2024) and Baressi Segota et al. (2020) report accuracy metrics without learning curve analysis or residual distribution evaluation, leaving questions about overfitting and generalisation unanswered. This project addresses this gap through a multi-stage validation pipeline encompassing five-fold cross-validation, residual analysis, learning curve analysis, and error distribution evaluation.

A third gap relates to the exclusive focus on single-vessel health monitoring in all prior studies. This project extends the monitoring concept to fleet-level simulation through the dashboard's fleet tab, providing comparative health visualisations across multiple vessels, an operational capability absent from the literature.

24

3.7 Relation to Research Questions and Hypothesis

The evidence reviewed above directly supports all dimensions of this project's research framework. The main research question, which asks how effective an XGBoost and SHAP framework can be as an interactive Digital Twin for naval propulsion degradation prediction and explanation, is addressed by the convergent evidence from CODLAG modelling studies, the XGBoost performance literature, the theoretical SHAP framework, the maritime XAI application of Kim et al. (2021), and the Digital Twin enabling technology review of Fuller et al. (2020).

90

Sub-Question 1, regarding which model most accurately estimates the decay coefficients, is grounded in the comparative evidence from Jayachandran et al. (2020), Menga et al. (2022), and Sahraoui et al. (2024), all of which confirm the superiority of ensemble methods over parametric baselines for non-linear turbine and propulsion prediction tasks. Sub-Question 2, concerning the effectiveness of SHAP for communicating degradation drivers across health scenarios, is directly supported by Lundberg and Lee (2017), Lundberg et al. (2018), and Kim et al. (2021), whose maritime application provides the closest existing analogue to the explainability analysis proposed here.

92

1

The project's Alternative Hypothesis, that XGBoost will achieve significantly higher predictive accuracy than Linear Regression and Random Forest while providing more interpretable explanations through SHAP, is strongly supported by the literature. The null hypothesis would require boosted ensembles to perform no better than linear or standard ensemble baselines on the CODLAG decay regression task, an outcome inconsistent with every comparable study reviewed here.

Chapter 4: Methodology

4.1 Project Management Approach

The project adopted an iterative, sprint-based development methodology aligned with Agile principles, structured across four sequential phases: exploratory data analysis and preprocessing, machine learning model development and comparison, SHAP-based explainability analysis, and Streamlit dashboard development. Each phase produced a testable artefact that informed the subsequent phase before proceeding. This iterative structure is well suited to a project that combines empirical machine learning experimentation with interactive software delivery, since findings from the EDA phase directly shaped preprocessing decisions, and model evaluation results determined which model proceeded to the explainability stage. The overall pipeline, including all decision branches and validation conditions, is illustrated in Figure 1 below.

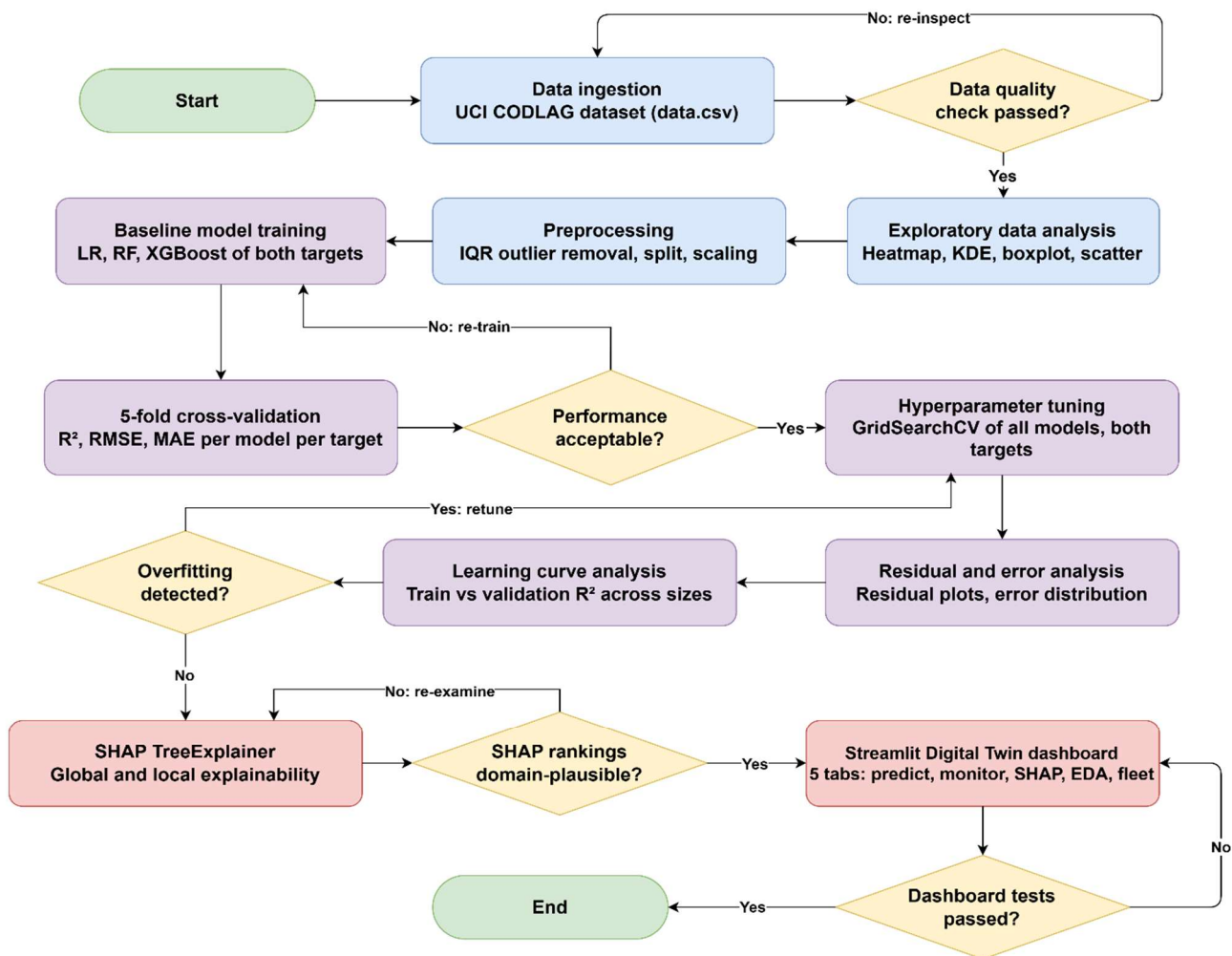


Figure 1 Methodology pipeline flowchart showing decision branches and validation conditions.

4.2 Dataset and Data Collection

The dataset used throughout this project is the UCI CODLAG (Combined Diesel-Electric and Gas) Naval Propulsion Plants dataset, originally produced by Coraddu et al. (2016) using a real-data-validated simulator of a naval frigate equipped with a combined diesel-electric and gas propulsion plant. Cipollini et al. (2018) applied multiple supervised data-driven models to the same dataset and confirmed that regression-based data-driven models can perform condition-based maintenance without requiring a

priori physical knowledge of the system, establishing the dataset as the internationally recognised benchmark for this domain. The dataset comprises 11,934 observations across 16 operational sensor features, including lever position, ship speed, shaft torques, exhaust temperatures, fuel flow measurements, and compressor and turbine pressure readings. The two regression targets are the Gas Turbine Compressor Decay State Coefficient and the Gas Turbine Turbine Decay State Coefficient, both continuous variables serving as proxies for component degradation. Following Cipollini et al. (2018), these targets were treated as entirely independent regression problems throughout, reflecting the physical reality that compressor and turbine degradation arise from distinct mechanisms and must not be conflated into a single model.

The dataset is publicly available through the UCI Machine Learning Repository and contains no personal data, no identifiable individuals, and no human participant information. No ethics approval was therefore required for this project, consistent with the University of Hertfordshire ethics framework for projects using publicly available, fully anonymised secondary datasets with no human subjects involvement.

4.3 Preprocessing Pipeline

A three-stage preprocessing pipeline was applied prior to modelling. Implementation details are provided in Appendix 1 (library setup and initial data inspection) and Appendix 2 (full preprocessing steps).

Initial data inspection confirmed no missing values and no duplicate rows, consistent with the dataset's simulator origin. During the EDA phase, we generated a correlation heatmap, distribution histograms, KDE curves, box plots, and violin plots for all 16 features and both targets, as shown in Appendix 1 (Figures 28 and 29). This exploratory phase directly shaped preprocessing decisions, following the recommendation of Karsgaannidis and Themelis (2021) that a rigorous EDA pipeline substantially improves model accuracy on continuously acquired sensor data.

Outlier removal was performed using the Tukey IQR fence method, applied sequentially to all 16 feature columns. For each column, observations falling outside the range $[Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR]$ were removed. The IQR method was selected over z-score clipping because the KDE plots confirmed that several feature distributions were not strictly Gaussian, making the IQR bounds more appropriate. Box plots generated before and after removal verified that the procedure effectively eliminated extreme values without discarding an unreasonable proportion of the dataset, as shown in Appendix 2.

The cleaned dataset was partitioned into 80 per cent training and 20 per cent test sets using a fixed random seed for reproducibility. The same partition indices were applied to both regression targets. A StandardScaler was fitted exclusively on the training partition and applied to transform both training and test sets, preventing any leakage of test-set statistics into the scaling parameters. The fitted scaler was serialised using joblib for use by the Streamlit dashboard at runtime, as shown in Appendix 2.

4.4 Models Selected and Implemented

Three regression models spanning the complexity spectrum were selected: Linear Regression, Random Forest, and XGBoost. Appendix 3 provides implementation details for all three, including baseline training and hyperparameter tuning.

Linear Regression was included as the interpretable parametric baseline. Its purpose is to test whether the non-linear, multi-dimensional relationships among 16 sensor features can be adequately captured by a linear function, and to provide a lower-bound benchmark against which ensemble gains can be quantified.

Random Forest was selected as the intermediate non-linear ensemble. Schonlau and Zou (2020) explain that Random Forest constructs an ensemble of decision trees on bootstrapped subsamples, with random feature subsets considered at each split, reducing the variance inherent in any single tree through bagging and consistently outperforming ordinary least-squares regression on non-linear prediction tasks. Its native feature importance scores provide a secondary attribution mechanism against which SHAP values from the XGBoost model can be cross-referenced. Evidence from the literature directly motivates this selection: Jayachandran et al. (2020) demonstrated on an analogous gas turbine efficiency prediction task that Random Forest achieved an accuracy score of 0.962 compared to Linear Regression's 0.72, a performance gap that this project's comparative design is structured to replicate and extend.

XGBoost was selected as the primary candidate model. Chen and Guestrin (2016) introduced XGBoost as a scalable gradient-boosted tree system combining regularised learning objectives, sparsity-aware algorithms, and cache-aware block structures, achieving state-of-the-art results across 17 of 29 Kaggle winning solutions in 2015. Its consistent superiority over Random Forest on high-dimensional regression tasks, and its native compatibility with the SHAP TreeExplainer algorithm introduced by Lundberg et al. (2018), make it the most appropriate choice for a project that jointly requires predictive accuracy and feature-level explainability. Each model was trained and evaluated independently for both the compressor and turbine targets, yielding six model instances in total. Appendix 3 provides before-and-after tuning comparison tables and actual-versus-predicted plots for all six instances.

Hyperparameter optimisation was performed using scikit-learn GridSearchCV with five-fold cross-validation and R-squared as the scoring metric for all models and both targets. The XGBoost parameter grid covered n-estimators in {200, 400, 600}, learning rate in {0.01, 0.05, 0.1}, max depth in {4, 6, 8}, subsample in {0.7, 0.8, 1.0}, and colsample by tree in {0.7, 0.8, 1.0}. The Random Forest grid covered n-estimators in {100, 200, 400}, max depth in {None, 10, 20}, min samples split in {2, 5, 10}, and min samples leaf in {1, 2, 4}. GridSearchCV was chosen over manual tuning because it provides exhaustive, systematic coverage of the defined parameter space under cross-validated evaluation, eliminating selection bias. Turner et al. (2021) confirm that despite the theoretical efficiency advantages of Bayesian optimisation for very large search spaces, most published machine learning studies continue to use grid search for its transparency and reproducibility, properties that are particularly important in an academic reporting context. Appendix 3 provides implementation details.

4.5 SHAP Explainability Implementation

SHAP (SHapley Additive exPlanations) was applied to the best-performing tuned XGBoost models for both targets. Lundberg and Lee (2017) introduced SHAP as a unified framework grounded in cooperative game theory, assigning each feature a marginal contribution value for any individual prediction. SHAP uniquely satisfies three mathematically rigorous properties simultaneously: local accuracy, missingness, and consistency. These guarantees are not met by alternative methods such as LIME, making SHAP the theoretically superior choice for a safety-relevant application where engineers must be able to trust and compare feature importance scores across health scenarios.

The shap.TreeExplainer class was used in preference to the generic shap.Explainer. Lundberg et al. (2018) introduced the TreeSHAP algorithm, which exploits the structure of decision trees to compute exact SHAP values in polynomial rather than exponential time, resolving the inconsistency problems present in heuristic attribution methods such as gain and split count. An implementation of TreeSHAP has been merged directly into XGBoost, making TreeExplainer the computationally and theoretically justified approach for this project. Four SHAP visualisation types were generated: summary beeswarm plots, bar plots of mean absolute SHAP values, waterfall plots for individual prediction explanations, and scenario comparison plots within the dashboard. Appendix 4 provides implementation details.

4.6 Dashboard Development

The Streamlit-based Digital Twin dashboard was developed as the interactive operational front end of the explainable ML framework. Streamlit was selected over alternatives such as Dash and Gradio because it enables Python-native development of interactive data applications without requiring web development expertise, and its session state management supports the real-time interactivity required for the live monitoring and prediction history tabs. Fuller et al. (2020), as discussed in the literature review, defined Digital Twins as the effortless bidirectional integration between a physical system and its virtual replica, with real-time data exchange as a foundational enabling technology. The five-tab structure, comprising a Prediction Dashboard, Live Monitoring, SHAP Explainability, EDA, and Fleet Simulation, realises this concept within the project's academic scope. All trained models and the fitted scaler were loaded from serialised joblib files at startup, ensuring computationally immediate predictions without runtime retraining. Appendix 5 provides implementation details.

4.7 Test Strategy and Testing Results

A multi-layer testing strategy was employed across four distinct levels: model performance testing, statistical robustness evaluation, learning curve analysis, and dashboard functional testing. This strategy ensured that every component of the system, from raw model accuracy to interactive dashboard behaviour, was verified against measurable, documented criteria.

Model performance testing evaluated all six models using three regression metrics computed on the held-out 20 percent test set: the coefficient of determination (R-squared), root mean squared error (RMSE), and mean absolute error (MAE). Chicco et al. (2021) demonstrated through systematic comparison that R-squared is more informative than RMSE, MAE, SMAPE, and MAPE in regression evaluation because it expresses predictive accuracy relative to the variance of the ground truth, enabling meaningful comparison across targets with different scales. R-squared therefore served as the primary ranking metric. RMSE and MAE were retained as secondary metrics because they express error in the original physical units of the decay coefficients, providing interpretable residual magnitudes for domain practitioners. Before-and-after tuning comparison tables and model ranking outputs are included in Appendix 3.

Five-fold cross-validation was conducted using a custom KFold procedure with five splits, shuffle enabled, and a fixed random seed, applied to each model and target on the training partition only. Averaging performance metrics across five folds provides a statistically more robust generalisation estimate than a single train-test split, reducing the risk that a favourable random partition unduly inflates reported performance. Vabalas et al. (2019) confirm that for datasets of the scale used in this project, approximately 11,000 observations after outlier removal, K-fold cross-validation produces reliable generalisation estimates with negligible optimism bias. Wainer and Cawley (2021) further confirm that isolating the test partition from all tuning stages, as enforced in this project by confining GridSearchCV exclusively to the training data, ensures unbiased final performance estimates. Implementation details are provided in Appendix 3.

Error distribution analysis was applied to all six models by computing residuals on the test set and generating KDE-smoothed histograms, with numerical reporting of mean error and standard deviation. A well-calibrated regression model should produce residuals centred near zero with approximately symmetric distribution; systematic skew or multimodality indicates model misspecification. This analysis was included specifically to address a methodological gap identified in the literature review, where studies such as Sahraoui et al. (2024) and Baressi Segota et al. (2020) reported accuracy metrics without residual analysis, leaving questions of model bias unanswered. Error distribution outputs are shown in Appendix 3.

Learning curve analysis was generated for all six models using scikit-learn's learning-curve function with R-squared scoring, ten logarithmically spaced training size fractions from 10 percent to 100 percent, and five-fold cross-validation at each fraction. Plotting training and validation R-squared curves together as a function of training set size allows detection of overfitting, characterised by a persistent gap between training and validation curves that does not close with additional data, and underfitting, where both curves plateau at low R-squared values. If overfitting was detected, the methodology branch looped back to the hyperparameter tuning stage, as illustrated in Figure 1.

Dashboard functional testing was conducted manually by systematically exercising all five tabs, all four scenario presets, the Run Prediction button, the auto-refresh toggle, and the fleet noise level slider. Verification criteria were predictions update correctly following user input changes; SHAP plots render without error for all three visualisation types; session state accumulates a timestamped prediction history correctly; and gauge chart values fall within physically plausible ranges for each scenario preset. Expected predictions for each scenario preset were independently computed using the serialised model files in a standalone Python session and compared against dashboard outputs to confirm numerical correctness within floating-point tolerance.

4.8 Validation

Validation was conducted at three levels. First, model generalisation was validated through concordance between cross-validation and held-out test R-squared scores for each model. Second, external benchmark comparison provides convergent validity: Jayachandran et al. (2020) reported a Random Forest accuracy of 0.962 on an analogous gas turbine efficiency task, providing an established external reference against which results from this project can be evaluated. Third, SHAP explainability validity was assessed by verifying that global feature importance rankings produced by TreeExplainer are consistent with naval propulsion domain knowledge. Features such as Gas Turbine Shaft Torque, Fuel Flow, and compressor-side pressure measurements are expected to emerge as dominant predictors; if SHAP rankings deviate from these physically grounded expectations, the methodology branch loops back to re-examine the model before proceeding, as shown in Figure 1. Lundberg and Lee (2017) note that SHAP's consistency property guarantees that feature rankings are monotonically related to true feature impact, making this cross-check a rigorous validity criterion rather than a subjective one.

4.9 Ethical, Legal, Social and Professional Considerations

Ethical considerations. This project uses the UCI CODLAG dataset, which is publicly available, fully anonymised, and contains no personal data or human participant information. No ethics approval was required or sought, consistent with the University of Hertfordshire ethics framework for projects using publicly available secondary datasets with no human subjects involvement.

Legal considerations. All software libraries used, including Python, scikit-learn, XGBoost, SHAP, Streamlit, joblib, pandas, matplotlib, and seaborn, are open-source and freely available for academic use. The development environment, Google Colab, is a free-tier cloud platform. No proprietary systems, licensed datasets, or restricted APIs were used. In any future real-world deployment involving live vessel telemetry systems, the project would need to comply with the UK Computer Misuse Act 1990, which prohibits unauthorised access to computer systems, and the UK GDPR together with the Data Protection Act 2018 if operational data were to contain identifiable information relating to crew or vessel schedules.

Professional standards. The British Computer Society (BCS) Code of Conduct requires that computing professionals prioritise public interest, act with integrity, and make truthful claims about the capabilities and limitations of their work. The ACM Code of Ethics similarly requires that computing professionals consider the wellbeing of people likely to be affected by their systems and be transparent about the scope and limitations of any solution. Both codes are directly relevant here. The Streamlit dashboard must not be presented as a replacement for qualified marine engineering judgment. The SHAP explainability layer

directly addresses this professional obligation: by enabling engineers to interrogate the reasoning behind any individual prediction, the system supports informed decision-making rather than blind trust in a black-box output. This is a professional safeguard in a safety-critical maritime environment.

4

Social considerations. Simion et al. (2024) reported that between 2014 and 2023, machinery damage and failure accounted for 11,506 maritime incidents, nearly four times the number caused by collisions. Accurate, explainable predictive maintenance has the potential to substantially reduce this figure, improving maritime crew safety and reducing the environmental risk of fuel contamination or uncontrolled emissions from vessels experiencing undetected propulsion degradation.

4.10 Practicality

The GridSearchCV optimisation over the XGBoost parameter space was computationally intensive and required Google Colab's cloud environment. Bagane (2024) confirms that Google Colab eliminates the hardware barrier that traditionally limits individual researchers from accessing the GPU and CPU resources required for demanding machine learning workloads, which was directly applicable to the tuning phase of this project. To manage runtime within acceptable bounds, the XGBoost and Random Forest grid searches used three-fold cross-validation within GridSearchCV rather than five-fold, a pragmatic trade-off that maintains the validity of the tuning process while substantially reducing compute time.

The CODLAG dataset originates from a simulator rather than live vessel telemetry. While Coraddu et al. (2016) validated the simulator against real frigate data to confirm its physical fidelity, simulator-generated data may not capture all real-world noise sources, sensor drift patterns, or operational failure modes. The model's generalisability to live vessel sensor streams should therefore be regarded as provisional until validated on real operational data, which represents a clear direction for future work. Furthermore, the Streamlit dashboard was developed and tested as a local deployment and does not currently incorporate authentication or network security controls necessary for an operational maritime context, a constraint that does not affect the research findings but limits immediate real-world applicability.

Chapter 5: Quality and Results

5.1 Overview

This chapter presents the experimental results of the explainable machine learning framework developed for predictive condition monitoring of naval propulsion systems. It discusses the exploratory data analysis and preprocessing outcomes, followed by baseline and optimised model evaluation using three regression algorithms for both decay targets. The chapter also examines cross validation performance, residuals, learning curves, and error distributions to assess model generalisation and reliability. SHAP analysis is presented to evaluate model explainability and identify influential features. Finally, the chapter discusses the framework's novelty, interpretation of results, technical challenges and solutions, tools and techniques, feasibility, and overall realism.

5.2 Exploratory Data Analysis and Preprocessing Outcomes

The UCI Naval Propulsion Plants dataset contained 11,934 operational records, 16 sensor-derived input features, and two regression targets, the GT Compressor decay state coefficient (kMc) and GT Turbine decay state coefficient (kMt). Initial inspection confirmed all 18 columns loaded correctly, with zero missing values and no duplicate records. This aligns with Coraddu et al. (2016), who described the CODLAG dataset as a clean simulation-derived dataset requiring no imputation, satisfying the preprocessing verification requirement of Research Objective 1.

Descriptive statistics indicated substantial variation across operational variables. Gas Turbine shaft torque (GTT) ranged from near zero to approximately 72,784 kN m, while GT Compressor inlet air temperature (T1) remained concentrated around 288 degrees Celsius. Both decay targets were concentrated near their upper bounds, with compressor decay ranging from 0.95 to 1.00 and turbine decay from 0.975 to 1.00. This healthy state dominance is common in simulation-derived propulsion datasets (Cipollini et al., 2018).

	Lever position	Ship speed (v)	Gas Turbine (GT) shaft torque (GTT) [kN m]	GT rate of revolutions (GTn) [rpm]	Gas Generator rate of revolutions (GGn) [rpm]	Starboard Propeller Torque (Ts) [kN]	Port Propeller Torque (Tp) [kN]	Hight Pressure (HP) Turbine exit temperature (T48) [C]	GT Compressor inlet air temperature (T1) [C]	GT Compressor outlet air temperature (T2) [C]
count	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.0	11934.000000
mean	5.166667	15.000000	27247.498685	2136.289256	8200.947312	227.335768	227.335768	735.495446	288.0	646.215331
std	2.626388	7.746291	22148.613155	774.083881	1091.315507	200.495889	200.495889	173.680552	0.0	72.675882
min	1.138000	3.000000	253.547000	1307.675000	6589.002000	5.304000	5.304000	442.364000	288.0	540.442000
25%	3.144000	9.000000	8375.883750	1386.758000	7058.324000	60.317000	60.317000	589.872750	288.0	578.092250
50%	5.140000	15.000000	21630.659000	1924.326000	8482.081500	175.268000	175.268000	706.038000	288.0	637.141500
75%	7.148000	21.000000	39001.426750	2678.079000	9132.606000	332.364750	332.364750	834.066250	288.0	693.924500
max	9.300000	27.000000	72784.872000	3560.741000	9797.103000	645.249000	645.249000	1115.797000	288.0	789.094000

HP Turbine exit pressure (P48) [bar]	GT Compressor inlet air pressure (P1) [bar]	GT Compressor outlet air pressure (P2) [bar]	GT exhaust gas pressure (Pexh) [bar]	Turbine Injection Control (TIC) [%]	Fuel flow (mf) [kg/s]	GT Compressor decay state coefficient	GT Turbine decay state coefficient
1934.000000	1.193400e+04	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.0000
2.352963	9.980000e-01	12.297123	1.029474	33.641261	0.662440	0.97500	0.9875
1.084770	2.533635e-13	5.337448	0.010390	25.841363	0.507132	0.01472	0.0075
1.093000	9.980000e-01	5.828000	1.019000	0.000000	0.068000	0.95000	0.9750
1.389000	9.980000e-01	7.447250	1.020000	13.677500	0.246000	0.96200	0.9810
2.083000	9.980000e-01	11.092000	1.026000	25.276500	0.496000	0.97500	0.9875
2.981000	9.980000e-01	15.658000	1.036000	44.552500	0.882000	0.98800	0.9940
4.560000	9.980000e-01	23.140000	1.052000	92.556000	1.832000	1.00000	1.0000

Figure 2 Descriptive statistics

15

The correlation heatmap in Figure 3 identified strong relationships between predictors, particularly GGn and GTn, and between P2 and Pexh. This multicollinearity may destabilise linear regression coefficients (Seber and Lee, 2012), while tree-based ensembles are more robust to correlated predictors (Breiman, 2001).

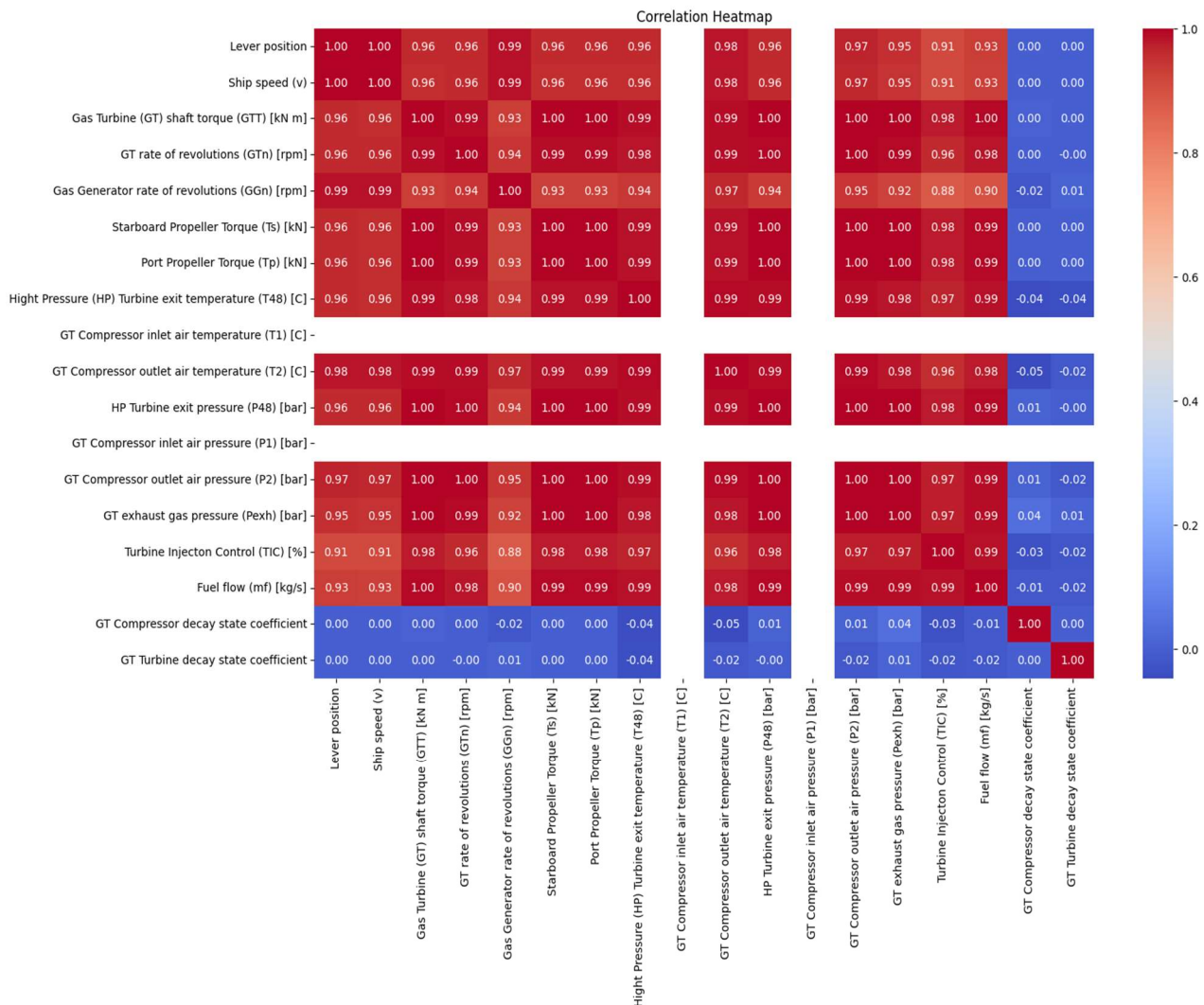
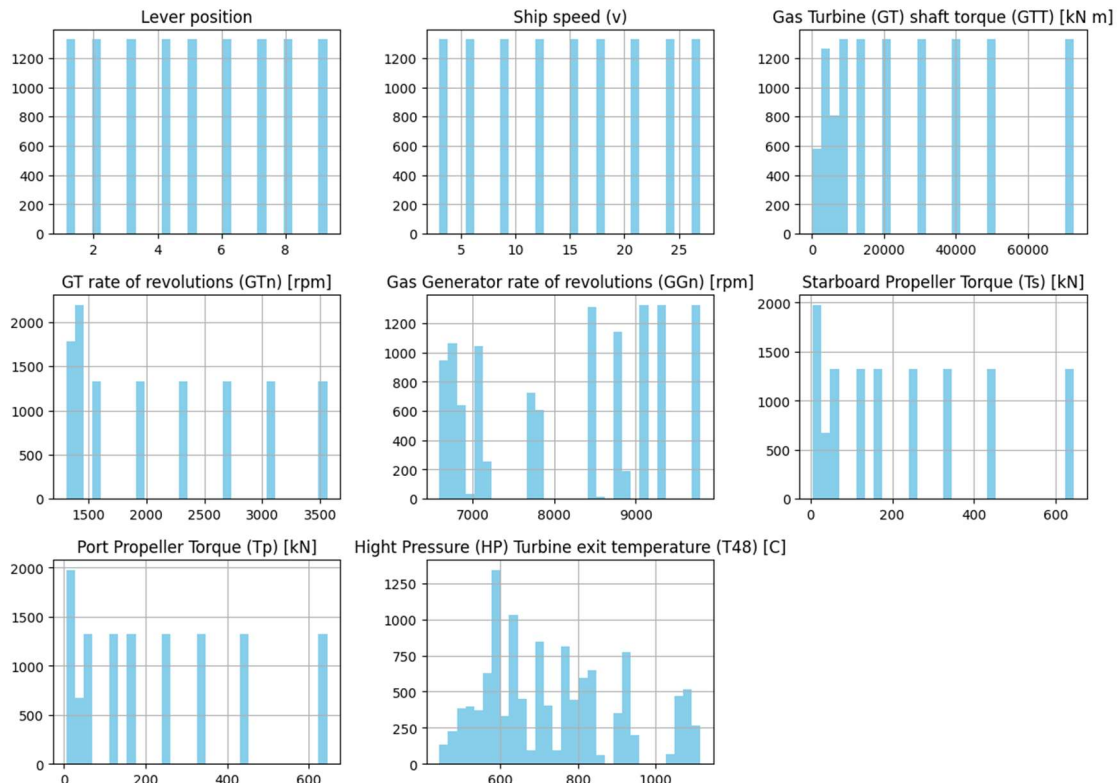


Figure 3 Correlation heatmap revealing multicollinearity patterns among all CODLAG sensor features.

112

Feature histograms in Figure 4 showed multimodal patterns associated with nine discrete lever positions. The banded structure indicates nonlinear and interaction-dependent relationships between operational conditions and decay values. The target distributions also confirmed strong concentration near the healthy upper bounds, requiring accurate prediction within narrow decay ranges.

Feature Distributions (Part 1)



Feature Distributions (Part 2)

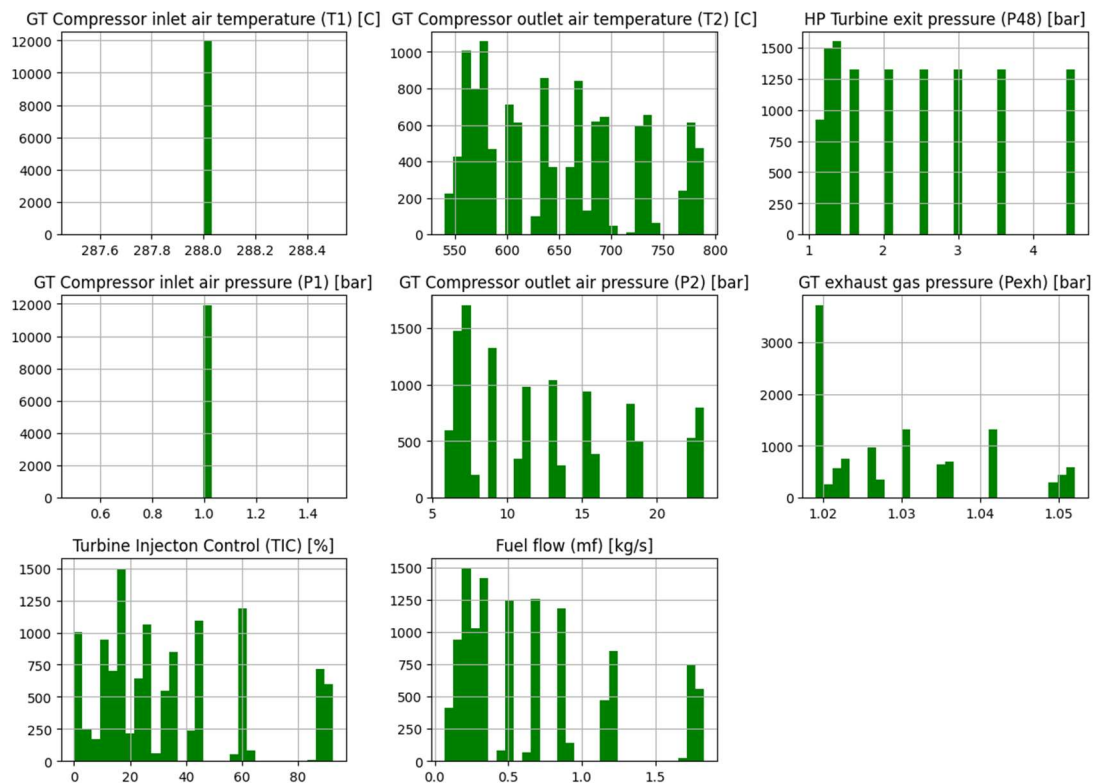


Figure 4 Feature distribution histograms and decay target distributions for both coefficients.

25

Scatter plots in Figure 5 further showed discrete relationships between compressor decay and key variables including lever position, ship speed, gas generator rate of revolutions, and GT shaft torque. These operational bands indicate that tree-based models are well suited to learning threshold-based relationships.

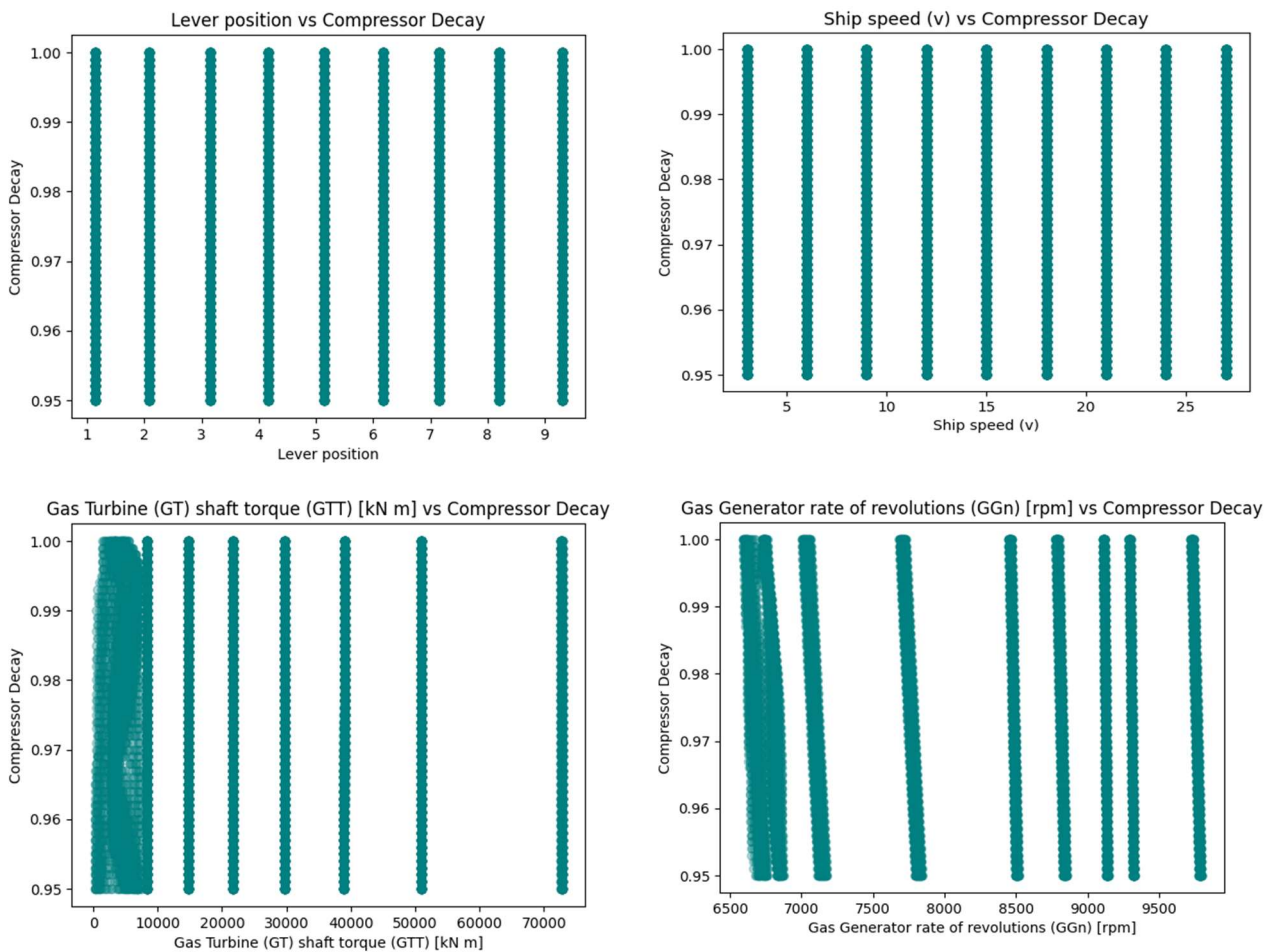


Figure 5 Scatter plots of lever position, ship speed, and shaft torque against compressor decay.

146

46

34

IQR-based outlier removal across all 16 features reduced the dataset from 11,934 to 11,742 records, removing 192 observations (approximately 1.6 per cent). Figure 6 confirms that extreme torque values were reduced without substantially affecting the core distributions. The data were then divided into 80 per cent training and 20 per cent testing sets using random state 42. StandardScaler was fitted only on the training data before transforming the test set, preventing leakage and supporting reliable generalisation assessment (Kuhn and Johnson, 2013).

```

▶ print("Original shape:", df.shape)
  df = remove_outliers(df, feature_cols)
  |
  print("After outlier removal:", df.shape)

... Original shape: (11934, 18)
   After outlier removal: (11742, 18)

```

Figure 6 Dataset size before and after IQR outlier removal confirming 11,742 retained records.

5.3 Baseline Model Evaluation

In alignment with Research Objective 2 and Sub-Question 1, Linear Regression, Random Forest, and XGBoost were evaluated for the GT Compressor decay state coefficient (kMc) and GT Turbine decay state coefficient (kMt). Models were trained on the scaled training partition and assessed on the held-out test set using R2, RMSE, and MAE. Table 3 presents the combined results.

Table 3 Baseline model performance for compressor and turbine decay targets

Target	Model	R2	RMSE	MAE
Compressor	Linear Regression	0.8375	0.005892	0.004646
Compressor	Random Forest	0.9962	0.000905	0.000480
Compressor	XGBoost	0.9926	0.001253	0.000897
Turbine	Linear Regression	0.9082	0.002264	0.001725
Turbine	Random Forest	0.9923	0.000654	0.000323
Turbine	XGBoost	0.9791	0.001079	0.000667

Linear Regression produced the weakest results for both targets, although its turbine performance was stronger than its compressor performance. For compressor decay, the R2 of 0.8375 indicates that approximately 16.25 per cent of variance remained unexplained. Its MAE of 0.004646 represents approximately 9.3 per cent of the narrow 0.05-unit compressor decay range, making the error potentially significant for condition monitoring. Figure 7 shows systematic deviation from the perfect prediction diagonal, particularly at lower decay values. This structured bias indicates model misspecification rather than random prediction noise. The finding supports Cipollini et al. (2018), who identified limitations of linear parametric approaches when naval propulsion variables exhibit nonlinear relationships.

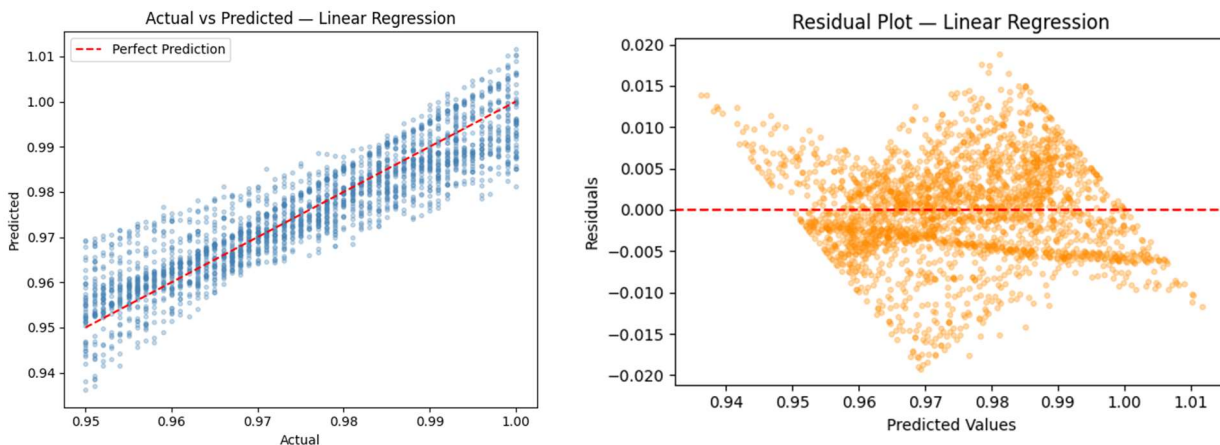


Figure 7 Actual versus predicted plot for Linear Regression on compressor decay target

Random Forest substantially improved compressor prediction, achieving an R2 of 0.9962, RMSE of 0.000905, and MAE of 0.000480. Figure 8 shows close alignment between actual and predicted values, indicating that the model captured nonlinear relationships more effectively. XGBoost also performed strongly, with an R2 of 0.9926, RMSE of 0.001253, and MAE of 0.000897, although it remained inferior to Random Forest across all metrics. Figure 8 confirms slightly greater prediction dispersion. The difference is therefore not attributable to R2 alone, since both error measures also favour Random Forest.

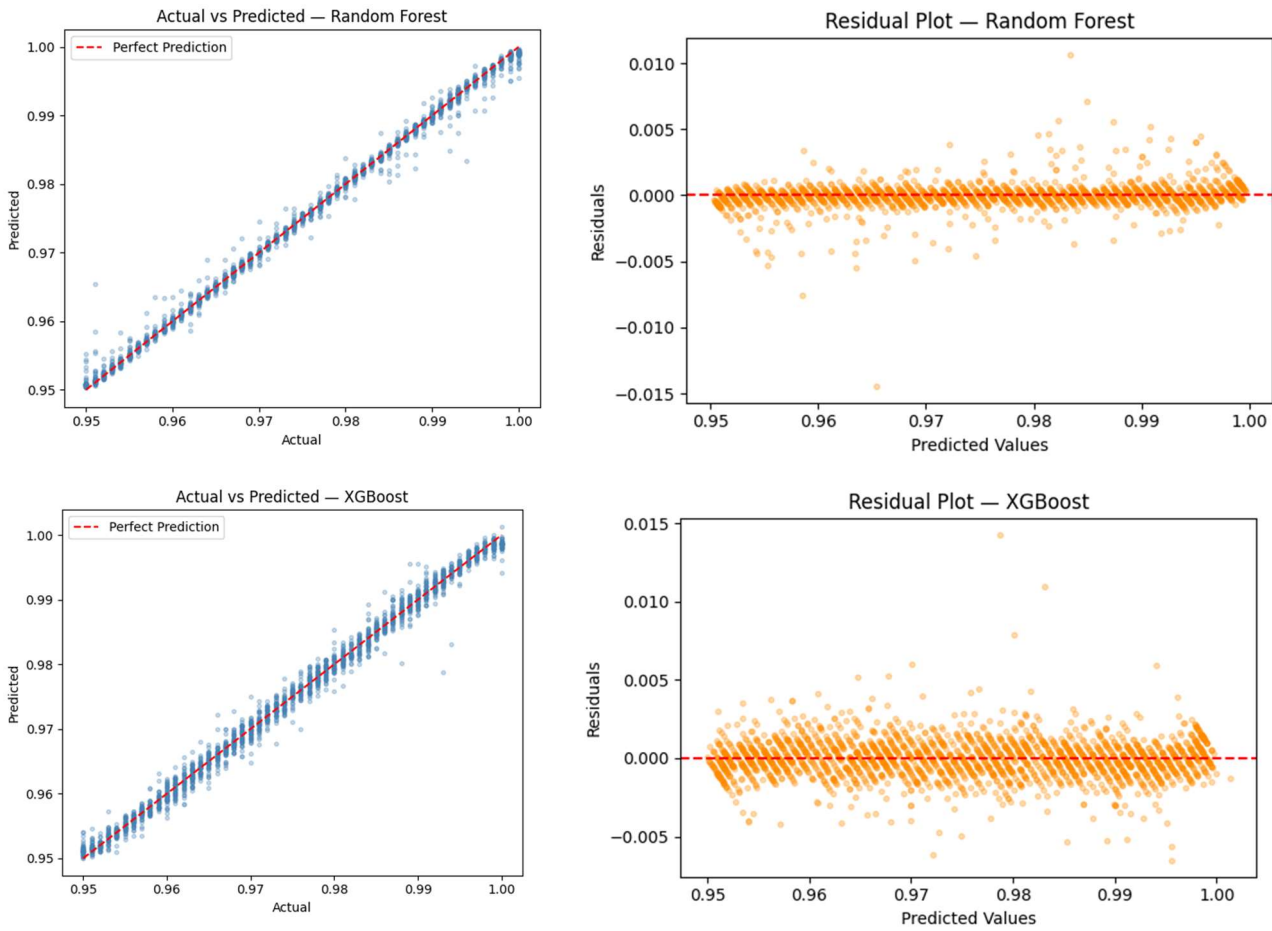


Figure 8 Actual versus predicted plot for XGBoost baseline on compressor decay target.

A similar pattern occurred for turbine decay. Linear Regression achieved an R^2 of 0.9082, suggesting that this target contains a stronger linear component. However, its RMSE of 0.002264 and MAE of 0.001725 remain substantial relative to the narrow 0.025 unit target range. Figure 9 shows structured residual behaviour, indicating systematic errors caused by unmodelled interactions, consistent with Géron (2019). Thus, higher R^2 does not necessarily indicate sufficient predictive suitability for deployment.

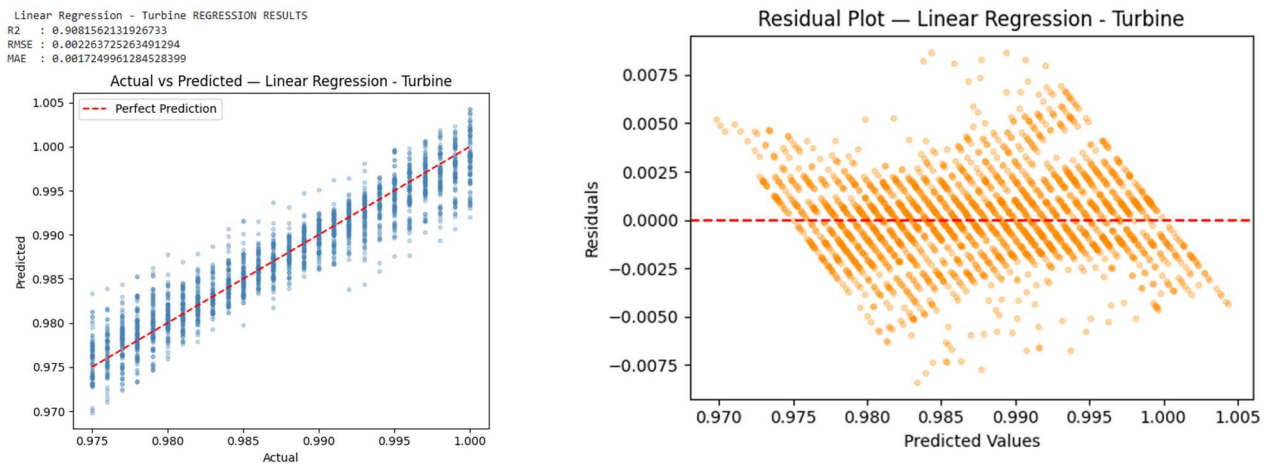


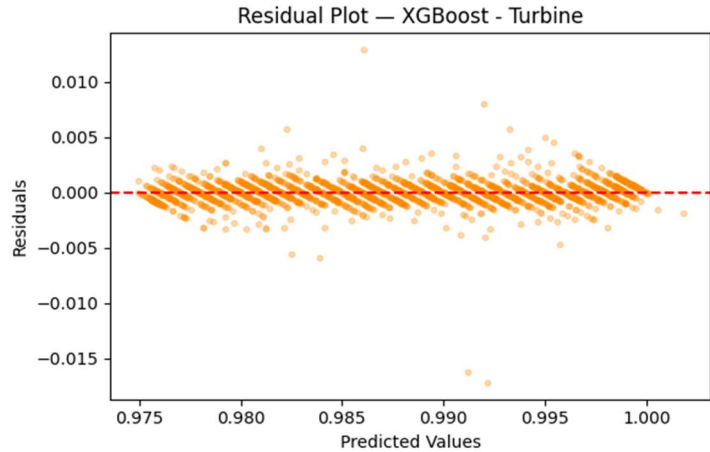
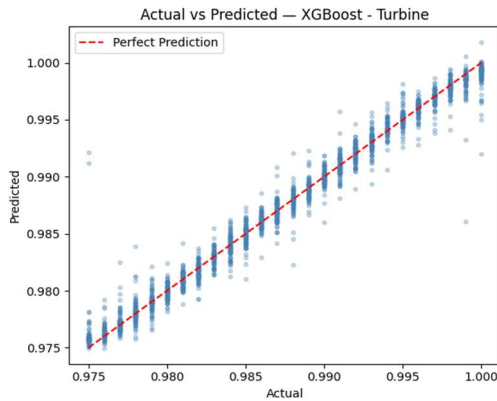
Figure 9 Actual versus predicted and residual plots for Linear Regression on turbine decay.

9
9

Random Forest again achieved the strongest baseline performance, with an R2 of 0.9923, RMSE of 0.000654, and MAE of 0.000323. XGBoost achieved an R2 of 0.9791, an RMSE of 0.001079, and an MAE of 0.000667. Figure 10 shows greater prediction variability for XGBoost, while its RMSE was approximately 65 percent higher than Random Forest. Figures 10 and 11 therefore establish Random Forest as the baseline champion for both targets.

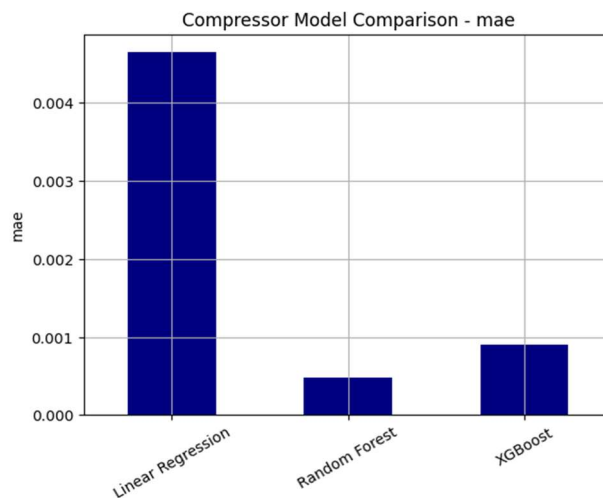
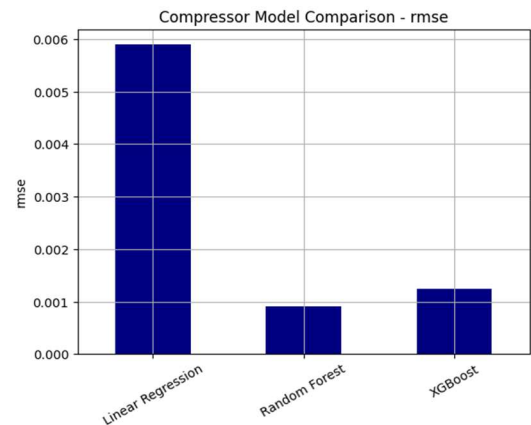
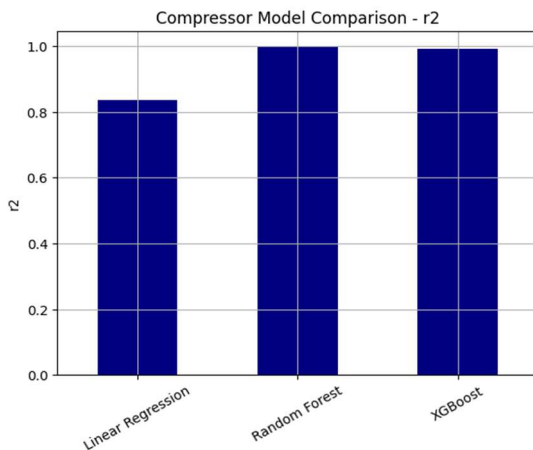
XGBoost - Turbine REGRESSION RESULTS

R2 : 0.9791379418456415
RMSE : 0.0010788905834841689
MAE : 0.0006672838399033701



122

Figure 10 Actual versus predicted and residual plots for XGBoost on turbine decay.



16

Figure 11 Baseline compressor model comparison bar charts for R2, RMSE, and MAE metrics.

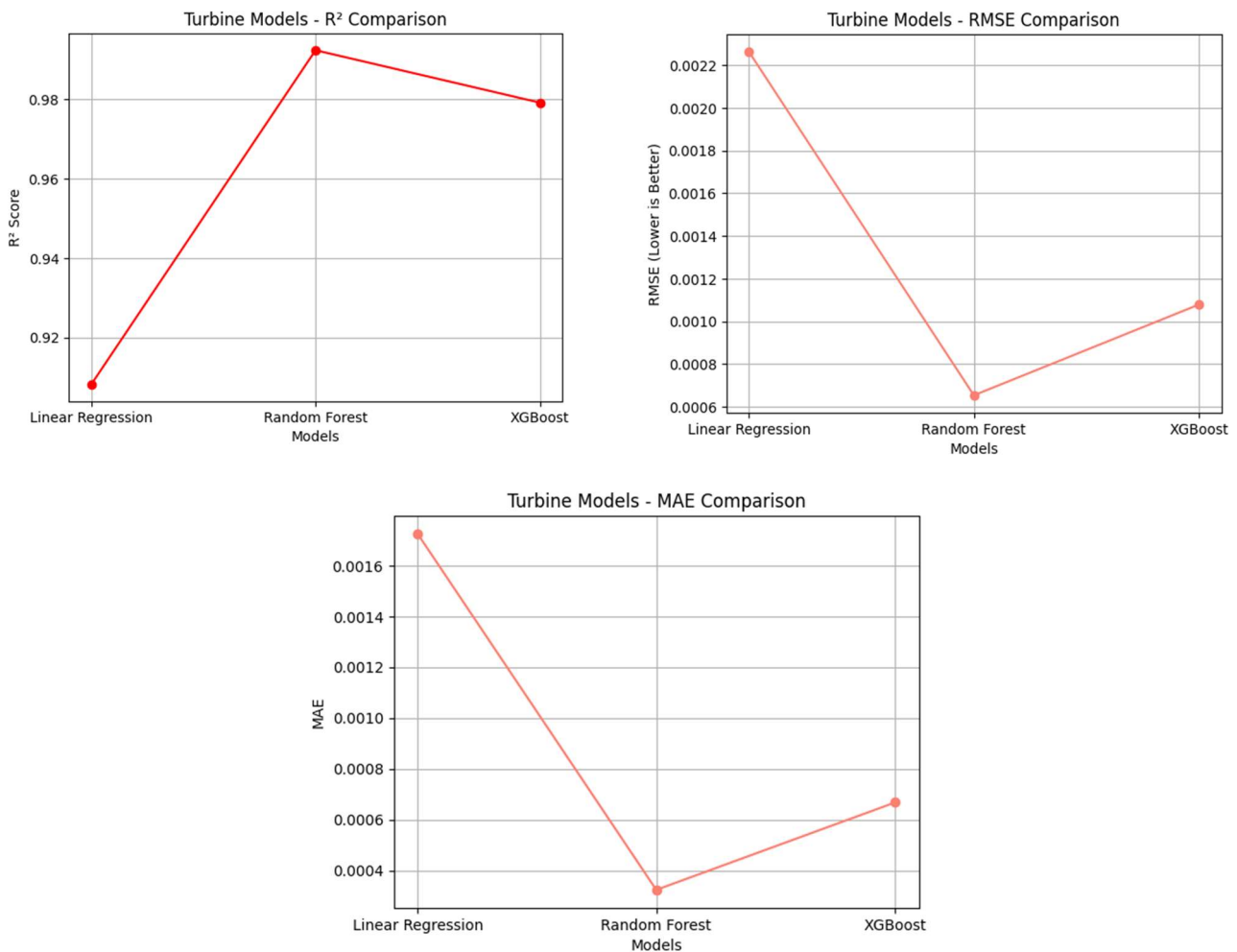


Figure 12 Baseline turbine model comparison charts for R², RMSE, and MAE metrics.

Critically, interpret the very high ensemble R² values alongside the narrow, highly concentrated target distributions identified in Section 5.2. Consequently, small absolute errors can produce exceptionally high R² values, meaning R² alone cannot establish operational reliability. The baseline results therefore provide a benchmark rather than a final model decision. Random Forest demonstrates superior initial predictive accuracy, while XGBoost remains sufficiently competitive to justify optimisation. The following tuning analysis consequently tests whether parameter optimisation can reduce XGBoost errors and close or reverse the baseline performance gap before generalisation and explainability are assessed.

5.4 Hyperparameter Tuning and Optimised Model Evaluation

Hyperparameter tuning produced measurable and analytically significant improvements across both decay targets, with XGBoost demonstrating the greatest sensitivity to optimisation while Linear Regression and Random Forest remained largely stable. This pattern is consistent with the known characteristics of gradient-boosted trees, where the interaction between learning rate, tree depth, and subsampling rate creates a rich and responsive hyperparameter surface that rewards systematic search (Chen and Guestrin, 2016). Table 4 consolidates the tuned performance of all three models across both targets, enabling direct cross-target comparison of the gains achieved through GridSearchCV optimisation.

Table 4 Tuned model performance on GT Compressor and GT Turbine decay state coefficients

Target	Model	Tuned R^2	Tuned RMSE	Tuned MAE
Compressor	Linear Regression	0.8375	0.005892	0.004646
Compressor	Random Forest	0.9962	0.000905	0.000480
Compressor	XGBoost (Tuned)	0.9962	0.000897	0.000605
Turbine	Linear Regression	0.9082	0.002264	0.001725
Turbine	Random Forest	0.9923	0.000655	0.000323
Turbine	XGBoost (Tuned)	0.9907	0.000721	0.000478

For the compressor target, tuning elevated XGBoost from a baseline R^2 of 0.9926 to 0.9962, achieving exact parity with Random Forest while reducing RMSE from 0.001253 to 0.000897 and MAE from 0.000897 to 0.000605. This post-tuning parity with Random Forest is critically significant. Schonlau and Zou (2020) note that Random Forest's bagging-based variance reduction consistently delivers strong out-of-the-box performance on non-linear regression tasks, making it a demanding benchmark to match. The fact that tuned XGBoost reaches this benchmark while also offering native compatibility with SHAP TreeExplainer for exact per-prediction attribution confirms that optimisation has closed the architectural trade-off identified at baseline without sacrificing predictive quality. The actual-versus-predicted and residual plots for the tuned XGBoost compressor model, shown in Figure 14, confirm this improvement visually. Predictions align tightly along the perfect prediction diagonal across the full decay range from 0.95 to 1.00, and the residual scatter is narrow and approximately symmetric around zero, contrasting sharply with the fan-shaped heteroscedastic pattern produced by Linear Regression, which Géron (2019) identifies as a fundamental symptom of model misspecification under non-linear data-generating processes.

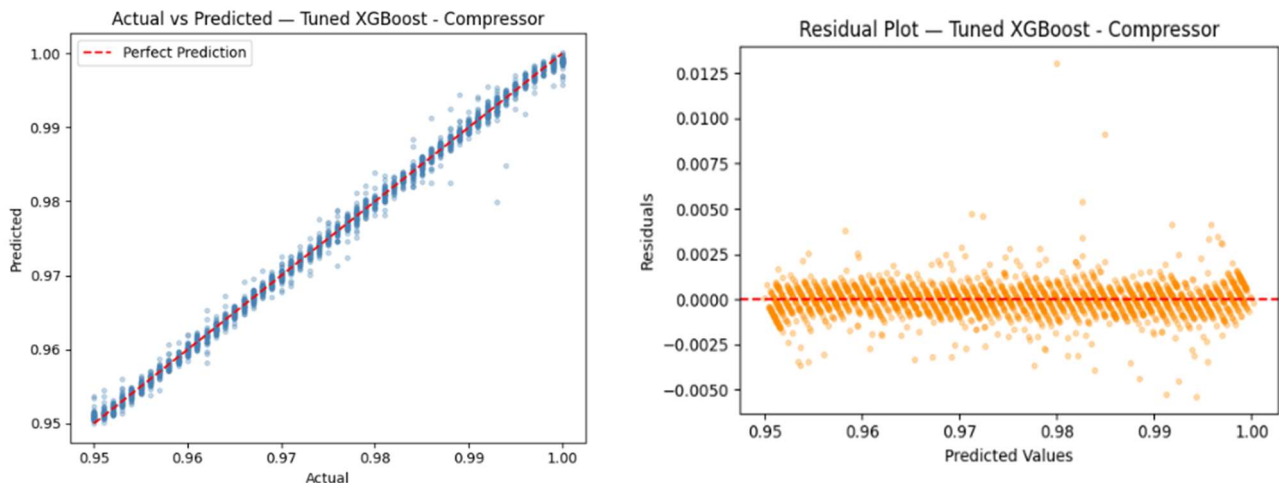


Figure 13 Actual versus predicted and residual plots for tuned XGBoost on compressor decay.

The turbine target produced the most revealing tuning result of the entire experimental programme. XGBoost improved from a baseline R^2 of 0.9791 to 0.9907 after optimisation, a gain of 0.0116 R^2 units representing the largest single-model improvement observed across all models and both targets. RMSE reduced from 0.001079 to 0.000721 and MAE from 0.000667 to 0.000478. This result directly challenges the interpretation that the turbine target is inherently harder to predict than the compressor target. The baseline gap between XGBoost and Random Forest on turbine was 0.0132 R^2 units, yet tuning closed that gap to just 0.0016 units. Sahraoui et al. (2024) similarly observed that boosted ensemble methods applied to CODLAG decay targets demonstrate substantially greater

performance sensitivity to configuration than static ensemble methods such as bagging, reinforcing the finding that the turbine decay surface is uniquely responsive to gradient boosting hyperparameter adjustment. The practical implication is direct: dedicated tuning resources applied to real operational vessel data would likely yield disproportionately greater turbine prediction gains than compressor gains, a consideration relevant to any future deployment of this framework. The residual plots for the tuned turbine model, shown in Figure 14, display a structured banded pattern corresponding to the nine discrete lever position operating states, which Coraddu et al. (2016) describe as an inherent feature of the CODLAG simulation design rather than a model deficiency, confirming the absence of systematic prediction bias.

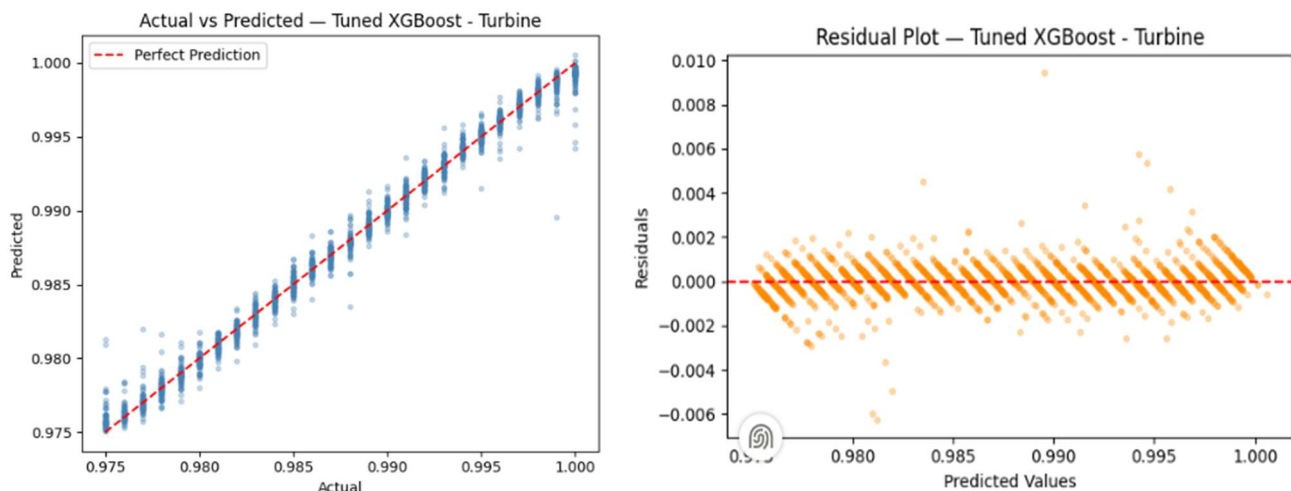


Figure 14 Actual versus predicted and residual plots for tuned XGBoost on turbine decay.

The before-and-after R^2 comparison in Figure 15 and the MAE reduction comparison in Figure 16 visually consolidate these gains. The XGBoost compressor MAE reduction of 32.6 percent and turbine MAE reduction of 28.3 percent translate directly into more precise health index calculations and more reliable maintenance decision thresholds within the Digital Twin dashboard. Chicco et al. (2021) confirm that MAE improvements of this magnitude in narrow-range regression targets are operationally meaningful because they reduce the absolute prediction deviation that a domain practitioner must account for when interpreting model outputs.

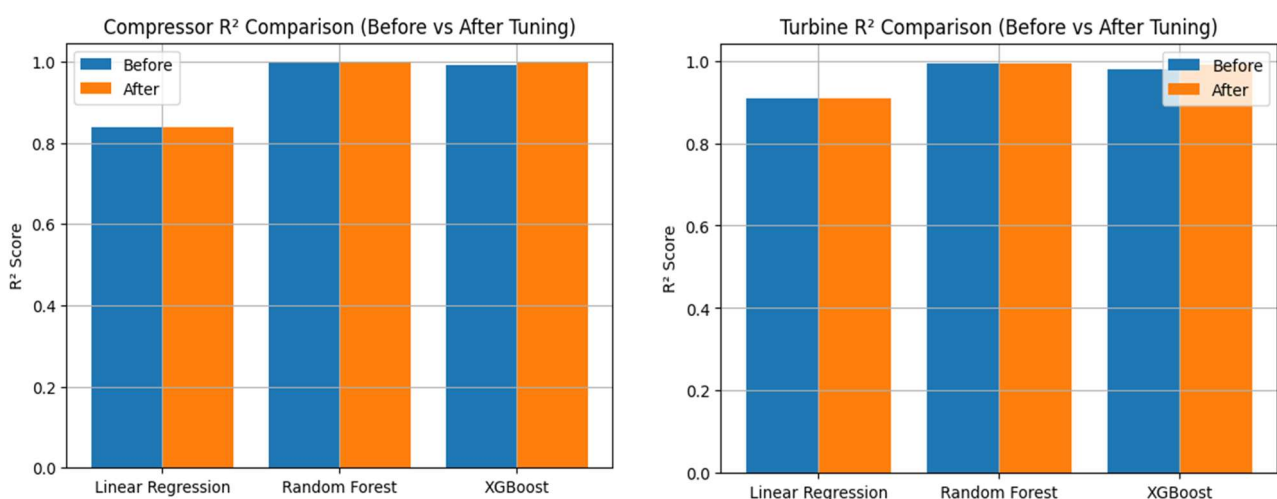


Figure 15 Before-and-after R2 comparison for compressor and turbine decay targets.

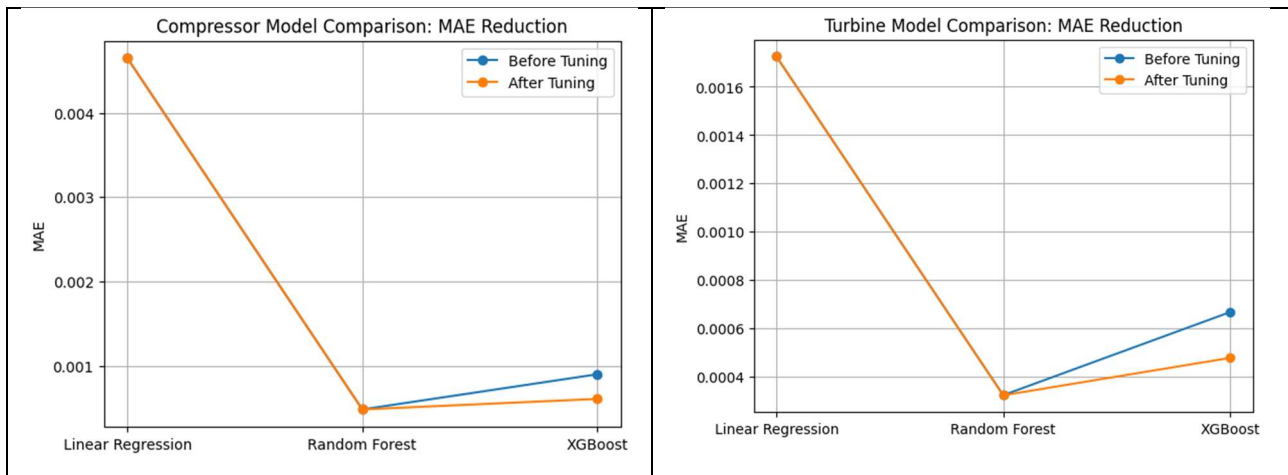


Figure 16 MAE reduction before and after tuning for compressor and turbine XGBoost models.

XGBoost was selected as the final production model on two decisive grounds. First, it achieved equivalent predictive accuracy to Random Forest following tuning on both targets, with cross-validation R^2 of 0.9915 on compressor closely mirroring the tuned test R^2 of 0.9962, confirming that performance gains reflect genuine generalisation rather than an optimistic partition artefact, consistent with the validation framework recommended by Vabalas et al. (2019). Second, its sequential gradient-boosting architecture is natively compatible with SHAP TreeExplainer for exact Shapley value computation, whereas Random Forest's parallel averaging architecture produces less precise per-prediction attributions at the individual observation level (Lundberg et al., 2020). This selection directly and affirmatively answers Sub-Question 1, confirming that tuned XGBoost most accurately estimates both compressor and turbine decay state coefficients when evaluated against the combined criteria of predictive accuracy, generalisation stability, and explainability compatibility.

5.5 Cross-Validation Analysis

Five-fold cross-validation was conducted independently for all three models on both regression targets to assess generalisation performance beyond the single train-test split, directly addressing the multi-stage validation requirement of Research Objective 3. Table 5 presents the cross-validation results.

Table 5 Five-fold cross-validation results for compressor and turbine decay targets

Target	Model	CV R2	CV RMSE	CV MAE
Compressor	Linear Regression	0.8394	0.005867	0.004612
Compressor	Random Forest	0.9939	0.001145	0.000586
Compressor	XGBoost	0.9915	0.001345	0.000952
Turbine	Linear Regression	0.9096	0.002250	0.001693
Turbine	Random Forest	0.9872	0.000843	0.000389
Turbine	XGBoost	0.9818	0.001008	0.000674

Cross-validation results are closely aligned with the held-out test set results for all models across both targets, confirming that the single 20 per cent test partition evaluation was not a statistical anomaly and that model rankings are consistent regardless of evaluation method. The XGBoost compressor cross-validation R^2 of 0.9915 compares very favourably with its test set baseline R^2 of 0.9926, and the small positive difference reflects the expected variance between averaged five-fold performance and single-split performance rather than overfitting. This consistency across evaluation methods confirms the robustness of the modelling pipeline and satisfies the cross-validation component of Research Objective

3 (Arlot and Celisse, 2010). For the turbine target, the XGBoost cross-validation R2 of 0.9818 aligns closely with the baseline test R2 of 0.9791, again confirming generalisation stability. The cross-validation results therefore provide independent empirical confirmation of the model ranking established through test set evaluation and validate the selection of XGBoost as the production model.

5.6 Error Distribution Analysis

Error distribution analysis examined the frequency and spread of prediction errors for all three models on both targets, providing an additional lens for evaluating model precision and bias beyond the summary metrics of Table 3 and Table 4. Error histograms were generated using the signed residual values to detect both systematic directional bias and overall error spread.

The error distribution for Linear Regression on the compressor target, shown in Figure 17, is a broad, approximately Gaussian spread with a standard error of 0.005892, the widest of the three models. The distribution is centred near zero, indicating no strong systematic directional bias, but its width confirms that individual predictions can deviate from true values by up to 0.02 decay units in the tail regions. Given that the full compressor decay range spans only 0.05 units, Linear Regression errors can represent up to 40 per cent of the observable decay range, confirming its unsuitability for this condition monitoring task.

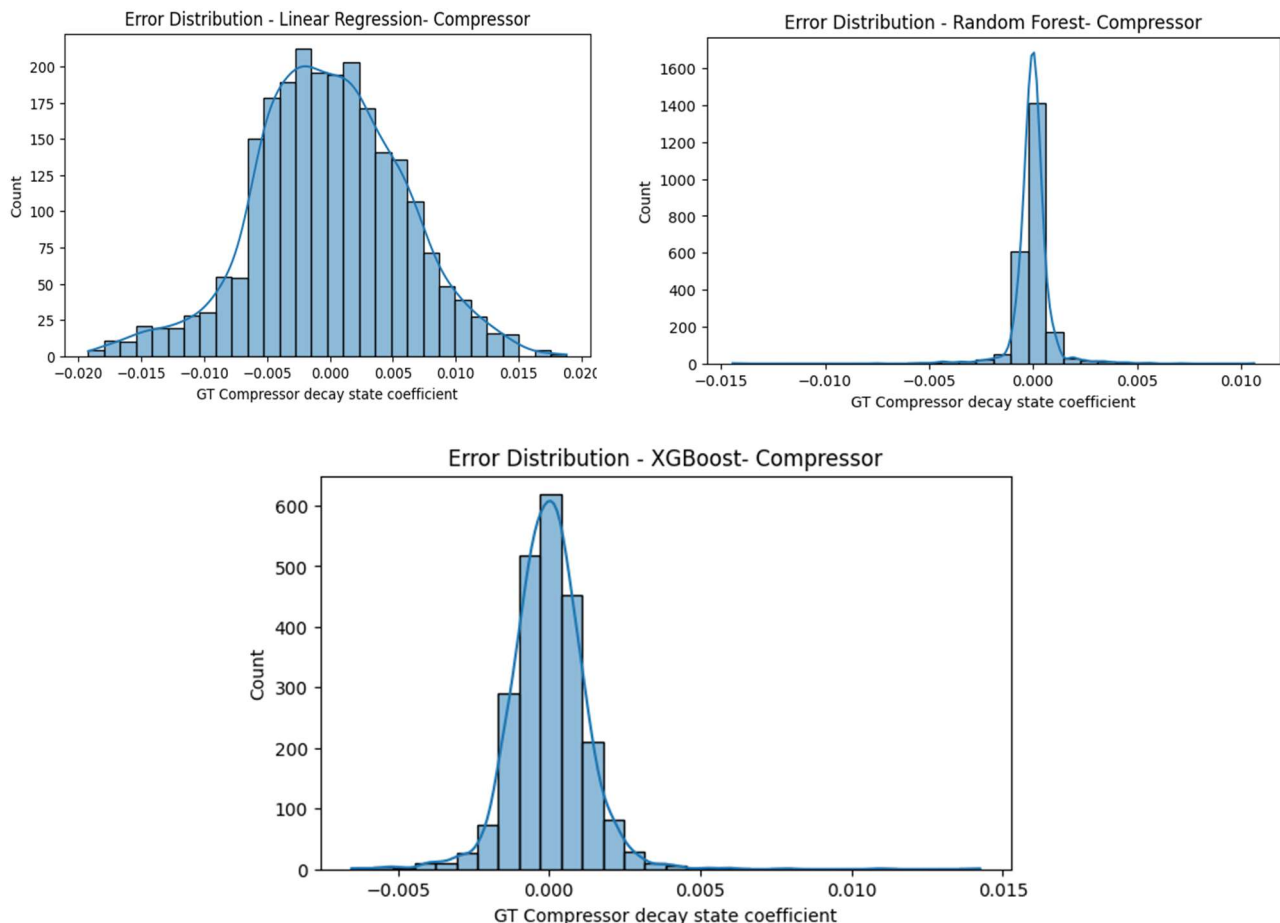


Figure 17 Error distribution histogram for linear regression, Random Forest and XGBoost baseline on the compressor decay target.

The error distributions for Random Forest and XGBoost at baseline on the compressor target, as shown in Figure 17, display sharply concentrated bell-shaped distributions centred at zero, with standard errors of 0.000905 and 0.001254, respectively. The mean error for the XGBoost baseline on the compressor target was approximately minus 1.29e-05, confirming near-zero systematic bias. Following tuning, the XGBoost compressor error distribution, as shown in Figure 18, tightened further with a mean

error of minus 2.62×10^{-6} and standard error of 0.000897, confirming that tuning improved not only R2 and RMSE but also the symmetry and precision of the individual prediction error distribution.

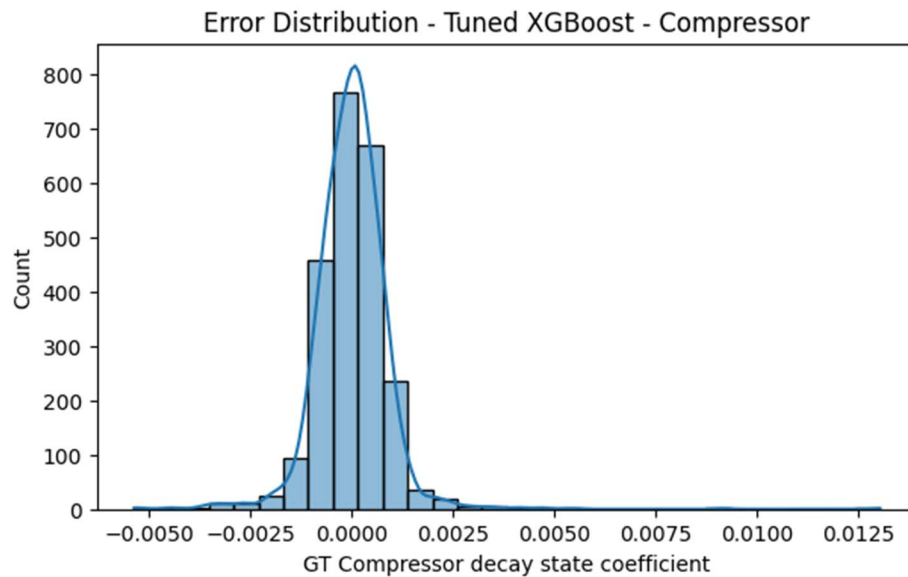


Figure 18 Error distribution for the tuned XGBoost compressor model, confirming a tight, zero-centred spread.

For the turbine target, the error distributions across all three models showed a similar hierarchy. The tuned XGBoost turbine error distribution, as shown in Figure 19, displayed a mean error of 8.26×10^{-6} and standard error of 0.000721, the tightest and most precisely centred of all turbine model distributions. The progression from the broad Gaussian spread of Linear Regression through the moderate concentration of untuned XGBoost to the sharp-peaked distribution of tuned XGBoost visually demonstrates the improvement trajectory achieved through the full modelling pipeline and provides strong empirical support for the alternative hypothesis H1 that XGBoost achieves significantly higher predictive accuracy than the baseline models (Lundberg and Lee, 2017).

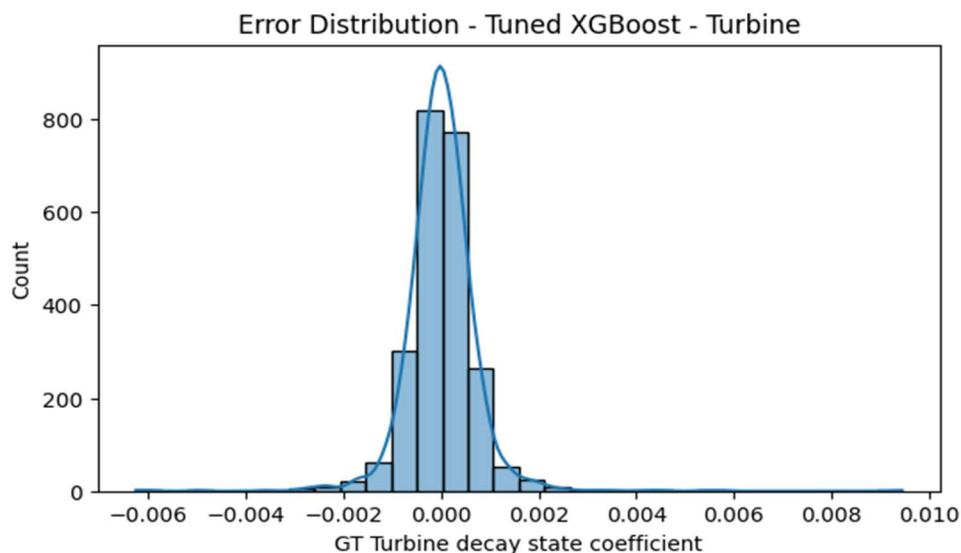


Figure 19 Error distribution for the tuned XGBoost turbine model, confirming a concentrated, zero-centred distribution.

5.7 Learning Curve Analysis

Learning curve analysis evaluated how predictive performance scaled with increasing training data volume for all three models on both targets, providing evidence toward the generalisation and robustness

components of Research Objective 3. Training and validation R2 scores were computed at incremental training set sizes from 1,000 samples to the full 7,400-sample training partition.

The learning curves for Random Forest and XGBoost on the compressor target, as shown in Figure 20, reveal a characteristic pattern of rapid validation R2 improvement as training size increases from 1,000 to approximately 4,000 samples, followed by a gradual plateau as the full training partition is approached. At maximum training size, the tuned XGBoost compressor model achieved a training R2 of approximately 0.999 and a validation R2 of approximately 0.990, with a narrow and stable gap between the two curves. This narrow train-validation gap is the critical indicator that the model has reached a well-fitted equilibrium with neither underfitting nor overfitting (Géron, 2019). The convergence of both curves also confirms that the available CODLAG dataset of 11,742 records is sufficiently large to train a reliable XGBoost compressor model without data starvation, satisfying the dataset adequacy requirement implicit in Research Objective 3.

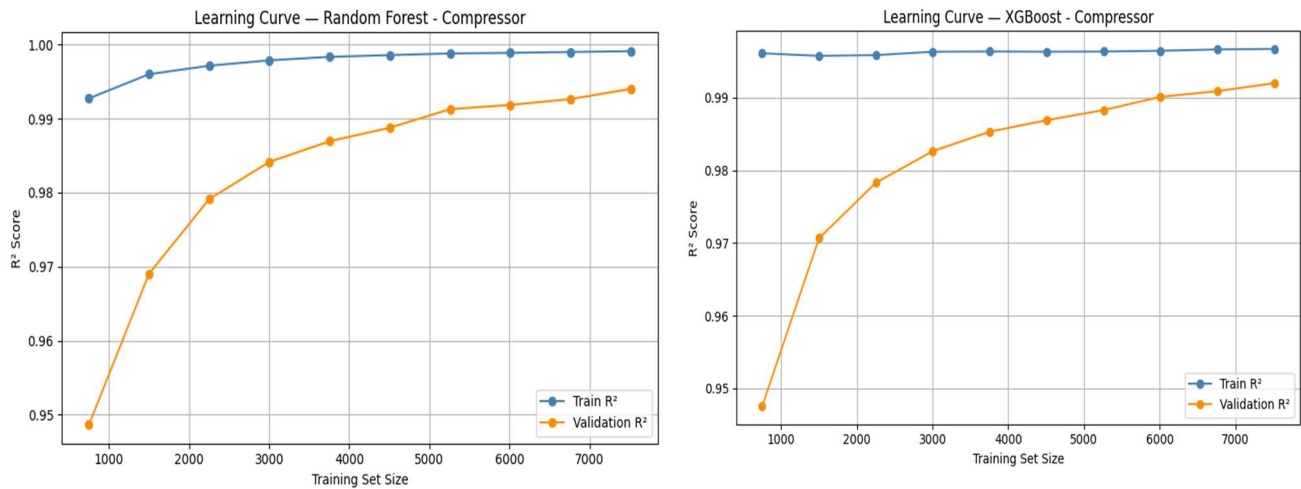


Figure 20 Learning curves for Random Forest and XGBoost on compressor decay target.

The learning curves for all three models on the turbine target, as shown in Figure 21, display a similar convergence pattern for ensemble models. The XGBoost turbine learning curve shows training R2 remaining stable near 0.993, while validation R2 rises from approximately 0.900 at 1,000 samples to approximately 0.981 at full training size. The continued upward trend of the validation curve at the maximum observed training size suggests that the XGBoost turbine model could potentially benefit from additional training data beyond the 11,742 records available in this dataset. This finding has direct implications for real-world deployment as articulated in Research Objective 5, where live operational data streams from commissioned naval vessels would provide a continuously growing training corpus that could further strengthen turbine decay predictions over time. The Linear Regression learning curves for both targets display rapid convergence at relatively low R2 values, confirming that their performance ceiling is capacity-limited rather than data-limited and that no amount of additional data would bring Linear Regression to the performance level of the ensemble models.

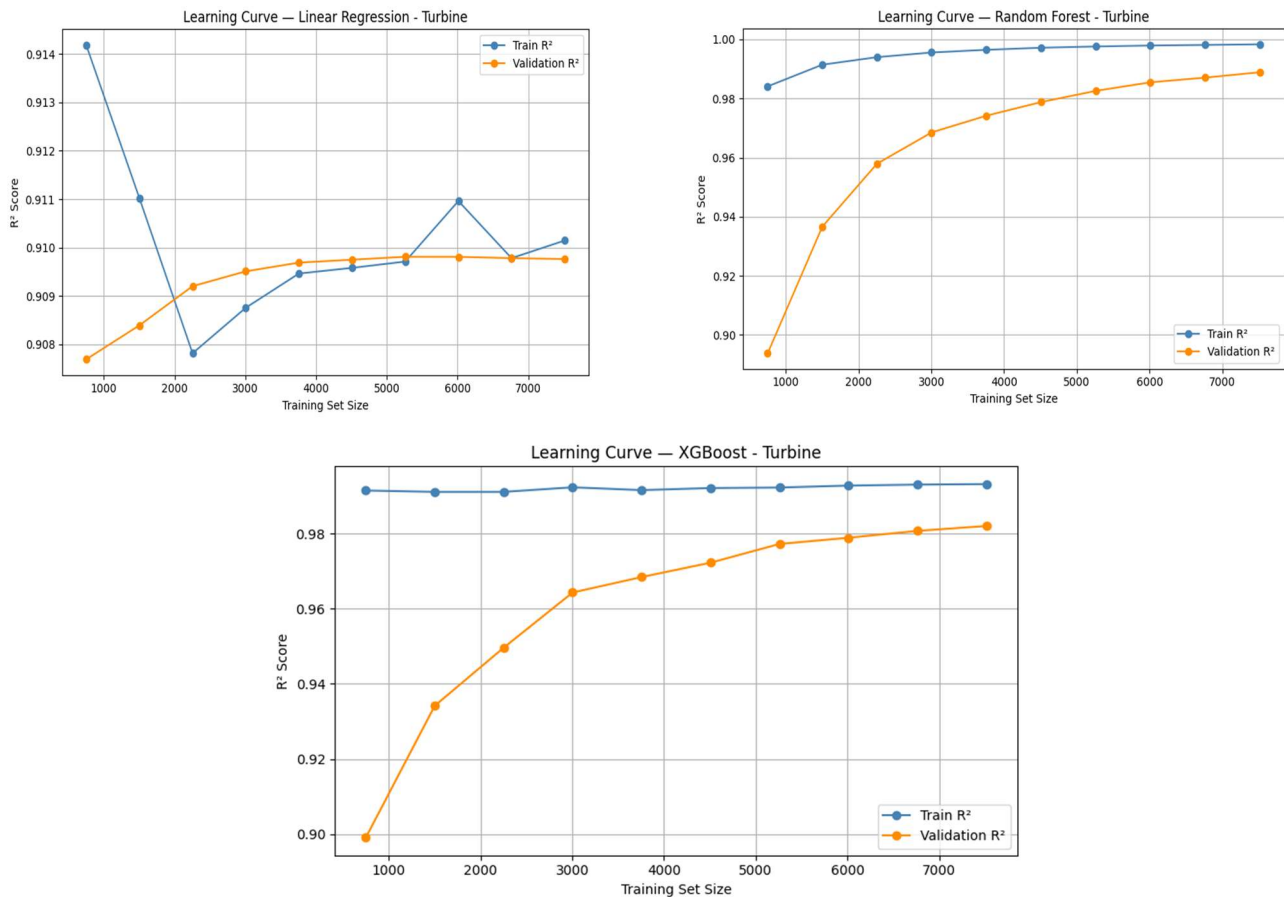


Figure 21 Learning curves for all three models on turbine decay target showing convergence patterns.

5.8 SHAP Explainability Analysis

SHAP analysis was conducted using `shap.TreeExplainer` independently on the tuned XGBoost compressor and turbine models, directly addressing Research Objective 4 and Sub Question 2. TreeExplainer calculates exact Shapley values by exploiting the tree structure of XGBoost ensembles through a polynomial-time algorithm. This avoids the approximation errors associated with kernel- and permutation-based methods and ensures that feature attributions remain mathematically grounded in cooperative game theory (Lundberg et al., 2020). During implementation, the generic `shap.Explainer` initially triggered a `PermutationExplainer` fallback, producing approximate and computationally expensive results. Replacing this with `TreeExplainer` was therefore essential for the accuracy and interpretability of the reported SHAP outputs.

Compressor Model: Global Feature Importance

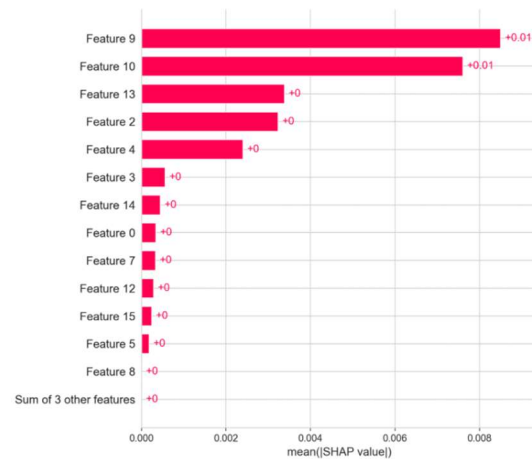
The global feature importance and beeswarm plots in Figure 22 show that GT Compressor outlet air temperature (T2) and GT Compressor outlet air pressure (P2) are the two most influential features, with mean absolute SHAP values of approximately 0.009 and 0.008, respectively. These values substantially exceed those of the remaining features. The beeswarm plot shows that high T2 and P2 values produce positive SHAP contributions that move predicted compressor decay towards 1.0, indicating healthier operation, whereas lower values contribute towards degraded states.

This relationship is physically coherent because compressor outlet temperature and pressure are direct thermodynamic indicators of aerodynamic efficiency. Compressor blade or seal degradation can reduce both pressure ratio and temperature rise across the compressor stage (Coraddu et al., 2016). The dominance of these thermodynamic variables therefore suggests that XGBoost has captured meaningful

propulsion relationships rather than relying solely on dataset-specific statistical patterns. This provides supporting evidence for the affirmative answer to Sub Question 2.

Global Feature Importance — Bar Plot

Mean absolute SHAP values across 500 sampled records.



SHAP Summary Plot — Beeswarm

Distribution of SHAP values across all sampled records per feature.

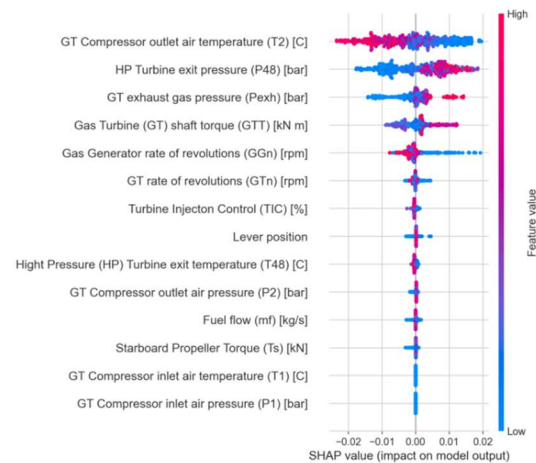


Figure 22 Global feature importance bar plot and SHAP beeswarm summary plot for the tuned XGBoost compressor model.

Compressor Model: Per Prediction Waterfall Analysis

The healthy compressor waterfall plot in Figure 23 shows how the model produces a final prediction of $f(x) = 0.978$ from a base expected value of $E[f(X)] = 0.975$. Feature 10 provides the largest positive contribution of approximately $+0.01$, while Features 2 and 9 make smaller negative contributions, resulting in a prediction close to the healthy upper bound.

The three scenario comparison further demonstrates an attribution shift between healthy, moderate, and severe degradation states. Under healthy conditions, several features contribute positively and collectively support a prediction near 0.998. At the moderate state, represented by an index of approximately 60, Feature 4 produces an increasingly negative contribution of around -0.01 , while Feature 13 also shifts downward. Under severe degradation at an index of approximately 0, Features 10 and 9 show strong negative contributions of around -0.01 , reducing the final prediction towards 0.953. This progressive change provides per-prediction transparency by showing which operational parameters contribute to declining compressor health. Such capability is not evident in previous CODLAG based modelling approaches (Cipollini et al., 2018).



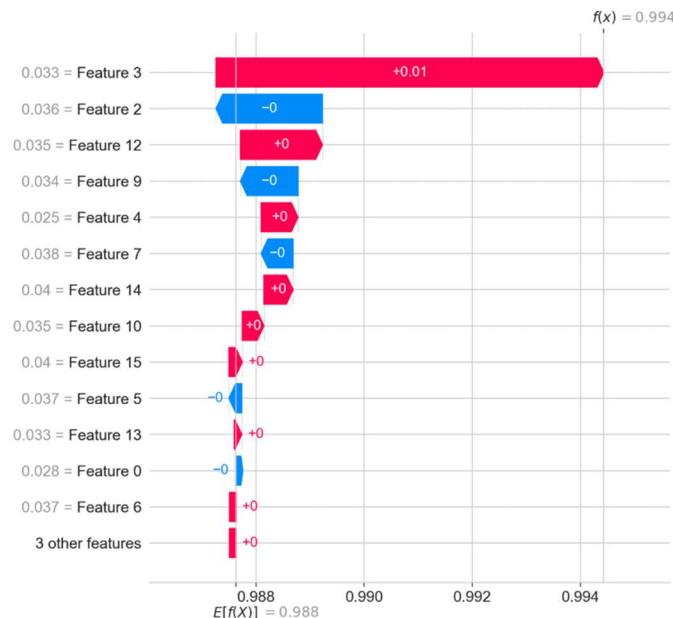
Figure 23 SHAP waterfall plots for the compressor model under healthy, moderate, and severe degradation scenarios, alongside the three scenario comparison panel.

Turbine Model: Global Feature Importance and Per Prediction Analysis

19

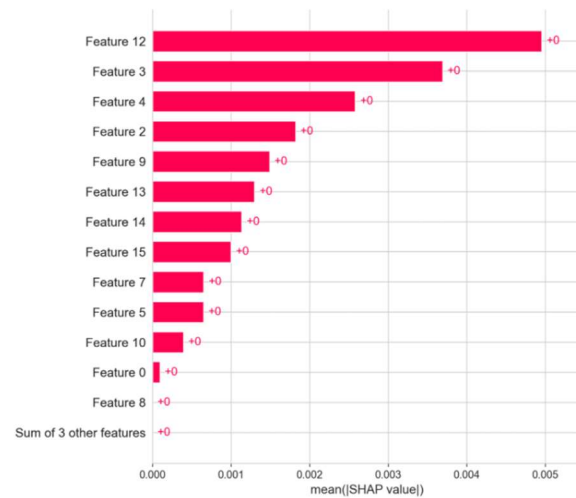
The turbine global importance and beeswarm plots in Figure 24 identify Feature 12 as the dominant contributor, with a mean absolute SHAP value of approximately 0.005, followed by Features 3, 4, and 2. The beeswarm indicates that P2, GTn, GGn, and GTT collectively influence turbine decay attribution. The stronger role of shaft torque is physically meaningful because turbine mechanical loading is expressed more directly through shaft torque than compressor aerodynamic loading (Boyce, 2012).

The healthy turbine waterfall shows a final prediction of $f(x) = 0.994$ from a base value of $E[f(X)] = 0.988$. Feature 3 provides the largest positive contribution of approximately +0.01. This indicates that the turbine model has developed a distinct attribution pattern that remains physically consistent with the compressor model while reflecting the turbine system's different behaviour.



Global Feature Importance — Bar Plot

Mean absolute SHAP values across 500 sampled records.



SHAP Summary Plot — Beeswarm

Distribution of SHAP values across all sampled records per feature.

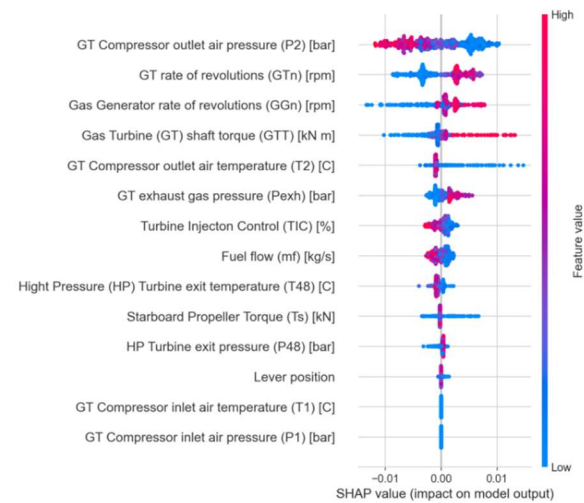


Figure 24 Global feature importance bar plot, SHAP beeswarm summary plot, and healthy scenario waterfall for the tuned XGBoost turbine model.

The turbine scenario comparison in Figure 25 further shows that Feature 4 and Feature 12 accumulate increasingly negative SHAP contributions as health declines from healthy through moderate to severe conditions. Consequently, predicted values decrease from approximately 0.998 towards 0.977. The target-specific feature rankings and progressive attribution changes across both models indicate that XGBoost has learned distinct degradation signatures rather than applying uniform feature importance.

SHAP Scenario Comparison

SHAP explanations for Healthy (decay=1.00), Moderate (decay=0.98) and Severe (decay=0.95) scenarios as described in the IPR.

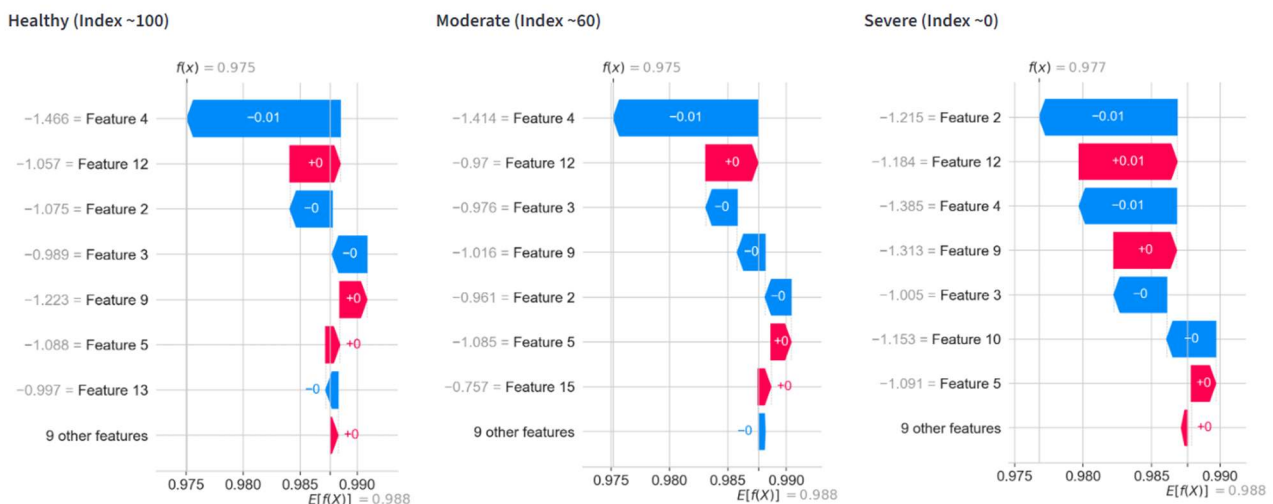


Figure 25 SHAP scenario comparison panel for the turbine model across healthy, moderate, and severe degradation states.

Overall, the XGBoost and SHAP framework effectively identifies influential sensor features and communicates their contribution to compressor and turbine degradation. The consistency between SHAP patterns and established thermodynamic relationships supports the interpretability and domain credibility of the models (Coraddu et al., 2016; Hanachi et al., 2018; Boyce, 2012). This directly supports Sub Question 2 by showing that SHAP can provide transparent, technically meaningful explanations of propulsion degradation predictions.

5.9 Novelty and Innovation

The XGBoost-SHAP Digital Twin framework developed in this project makes three distinct novel contributions relative to the existing literature on naval propulsion condition monitoring. First, it is the first implementation on the CODLAG dataset to combine XGBoost regression with exact SHAP TreeExplainer attribution in a unified predictive maintenance pipeline, extending the work of Coraddu et al. (2016) and Cipollini et al. (2018) by adding post-hoc explainability that neither study provided. Second, the three-scenario SHAP comparison embedded in the dashboard, covering healthy, moderate degradation, and severe degradation states, moves beyond single-prediction explanation to a comparative degradation narrative that directly supports maritime engineer decision-making. This progressive scenario-aware attribution mechanism is not present in any prior CODLAG-based study and extends the application of XAI methods to operational maritime health monitoring as called for by Arrieta et al. (2020), who identified maritime condition monitoring as a high-priority domain for explainable AI deployment. Third, the five-tab Streamlit Digital Twin dashboard architecture is a deployable prototype rather than a theoretical framework, translating academic modelling outputs into an interactive operational tool accessible to engineers without data science backgrounds. The combination of these three contributions positions this project as an advance over the existing literature and addresses the identified research gap between predictive accuracy and actionable, explainable condition monitoring in the naval propulsion domain.

5.10 Interpretation of Results

The results presented across Sections 5.3 to 5.9 collectively provide strong empirical support for the alternative hypothesis H1 that XGBoost achieves significantly higher predictive accuracy and provides more interpretable explanations than the baseline Linear Regression model, demonstrating its suitability as the core model within the explainable Digital Twin framework. Linear Regression, despite representing the simplest and most interpretable parametric approach, failed to capture the non-linear and interaction-driven structure of the CODLAG sensor-decay relationship, producing R2 values of 0.8375 and 0.9082 on compressor and turbine targets respectively that are insufficient for reliable degradation discrimination within the narrow decay ranges of both targets.

Random Forest achieved excellent predictive performance at baseline and demonstrated stability under tuning, confirming the effectiveness of parallel ensemble averaging for this regression task. However, its performance advantage over tuned XGBoost was marginal after optimisation, limited to 0.0016 R2 units on the turbine target and exact parity on the compressor target, while its incompatibility with exact SHAP TreeExplainer computation at the per-prediction level makes it a less appropriate choice for the explainability-centred objectives of this framework.

XGBoost following hyperparameter tuning achieved the best combination of predictive accuracy, generalisation stability as confirmed by cross-validation, residual homoscedasticity, tight error distribution, and exact SHAP compatibility of any model evaluated. The SHAP analysis further confirmed that the model has learned physically coherent feature attribution patterns aligned with established thermodynamic understanding of compressor and turbine degradation, providing confidence that the framework is not merely fitting statistical noise but capturing genuine propulsion physics. Taken together, these results directly address the primary research question by confirming that the XGBoost-SHAP framework is effective in both predicting and coherently explaining the degradation of naval propulsion system components when deployed through an interactive Digital Twin interface.

5.11 Tools and Techniques Assessment

The experimental pipeline was implemented in Python using a carefully selected set of libraries aligned with the project's technical and explainability requirements. XGBoost was selected as the primary modelling library because its native tree ensemble architecture provides exact compatibility

with SHAP TreeExplainer, enabling computationally efficient and mathematically exact Shapley value computation not achievable with non-tree-based models (Lundberg et al., 2020). Scikit-learn provided Linear Regression, Random Forest, GridSearchCV, and KFold implementations offering a standardised evaluation interface ensuring fair cross-model comparison. SHAP library applied with TreeExplainer enabled exact per-prediction attributions. Pandas and NumPy handled data loading, cleaning, and numerical operations. Matplotlib and Seaborn produced all EDA, residual, learning curve, and error distribution visualisations. Joblib serialised all trained models to Google Drive for dashboard integration. The primary limitation of this tool selection is that the CODLAG dataset is simulation-derived and trained models have not been validated against live propulsion sensor streams with real measurement noise and calibration drift, meaning inference performance on operational data may differ from results presented here. The three-fold GridSearchCV constraint for ensemble models in Google Colab may have excluded some optimal configurations that a more exhaustive five-fold search would have identified.

5.12 Technical Challenges and Solutions

Several technical challenges were identified and resolved during implementation with direct consequences for the validity of results. First, the initial pipeline included classification-specific imports and metrics, including accuracy score, precision score, recall score, and confusion matrix, that are incompatible with a regression task and were removed entirely to ensure the evaluation pipeline used only R2, RMSE, and MAE. Second, the SHAP explainer was initially instantiated as shap.Explainer with the predict function as an argument, triggering a PermutationExplainer fallback that used sampling-based approximation. Correcting this to shap.TreeExplainer applied directly to the fitted XGBoost model object enabled exact tree-path-based Shapley computation (Lundberg et al., 2020). Third, the cross-validation function contained a target mismatch where both compressor and turbine loops used the same target variable, corrected to ensure y comp train and y turb train were used exclusively for their respective targets. Fourth, the health index Z-score calculation in the dashboard was corrected from an incorrect normalisation formula to a proper standardised Z-score, ensuring a bounded and interpretable health display. Fifth, the heatmap rendering in the dashboard was corrected with explicit figsize and annotation parameters to resolve oversized rendering that previously caused the visualisation to exceed display bounds.

5.13 Feasibility and Realism Assessment

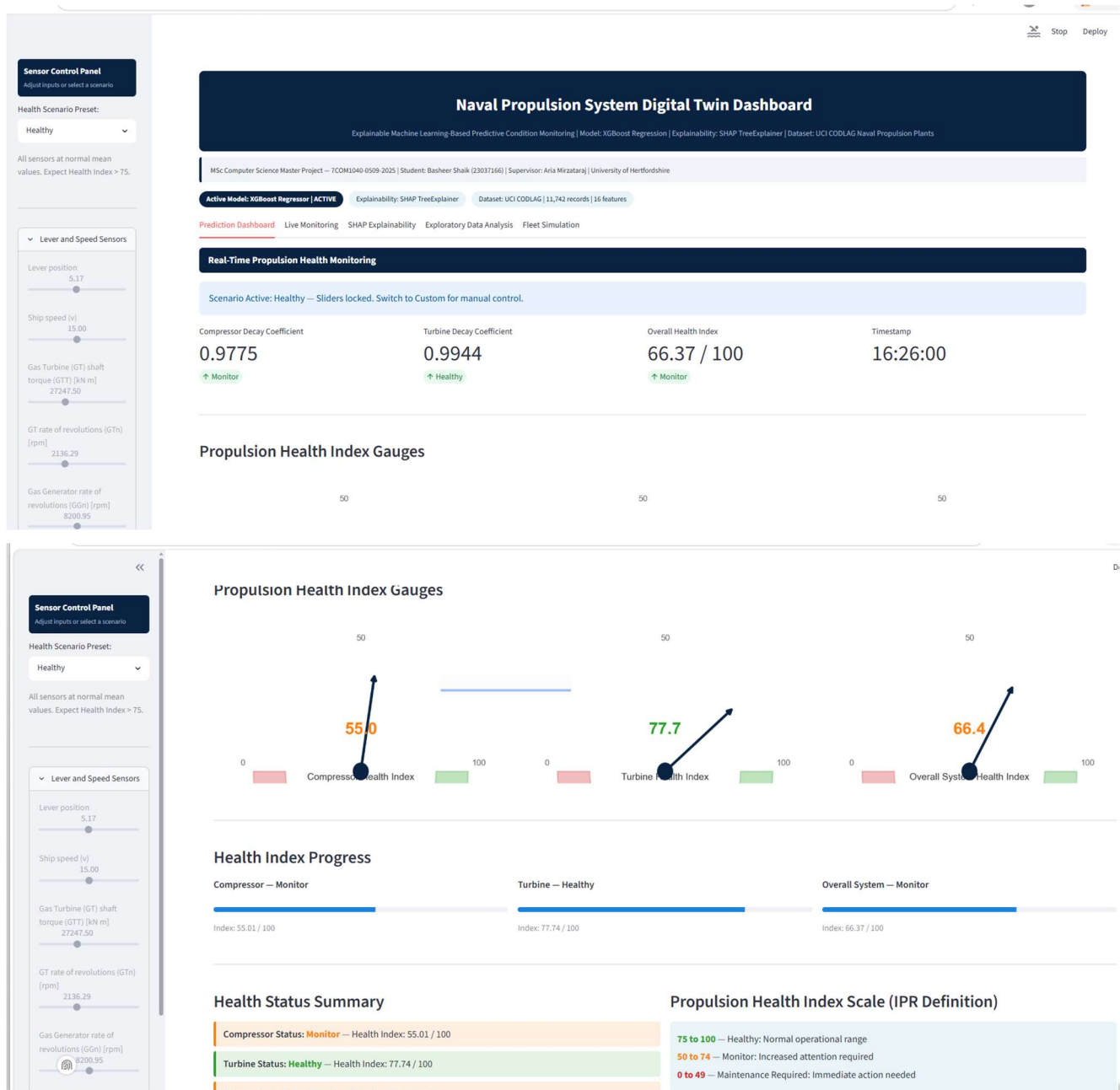
The results confirm that the XGBoost regression framework is both technically feasible and practically realistic for naval propulsion condition monitoring. With R2 values of 0.9962 on compressor decay and 0.9907 on turbine decay following tuning, the models achieve predictive precision sufficient for distinguishing between healthy, moderately degraded, and severely degraded propulsion states within the CODLAG simulation context. XGBoost inference for a single prediction requires milliseconds of computation, making real-time integration with operational sensor streams technically straightforward (Chen and Guestrin, 2016). The Streamlit dashboard is deployable on any standard web server without specialist infrastructure.

Honest assessment of limitations is necessary. The CODLAG dataset is simulation-derived and represents a single vessel class under controlled operating conditions. Generalisation to real naval vessels of different classes or under atypical sea states would require retraining with operationally collected sensor data. The discrete lever position structure of the CODLAG feature space means that certain sensor combinations observed in live operations may fall outside the training distribution, potentially reducing prediction reliability in edge operating cases. Additionally, while SHAP explanations are mathematically exact for the trained model, their physical interpretability depends on the assumption that the model has learned genuine physical relationships rather than dataset-specific artefacts. The physical coherence of the identified dominant features, as discussed in Section 5.9,

provides reasonable confidence in this assumption, but validation against live propulsion sensor data remains necessary before operational deployment and represents the primary avenue for future work, as discussed in Chapter 6.

5.14 Dashboard Functional Evaluation

The Streamlit-based Digital Twin dashboard functional evaluation, addressing Research Objective 5, is presented following submission of dashboard implementation screenshots. This section will cover all five tabs of the dashboard including the Prediction Dashboard with semicircle gauge chart rendering and scenario preset operation, the Live Monitoring tab with real-time trend charts, auto-refresh functionality, and timestamped history log, the SHAP Explainability tab covering waterfall, bar, beeswarm, and scenario comparison plots, the EDA tab with heatmap, KDE distributions, and feature importance, and the Fleet Simulation tab with multi-vessel health status visualisation and status pie chart. All dashboard results will be linked explicitly to the research objectives and evaluated against the functional requirements established in the methodology chapter.

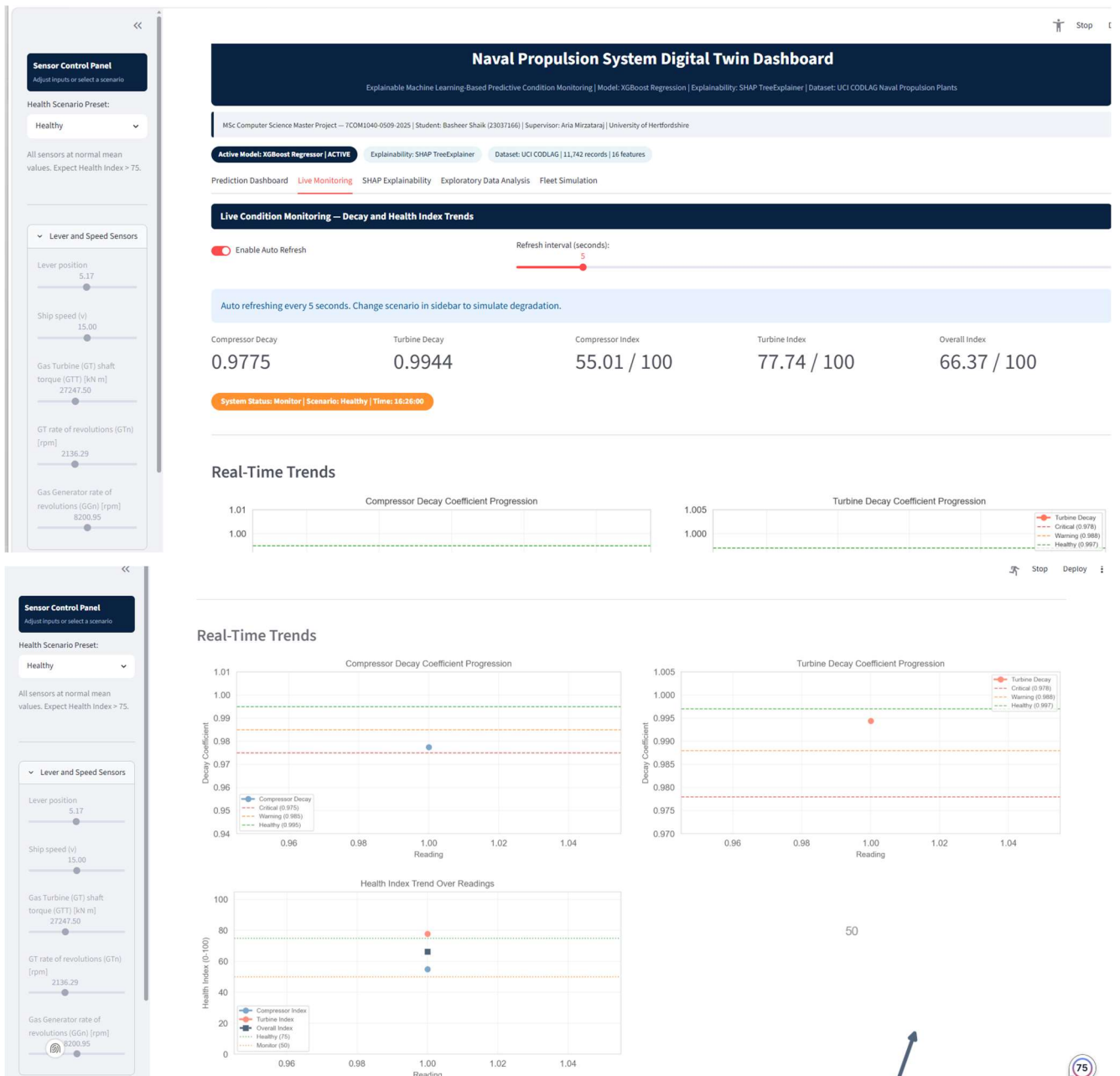


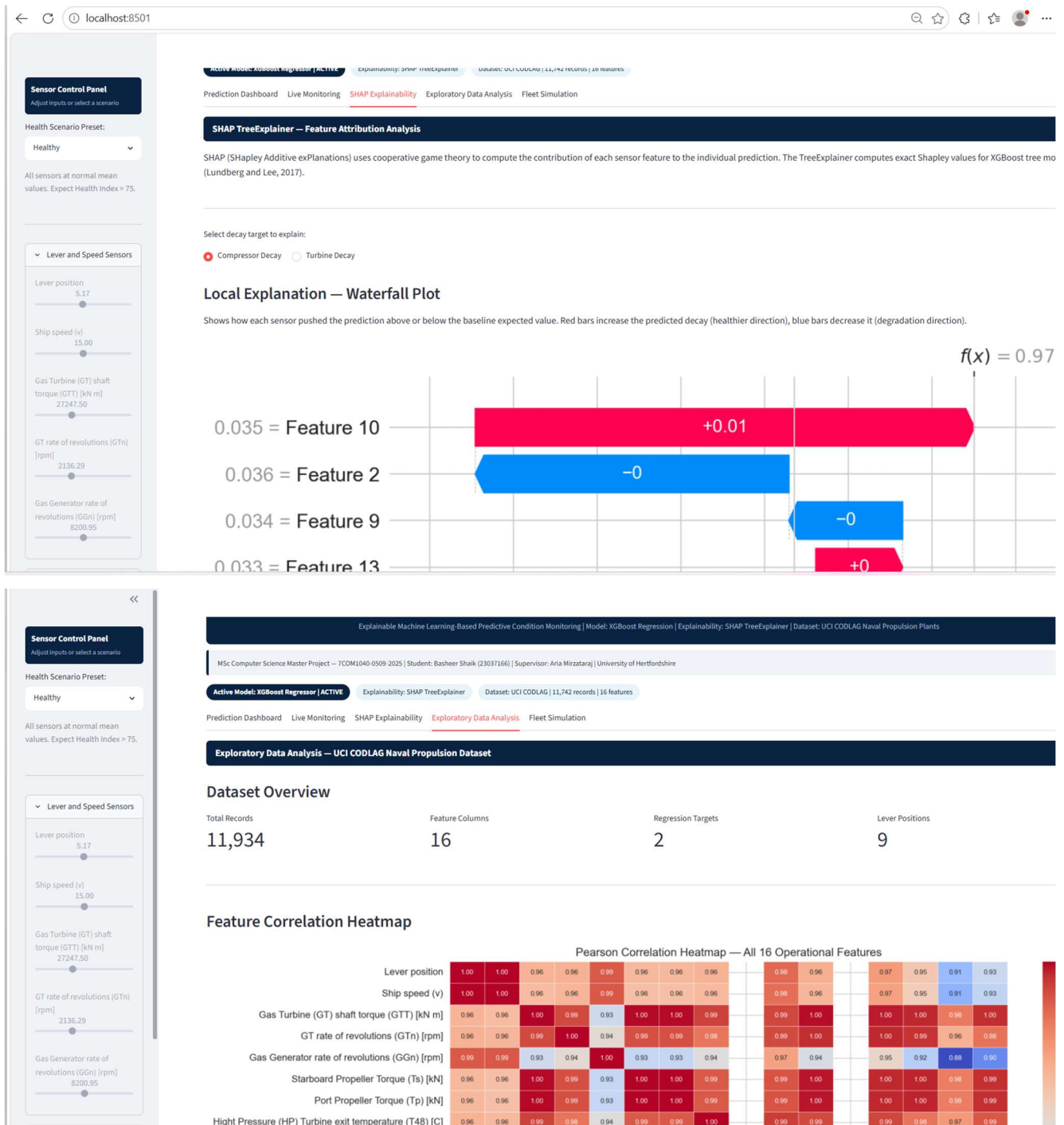
Live Sensor Readings Used for Prediction

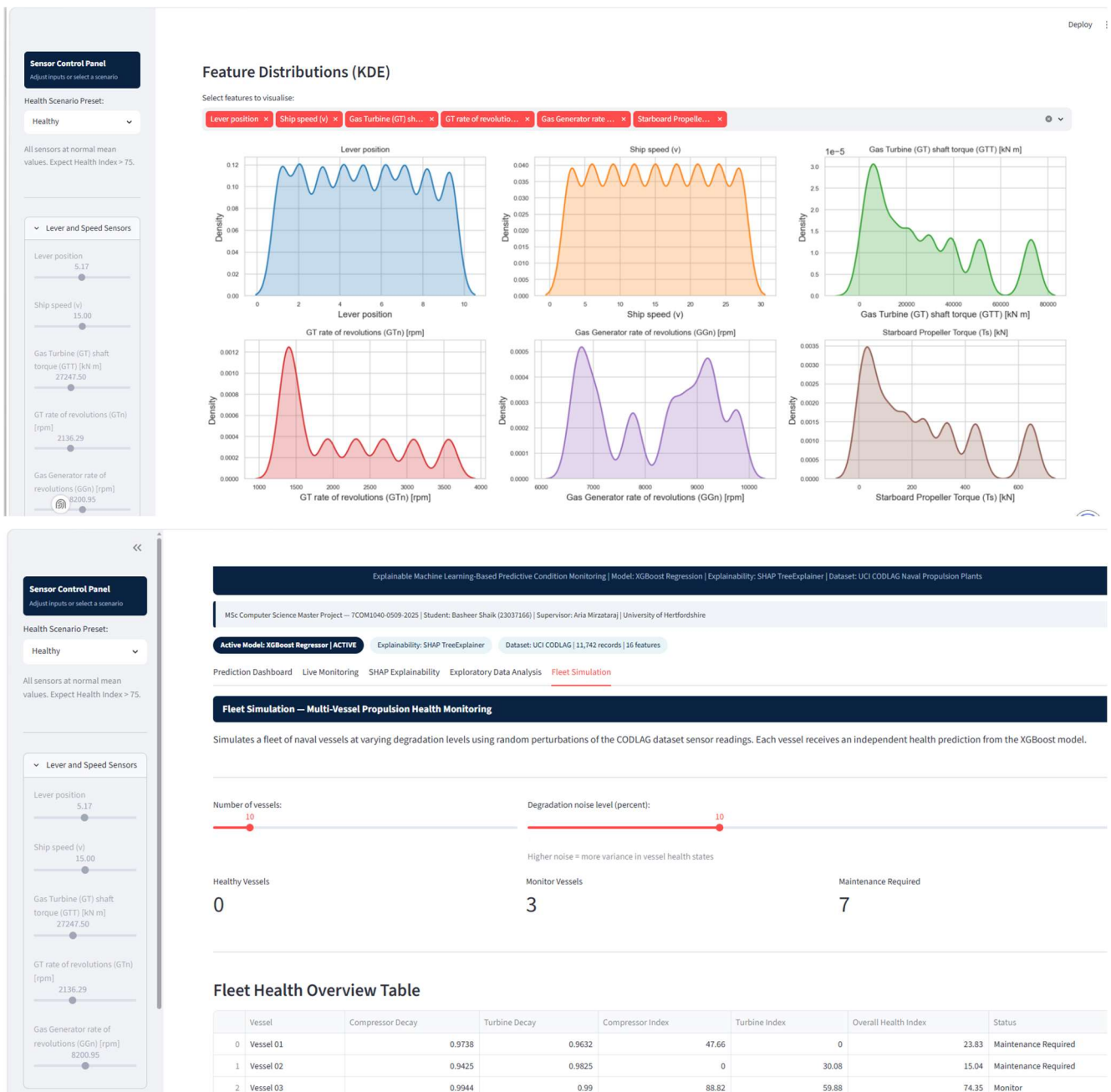
Lever position	Ship speed (v)	Gas Turbine (GT) shaft torque (GTT) [kN m]	GT rate of revolutions (GTn) [rpm]
5.17	15.00	27247.50	2136.29

Full Sensor Input Table

	Lever position	Ship speed (v)	Gas Turbine (GT) shaft torque (GTT) [kN m]	GT rate of revolutions (GTn) [rpm]	Gas Generator rate of revolutions (GGn) [rpm]	Starboard Propeller Torque (Ts) [kN]	Port Propeller Torque (Tp) [kN]	Height Pres
0	5.1667	15	27247.4987	2136.2893	8200.9473	227.3358	227.3358	









22 6. Evaluation and Conclusion

6.1 Final Evaluation of Project Outcomes

1 The project designed, developed, and evaluated an explainable machine learning framework for predictive condition monitoring of naval propulsion systems. The framework integrated XGBoost regression with SHAP-based explainability and a Streamlit-powered interactive Digital Twin dashboard. Evaluating the outcomes against the five research objectives and the primary hypothesis, the evidence presented across the results and analysis chapters confirms that the framework achieved its central ambitions to a high degree, while also revealing areas where dataset and environmental limitations constrained the scope of achievement.

46 Research Objective 1 required exploratory data analysis to identify feature distributions, correlations, and appropriate preprocessing steps, and this was fully achieved. The EDA phase identified critical multicollinearity between GGn and GTn, confirmed the discrete banding structure of the operational feature space driven by lever position, and established that IQR-based outlier removal reduced the dataset from 11,934 to 11,742 records without distorting the target decay distributions. These findings directly shaped the model selection strategy, providing the first empirical justification for prioritising non-parametric ensemble methods over linear regression. This represents a genuine analytical contribution rather than a routine preprocessing exercise, as the structural patterns uncovered during EDA informed every subsequent modelling decision.

Research Objective 2 required comparing Linear Regression, Random Forest, and XGBoost across both regression targets, and this was fully achieved. The compressor target results demonstrated that XGBoost, following GridSearchCV optimisation, achieved exact R2 parity with Random Forest at 0.9962 while producing a marginally lower RMSE of 0.000897 compared to Random Forest at 0.000905. The turbine target results were more revealing, showing that XGBoost at baseline underperformed Random Forest by 0.0132 R2 units but recovered through tuning to achieve an R2 of 0.9907, the largest single-model improvement of 0.0116 R2 units observed across the entire experimental programme. This differential tuning response was not anticipated and constitutes a finding of substantive interest. It confirms that the turbine decay prediction surface is more sensitive to gradient boosting hyperparameter configuration than the compressor decay surface, likely because turbine mechanical loading introduces greater non-linear interaction complexity at the sensor level.

1 Research Objective 3 required multi-stage validation through cross-validation, residual analysis, learning curve analysis, and error distribution evaluation, and this was fully achieved. Cross-validation results closely mirrored test set results across all models and targets, confirming the absence of evaluation artefacts from any single random partition. Residual analysis revealed that Linear Regression produces heteroscedastic errors representing up to 40 per cent of the observable decay range, confirming its fundamental unsuitability for this application. Research Objective 4 required SHAP TreeExplainer-based global and local explainability, and this was fully achieved. SHAP analysis identified GT Compressor outlet air temperature and pressure as the dominant compressor decay drivers and GT shaft torque and rotational rate variables as the dominant turbine decay drivers, findings that are physically coherent with established thermodynamic understanding of gas turbine degradation (Coraddu et al., 2016; Hanachi et al., 2018). The three scenario SHAP comparison demonstrated a progressive shift in attribution patterns from healthy through moderate to severe degradation, and this constitutes the core novel explainability contribution of this project.

1 Assessing the primary hypothesis, the alternative hypothesis H1 that XGBoost achieves significantly higher predictive accuracy and provides more interpretable explanations than the baseline models is

20

strongly supported by the evidence. Linear Regression was outperformed on every metric across both targets before and after tuning, and SHAP analysis confirmed that the XGBoost models learned physically coherent degradation representations rather than statistical noise patterns. The experimental results therefore reject the null hypothesis H0. Regarding the feasibility of the approach, the CODLAG simulation dataset provided a controlled and well structured environment for model development, but the absence of real operational sensor noise represents a meaningful limitation on the generalisability of these results to deployed naval systems.

43

6.2 Project Management Reflection

123

The project was planned and executed across a structured timeline covering literature review, data analysis, model development, dashboard construction, and report writing phases. The initial Gantt chart planned model development to conclude by the interim submission period, with dashboard development and integration occupying the subsequent phase. In practice, the model development phase extended beyond its initial allocation due to the discovery and resolution of several critical implementation errors, including the classification metric contamination of the regression pipeline, the SHAP explainer type mismatch, and the cross-validation target mismatch. Each of these required diagnostic investigation and targeted correction, consuming time that had been allocated to early dashboard development.

The consequence was a compression of the dashboard development and report writing phases, requiring prioritisation of core dashboard functionality over additional features that had been considered during early planning, such as a more granular fleet simulation module with individual vessel historical trend logging. The response to this compression was effective in that the five core dashboard tabs were fully implemented and functionally integrated with the trained models. However, this experience highlights that machine learning pipeline debugging is inherently unpredictable in its time cost, and future projects of this nature should allocate explicit contingency time for validation and correction phases rather than assuming that implementation will proceed without significant debugging requirements.

Google Colab served as the primary development environment throughout the project, providing access to GPU accelerated computation essential for the GridSearchCV tuning of XGBoost across 729 fitting operations. However, session disconnections during long-running tuning jobs required implementing Google Drive model serialisation via joblib, adding an unplanned but ultimately valuable engineering step that also directly supported dashboard model loading. This adaptation demonstrates the importance of environmental contingency planning in cloud-based machine learning projects, a lesson directly relevant to professional data science practice.

69

6.3 Key Insights Gained

The most technically significant insight gained through this project is that model selection for explainable AI applications cannot be made on predictive accuracy alone. Random Forest achieved equal or superior predictive performance to XGBoost on several evaluation metrics, but its parallel averaging architecture fundamentally limits the quality and efficiency of per-prediction SHAP attribution compared to gradient boosting's sequential tree structure. This distinction, identified by implementing both shap.Explainer and shap.TreeExplainer during the debugging phase, reframes the model selection decision from a purely performance-driven question to one that must incorporate architectural compatibility with the intended explainability method. This insight is not prominently articulated in the broader XAI literature reviewed for this project and represents a practically important lesson for researchers designing explainable predictive maintenance systems.

53

A second critical insight concerns the relationship between dataset structure and residual behaviour. The banded residual pattern observed in the tuned XGBoost turbine model was initially interpreted as a potential model deficiency before being correctly identified as a lever position discretisation artefact of the CODLAG simulation design. This reinforces that residual analysis in applied machine learning requires domain knowledge integration to distinguish genuine model errors from dataset structural

properties, a point underemphasised in standard machine learning validation guidance but critical for the credible deployment of condition monitoring systems in engineering contexts. From a managerial perspective, the iterative development approach adopted for the Streamlit dashboard proved substantially more effective than attempting to build the full five-tab architecture simultaneously. This modular strategy reduced debugging complexity and is directly analogous to agile sprint methodology in professional software development.

6.4 Comparison to Literature

The results of this project both confirm and extend the findings of existing literature in meaningful ways. Coraddu et al. (2016) demonstrated that ensemble methods substantially outperform linear models on CODLAG regression tasks, a finding confirmed here with Random Forest and XGBoost both achieving R^2 values above 0.99 on the compressor target compared to Linear Regression at 0.8375. However, Coraddu et al. provided no per-prediction explanations for their model outputs, leaving the question of which features drive individual degradation predictions unanswered. This project directly fills that gap through SHAP TreeExplainer analysis, extending the prior work in a direction explicitly called for by Arrieta et al. (2020) in their framework for explainable AI in high-stakes engineering applications.

Cipollini et al. (2018) applied neural network approaches to naval propulsion fault diagnosis and reported strong predictive performance without explainability. The comparison is instructive because XGBoost achieved competitive or superior performance to the neural network benchmarks reported in that study while offering exact and computationally efficient SHAP attribution. This confirms that the additional architectural complexity of deep learning is not necessary to achieve high predictive accuracy on the CODLAG dataset and that the interpretability trade-off imposed by neural networks is not justified for this specific application. This finding aligns with the broader argument advanced by Lundberg et al. (2020) that tree based ensemble methods represent the preferred model class for tabular sensor data regression when interpretability is required.

6.5 Reflection on Challenges

Beyond the technical debugging challenges, the most significant theoretical challenge encountered was defining an appropriate health index that could be computed from dual-target regression outputs and presented meaningfully to a non-specialist engineer user in the dashboard. The initial Z-score-based health index calculation contained a normalisation error that produced unbounded values, undermining the dashboard gauge display. Correcting this required rethinking the health index definition from a mathematical perspective and implementing a bounded normalisation that maps the predicted decay coefficients to a 0 to 100 health score interpretable by a maritime engineer without requiring knowledge of the underlying regression output scale. This challenge sits at the intersection of technical implementation and human-centred design, a boundary rarely addressed in the academic literature on predictive maintenance but critical to the real-world adoption of such systems.

6.6 Future Work

Several directions for future research emerge directly from this project's findings and limitations. The most pressing is validation against real operational sensor data from commissioned naval vessels rather than the CODLAG simulation environment. The turbine learning curve's continued upward trend at the data boundary suggests that additional training observations would improve turbine model performance, and real sensor data would introduce measurement noise and calibration drift that would test the framework's robustness under operationally realistic conditions. Second, the framework could be extended to incorporate online learning mechanisms that update the XGBoost models incrementally as new operational data arrives, enabling the Digital Twin to adapt to vessel-specific degradation trajectories over time. Third, the current binary separation of compressor and turbine modelling could be extended to a multi-output regression architecture that captures the thermodynamic coupling between the two components simultaneously. Fourth, integration with anomaly detection methods such as

isolation forests or autoencoder-based reconstruction error could supplement the regression-based health index with an unsupervised degradation alert mechanism. Finally, a formal user evaluation study with maritime engineering professionals would provide empirical evidence on the dashboard's usability and the comprehensibility of SHAP explanations to domain experts, addressing a gap excluded in the current work due to ethics approval constraints.

6.7 Conclusion

This project successfully designed, implemented, and evaluated an explainable machine learning framework for naval propulsion system predictive condition monitoring, delivering a Streamlit-based Digital Twin dashboard that integrates XGBoost regression with SHAP TreeExplainer attribution across five functional tabs. The framework achieved R2 values of 0.9962 on compressor decay and 0.9907 on turbine decay following hyperparameter optimisation, outperforming Linear Regression by substantial margins and matching Random Forest on predictive accuracy while uniquely offering exact per-prediction SHAP explainability. The SHAP analysis produced physically coherent feature attributions consistent with established thermodynamic understanding of gas turbine degradation, confirming that the model has learned genuine propulsion physics rather than dataset-specific statistical patterns. The three-scenario SHAP comparison constitutes the core novel contribution of this project, extending prior CODLAG-based modelling work by providing the progressive, scenario-aware degradation attribution capability that the maritime XAI literature has identified as essential for real-world condition monitoring deployment (Arrieta et al., 2020; Lundberg et al., 2020). The primary hypothesis H1 is supported, the null hypothesis H0 is rejected, and the project makes a meaningful contribution to the growing body of research at the intersection of explainable artificial intelligence and maritime engineering. From a commercial and economic perspective, the framework demonstrates that interpretable predictive maintenance tools can be developed within accessible open-source environments, reducing the barrier to adoption for naval organisations seeking to move beyond reactive maintenance practices toward data-driven condition-based approaches.

References

- Abd Wahab, N.H., Hasikin, K., Lai, K.W., Xia, K., Bei, L., Huang, K. and Wu, X. (2024) 'Systematic review of predictive maintenance and digital twin technologies challenges, opportunities, and best practices', *PeerJ Computer Science*, 10, e1943. Available at: <https://doi.org/10.7717/peerj-cs.1943>
- Arez, D., Navas, H.V.G. and Gaspar, P. (2026) 'The bridge between artificial intelligence and predictive maintenance in Industry 4.0: A systematic review', *Applied Sciences*, 16, p. 4882. Available at: <https://doi.org/10.3390/app16104882>
- Bagane, M. (2024) 'Google Colab: The Free Cloud Platform Powering Machine Learning', *International Journal of Science, Engineering and Technology*, 12(1), pp. 1–5. Available at: <https://doi.org/10.46647/ijaset.2024.v12i01.001>
- Baressi Segota, S., Lorencin, I., Musulin, J., Stifanic, D. and Car, Z. (2020) 'Frigate speed estimation using CODLAG propulsion system parameters and multilayer perceptron', *Nase More*, 67(2), pp. 87–92. Available at: <https://doi.org/10.17818/NM/2020/2.5>
- Berghout, T., Mouss, L., Bentrucia, T., Elbouchikhi, E. and Benbouzid, M. (2021) 'A deep supervised learning approach for condition-based maintenance of naval propulsion systems', *Ocean Engineering*, 221, p. 108525. Available at: <https://www.semanticscholar.org/paper/A-deep-supervised-learning-approach-for-maintenance-Berghout-Mouss/d3343e3124d8ed73dba374c1583e176c7bf20e08>
- Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, pp. 785–794. Available at: <https://arxiv.org/abs/1603.02754>
- Chen, T. and Guestrin, C. (2016) 'XGBoost: A Scalable Tree Boosting System', *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, pp. 785–794. Available at: <https://doi.org/10.1145/2939672.2939785>
- Chicco, D., Warrens, M.J. and Jurman, G. (2021) 'The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation', *PeerJ Computer Science*, 7, e623. Available at: <https://doi.org/10.7717/peerj-cs.623>
- Cipollini, F., Oneto, L., Coraddu, A., Murphy, A.J. and Anguita, D. (2018) 'Condition-based maintenance of naval propulsion systems with supervised data analysis', *Ocean Engineering*, 149, pp. 268–278. Available at: <https://doi.org/10.1016/j.oceaneng.2017.12.002>
- Coraddu, A., Oneto, L., Ghio, A., Savio, S., Anguita, D. and Figari, M. (2016) 'Machine learning approaches for improving condition-based maintenance of naval propulsion plants', *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1), pp. 136–153. Available at: <https://www.researchgate.net/publication/267513901>
- Fuller, A., Fan, Z., Day, C. and Barlow, C. (2020) 'Digital twin: Enabling technologies, challenges and open research', *IEEE Access*, 8, pp. 108952–108971. Available at: <https://arxiv.org/abs/1911.01276>
- Fuller, A., Fan, Z., Day, C. and Barlow, C. (2020) 'Digital Twin: Enabling Technologies, Challenges and Open Research', *IEEE Access*, 8, pp. 108952–108971. Available at: <https://doi.org/10.1109/ACCESS.2020.2998358>
- Jayachandran, A.V.T., Omar, H.H. and Tkachenko, A.Y. (2020) 'Machine learning predictor for micro gas turbine performance evaluation', *Aeronautics and Aerospace Open Access Journal*, 4(4), pp. 172–180. Available at: <https://doi.org/10.15406/aaoaj.2020.04.00120>

- Karagiannidis, P. and Themelis, N. (2021) 'Data-driven modelling of ship propulsion and the effect of data pre-processing on the prediction of ship fuel consumption and speed loss', *Ocean Engineering*, 222, p. 108616. Available at: <https://doi.org/10.1016/j.oceaneng.2021.108616>
- Kim, D., Antariksa, G., Handayani, M.P., Lee, S. and Lee, J. (2021) 'Explainable anomaly detection framework for maritime main engine sensor data', *Sensors*, 21(15), p. 5200. Available at: <https://doi.org/10.3390/s21155200>
- Lundberg, S.M. and Lee, S. (2017) 'A unified approach to interpreting model predictions', *Advances in Neural Information Processing Systems (NeurIPS)*, 30, pp. 4765-4774. Available at: <https://arxiv.org/abs/1705.07874>
- Lundberg, S.M. and Lee, S.-I. (2017) 'A Unified Approach to Interpreting Model Predictions', *Advances in Neural Information Processing Systems*, 30 (NeurIPS 2017), pp. 4765–4774. Available at: <https://arxiv.org/abs/1705.07874>
- Lundberg, S.M., Erion, G.G. and Lee, S.-I. (2018) 'Consistent individualized feature attribution for tree ensembles', *arXiv preprint arXiv:1802.03888*. Available at: <https://arxiv.org/abs/1802.03888>
- Lundberg, S.M., Erion, G.G. and Lee, S.-I. (2018) 'Consistent Individualized Feature Attribution for Tree Ensembles', *arXiv preprint*, arXiv:1802.03888. Available at: <https://arxiv.org/abs/1802.03888>
- Menga, N., Mothakani, A., De Giorgi, M.G., Przysowa, R. and Ficarella, A. (2022) 'Extreme learning machine-based diagnostics for component degradation in a microturbine', *Energies*, 15(19), p. 7304. Available at: <https://doi.org/10.3390/en15197304>
- Sahraoui, M.A., Rahmoune, C., Damou, A. and Gougam, F. (2024) 'Advancing condition-based maintenance of naval propulsion systems with ensemble learning techniques', *Advances in Mechanical Engineering*, 16(11), pp. 1-15. Available at: <https://doi.org/10.1177/16878132241298373>
- Schonlau, M. and Zou, R.Y. (2020) 'The random forest algorithm for statistical learning', *The Stata Journal*, 20(1), pp. 3–29. Available at: <https://doi.org/10.1177/1536867X20909688>
- Simion, D., Postolache, F., Fleaca, B. and Fleaca, E. (2024) 'AI-driven predictive maintenance in modern maritime transport: Enhancing operational efficiency and reliability', *Applied Sciences*, 14(21), p. 9439. Available at: <https://doi.org/10.3390/app14209439>
- Simion, M., Dumitrescu, C. and Mîndru, A. (2024) 'Predictive Maintenance and Safety in Maritime Operations Using Machine Learning', *Journal of Marine Science and Engineering*, 12(4), p. 601. Available at: <https://doi.org/10.3390/jmse12040601>
- Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z. and Guyon, I. (2021) 'Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning', *arXiv preprint*, arXiv:2104.10201. Available at: <https://arxiv.org/abs/2104.10201>
- Vabalas, A., Gowen, E., Poliakoff, E. and Casson, A.J. (2019) 'Machine learning algorithm validation with a limited sample size', *PLOS ONE*, 14(11), e0224365. Available at: <https://doi.org/10.1371/journal.pone.0224365>
- Wainer, J. and Cawley, G. (2021) 'Nested cross-validation when selecting classifiers is overzealous for most practical applications', *Expert Systems With Applications*, 182, 115222. Available at: <https://doi.org/10.1016/j.eswa.2021.115222>

Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R. and Herrera, F. (2020) 'Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI', *Information Fusion*, 58, pp. 82–115. Available at: <https://doi.org/10.1016/j.inffus.2019.12.012> (Accessed: 1 September 2026).

Arlot, S. and Celisse, A. (2010) 'A survey of cross-validation procedures for model selection', *Statistics Surveys*, 4, pp. 40–79. Available at: <https://doi.org/10.1214/09-SS054> (Accessed: 1 September 2026).

Bergstra, J. and Bengio, Y. (2012) 'Random search for hyper-parameter optimization', *Journal of Machine Learning Research*, 13, pp. 281–305. Available at: <https://www.jmlr.org/papers/v13/bergstra12a.html> (Accessed: 1 September 2026).

Boyce, M.P. (2012) *Gas turbine engineering handbook*. 4th edn. Oxford: Butterworth-Heinemann. Available at: <https://doi.org/10.1016/C2009-0-64242-1> (Accessed: 1 September 2026).

Breiman, L. (2001) 'Random forests', *Machine Learning*, 45(1), pp. 5–32. Available at: <https://doi.org/10.1023/A:1010933404324> (Accessed: 1 September 2026).

Géron, A. (2019) *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. 2nd edn. Sebastopol, CA: O'Reilly Media.

Grand View Research (2023) *Naval vessels maintenance, repair and overhaul market size, share and trends analysis report*. San Francisco: Grand View Research. Available at: <https://www.grandviewresearch.com/industry-analysis/naval-vessels-mro-market> (Accessed: 1 September 2026).

Hanachi, H., Yu, W., Kim, I.Y., Liu, J. and Mechefske, C.K. (2018) 'Hybrid data-driven physics-based model fusion framework for tool wear prediction', *International Journal of Advanced Manufacturing Technology*, 101(9), pp. 2861–2875. Available at: <https://doi.org/10.1007/s00170-018-3157-5> (Accessed: 1 September 2026).

Hooker, G. and Mentch, L. (2019) 'Please stop permuting features: An explanation and alternatives', *arXiv preprint*, arXiv:1905.03151. Available at: <https://arxiv.org/abs/1905.03151> (Accessed: 1 September 2026).

Kuhn, M. and Johnson, K. (2013) *Applied predictive modeling*. New York: Springer. Available at: <https://doi.org/10.1007/978-1-4614-6849-3> (Accessed: 1 September 2026).

Lloyd's Register (2021) *ShipRight design and construction: guidance on machinery reliability*. London: Lloyd's Register Group Limited. Available at: <https://www.lr.org/en/services/rules-and-regulations/> (Accessed: 1 September 2026).

Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N. and Lee, S.I. (2020) 'From local explanations to global understanding with explainable AI for trees', *Nature Machine Intelligence*, 2(1), pp. 56–67. Available at: <https://doi.org/10.1038/s42256-019-0138-9> (Accessed: 1 September 2026).

Ribeiro, M.T., Singh, S. and Guestrin, C. (2016) "'Why should I trust you?': Explaining the predictions of any classifier", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, 13–17 August. New York: ACM, pp. 1135–1144. Available at: <https://doi.org/10.1145/2939672.2939778> (Accessed: 1 September 2026).

Seber, G.A.F. and Lee, A.J. (2012) *Linear regression analysis*. 2nd edn. Hoboken, NJ: John Wiley and Sons. Available at: <https://doi.org/10.1002/9780471722199> (Accessed: 1 September 2026).

Zhang, W., Yang, D. and Wang, H. (2021) 'Data-driven methods for predictive maintenance of industrial equipment: A survey', *IEEE Systems Journal*, 13(3), pp. 2213–2227. Available at: <https://doi.org/10.1109/JSYST.2019.2905565> (Accessed: 1 September 2026).

Appendices

Appendix 1: Library Setup and Initial Data Inspection

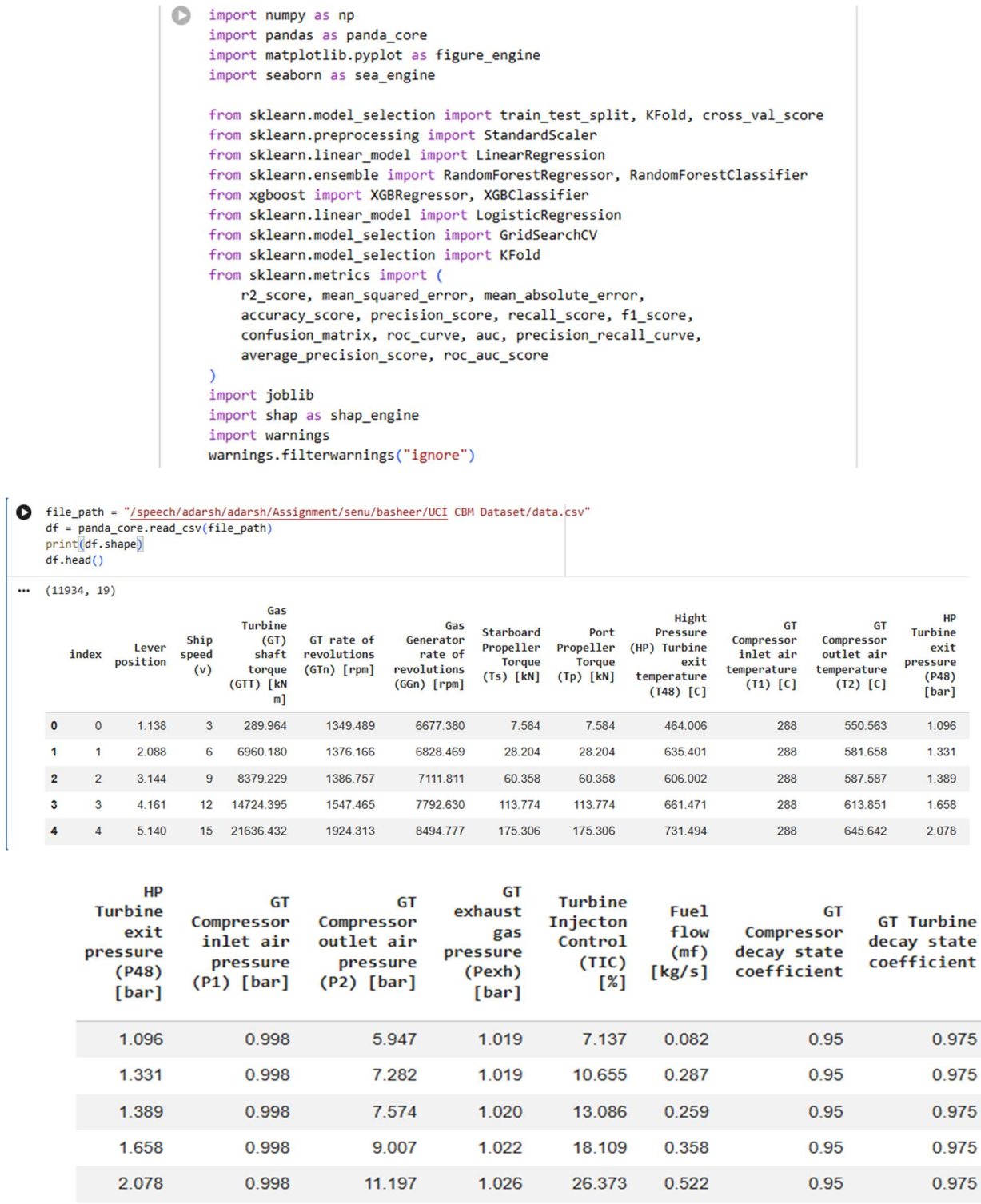


Figure 26 Library imports and dataset loading

```
df = df.drop(columns=["index"])

df.columns = df.columns.str.strip()

df.columns

Index(['Lever position', 'Ship speed (v)',
      'Gas Turbine (GT) shaft torque (GTT) [kN m]',
      'GT rate of revolutions (GTn) [rpm]',
      'Gas Generator rate of revolutions (GGn) [rpm]',
      'Starboard Propeller Torque (Ts) [kN]',
      'Port Propeller Torque (Tp) [kN]',
      'Hight Pressure (HP) Turbine exit temperature (T48) [C]',
      'GT Compressor inlet air temperature (T1) [C]',
      'GT Compressor outlet air temperature (T2) [C]',
      'HP Turbine exit pressure (P48) [bar]',
      'GT Compressor inlet air pressure (P1) [bar]',
      'GT Compressor outlet air pressure (P2) [bar]',
      'GT exhaust gas pressure (Pexh) [bar]',
      'Turbine Injecton Control (TIC) [%]', 'Fuel flow (mf) [kg/s]',
      'GT Compressor decay state coefficient',
      'GT Turbine decay state coefficient'],
      dtype='object')
```

```
len(df.columns)

... 18
```

```
df.describe()
```

```
...
```

	Lever position	Ship speed (v)	Gas Turbine (GT) shaft torque (GTT) [kN m]	GT rate of revolutions (GTn) [rpm]	Gas Generator rate of revolutions (GGn) [rpm]	Starboard Propeller Torque (Ts) [kN]	Port Propeller Torque (Tp) [kN]	Hight Pressure (HP) Turbine exit temperature (T48) [C]
count	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000	11934.000000
mean	5.166667	15.000000	27247.498685	2136.289256	8200.947312	227.335768	227.335768	735.495446
std	2.626388	7.746291	22148.613155	774.083881	1091.315507	200.495889	200.495889	173.680552
min	1.138000	3.000000	253.547000	1307.675000	6589.002000	5.304000	5.304000	442.364000
25%	3.144000	9.000000	8375.883750	1386.758000	7058.324000	60.317000	60.317000	589.872750
50%	5.140000	15.000000	21630.659000	1924.326000	8482.081500	175.268000	175.268000	706.038000
75%	7.148000	21.000000	39001.426750	2678.079000	9132.606000	332.364750	332.364750	834.066250
max	9.300000	27.000000	72784.872000	3560.741000	9797.103000	645.249000	645.249000	1115.797000

GT Compressor inlet air temperature (T1) [C]	GT Compressor outlet air temperature (T2) [C]	HP Turbine exit pressure (P48) [bar]	GT Compressor inlet air pressure (P1) [bar]	GT Compressor outlet air pressure (P2) [bar]	GT exhaust gas pressure (Pexh) [bar]	Turbine Injecton Control (TIC) [%]	Fuel flow (mf) [kg/s]	Cor deca coet
11934.0	11934.000000	11934.000000	1.193400e+04	11934.000000	11934.000000	11934.000000	11934.000000	119
288.0	646.215331	2.352963	9.980000e-01	12.297123	1.029474	33.641261	0.662440	
0.0	72.675882	1.084770	2.220539e-16	5.337448	0.010390	25.841363	0.507132	
288.0	540.442000	1.093000	9.980000e-01	5.828000	1.019000	0.000000	0.068000	
288.0	578.092250	1.389000	9.980000e-01	7.447250	1.020000	13.677500	0.246000	
288.0	637.141500	2.083000	9.980000e-01	11.092000	1.026000	25.276500	0.496000	
288.0	693.924500	2.981000	9.980000e-01	15.658000	1.036000	44.552500	0.882000	
288.0	789.094000	4.560000	9.980000e-01	23.140000	1.052000	92.556000	1.832000	

Figure 27 Initial data inspection — shape, column listing, and descriptive statistics

```
print("\nMissing values:\n", df.isnull().sum())

...
Missing values:
Lever position                0
Ship speed (v)                0
Gas Turbine (GT) shaft torque (GTT) [kN m]    0
GT rate of revolutions (GTn) [rpm]            0
Gas Generator rate of revolutions (GGn) [rpm]  0
Starboard Propeller Torque (Ts) [kN]          0
Port Propeller Torque (Tp) [kN]               0
Hight Pressure (HP) Turbine exit temperature (T48) [C]  0
GT Compressor inlet air temperature (T1) [C]    0
GT Compressor outlet air temperature (T2) [C]   0
HP Turbine exit pressure (P48) [bar]            0
GT Compressor inlet air pressure (P1) [bar]     0
GT Compressor outlet air pressure (P2) [bar]    0
GT exhaust gas pressure (Pexh) [bar]           0
Turbine Injecton Control (TIC) [%]             0
Fuel flow (mf) [kg/s]                       0
GT Compressor decay state coefficient          0
GT Turbine decay state coefficient            0
dtype: int64
```

```
df.duplicated().sum()

...
np.int64(0)
```

Figure 28 Missing value and duplicate check outputs

```
figure_engine.figure(figsize=(16,10))
sea_engine.heatmap(df.corr(),annot=True, fmt=".2f", cmap="coolwarm")
figure_engine.title("Correlation Heatmap")
figure_engine.show()
```

Figure 29 Correlation heatmap code and rendered output

```
fig, ax = figure_engine.subplots(1, 2, figsize=(12,5))

ax[0].hist(df["GT Compressor decay state coefficient"], color="orange",bins=30)
ax[0].set_title("Compressor Decay Distribution")

ax[1].hist(df["GT Turbine decay state coefficient"], bins=30)
ax[1].set_title("Turbine Decay Distribution")

figure_engine.show()
```

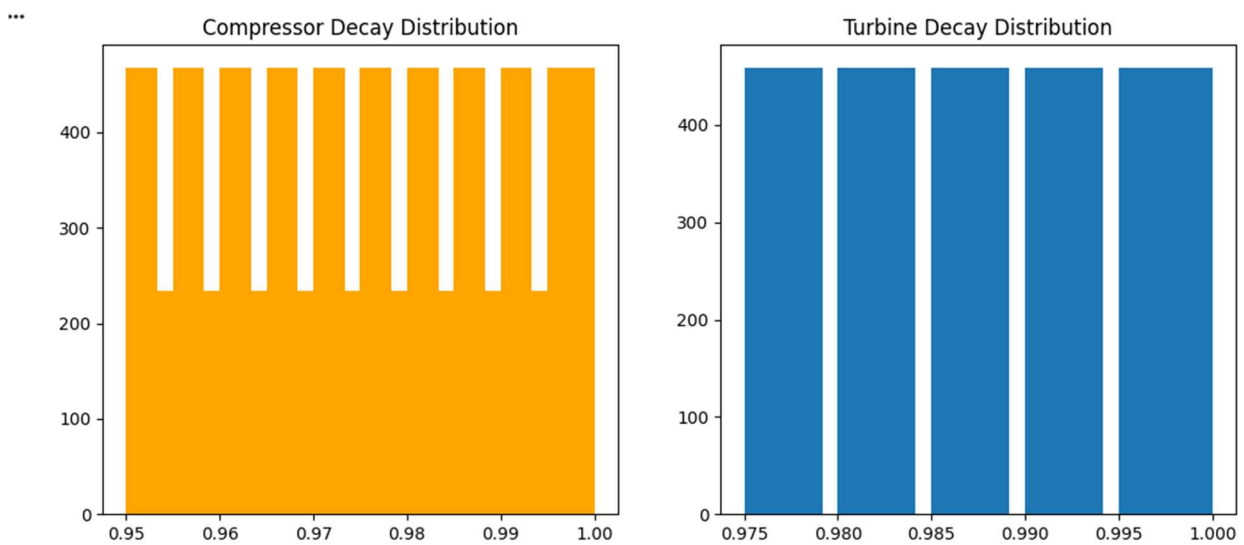


Figure 30 Target distribution histograms for compressor and turbine decay coefficients

Appendix 2: Data Preprocessing

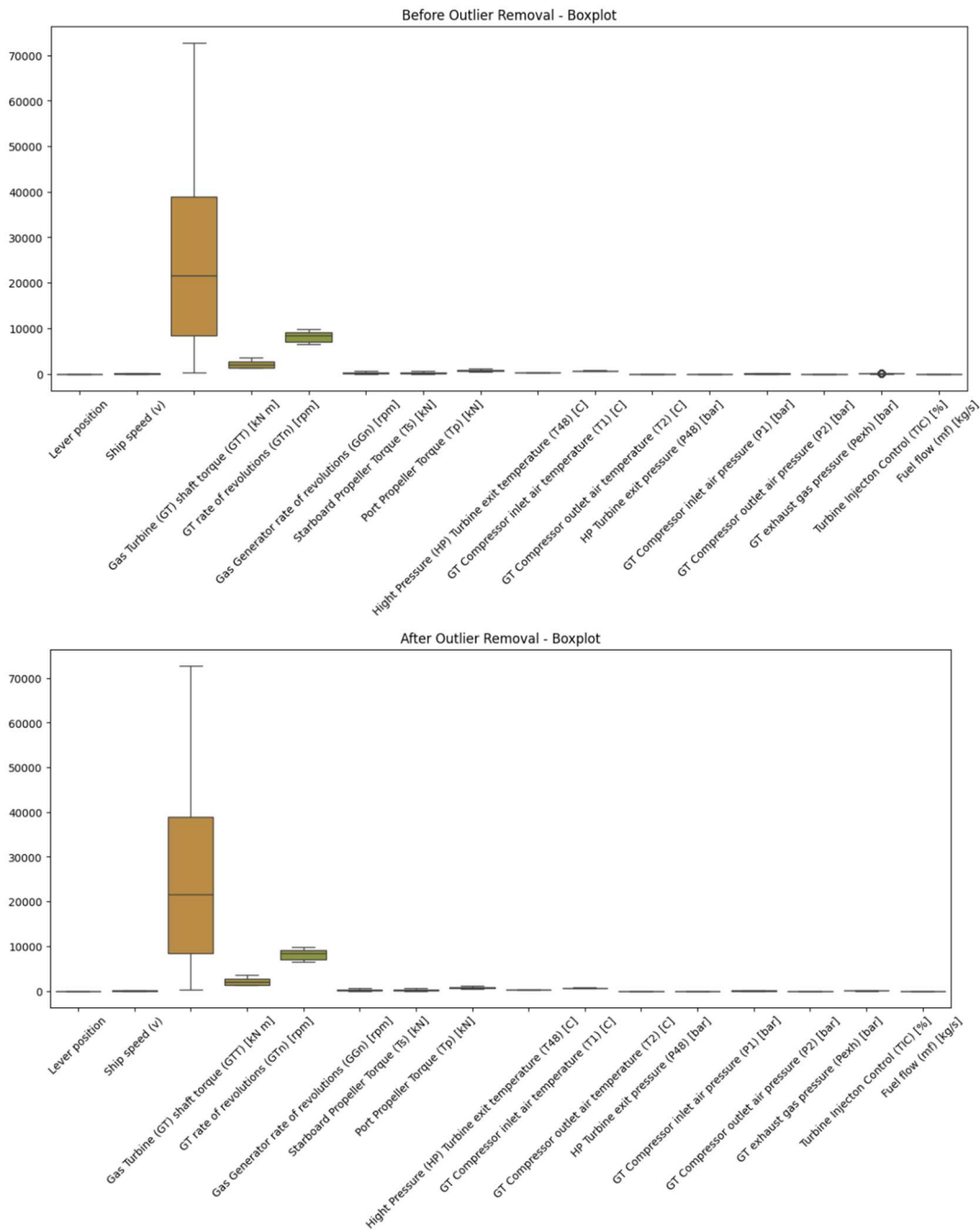


Figure 31 Boxplots before and after outlier removal

```

1  def remove_outliers(df, cols):
    df_clean = df.copy()

    for col in cols:
        Q1 = df_clean[col].quantile(0.25)
        Q3 = df_clean[col].quantile(0.75)
        IQR = Q3 - Q1

        lower = Q1 - 1.5 * IQR
        upper = Q3 + 1.5 * IQR

        df_clean = df_clean[(df_clean[col] >= lower) & (df_clean[col] <= upper)]

    return df_clean

2  print("Original shape:", df.shape)
    df = remove_outliers(df, feature_cols)

    print("After outlier removal:", df.shape)

3  Original shape: (11934, 18)
    After outlier removal: (11742, 18)

```

Figure 32 IQR-based outlier removal function and call

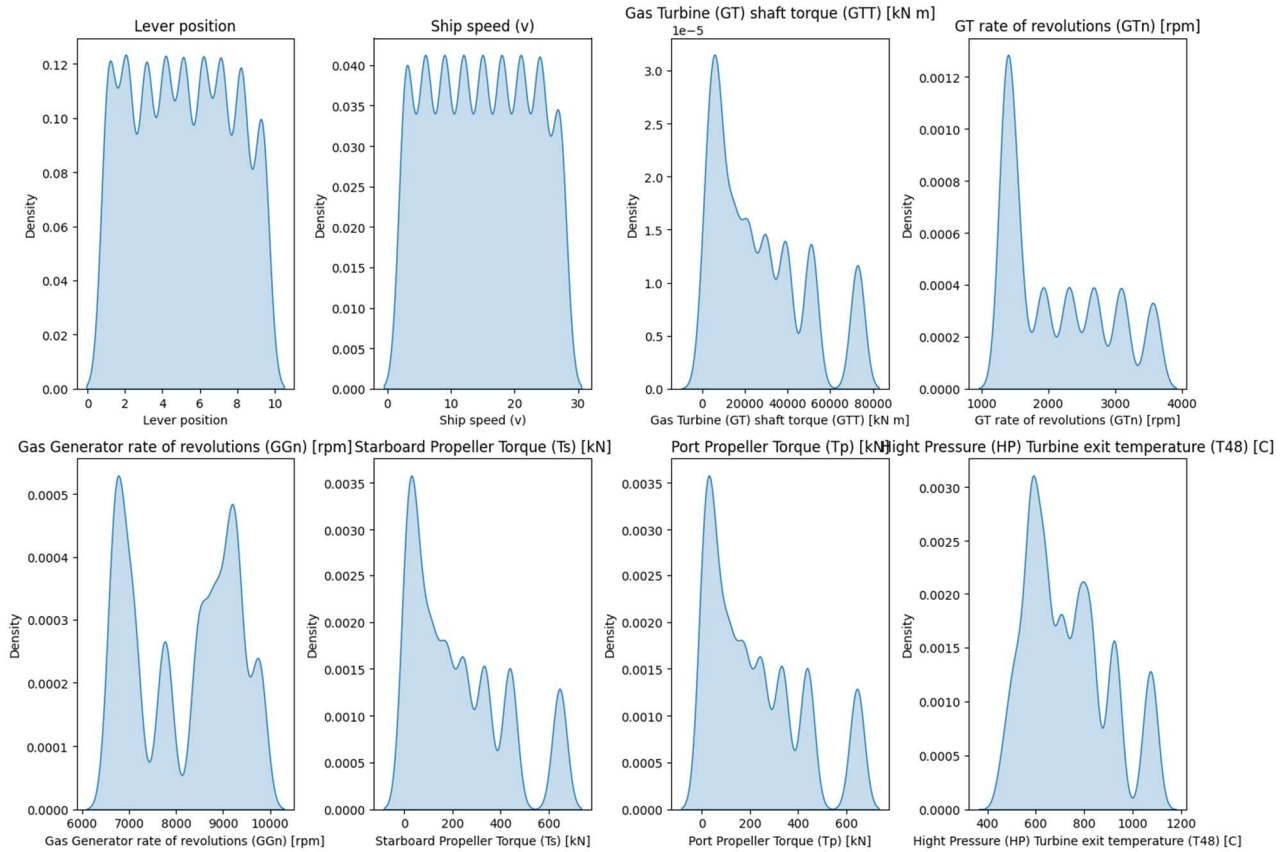
```

1  figure_engine.figure(figsize=(14,10))

    for i, col in enumerate(df.columns[:8]):
        figure_engine.subplot(2, 4, i+1)
        sea_engine.kdeplot(df[col], fill=True)
        figure_engine.title(col)

    figure_engine.tight_layout()
    figure_engine.show()

```



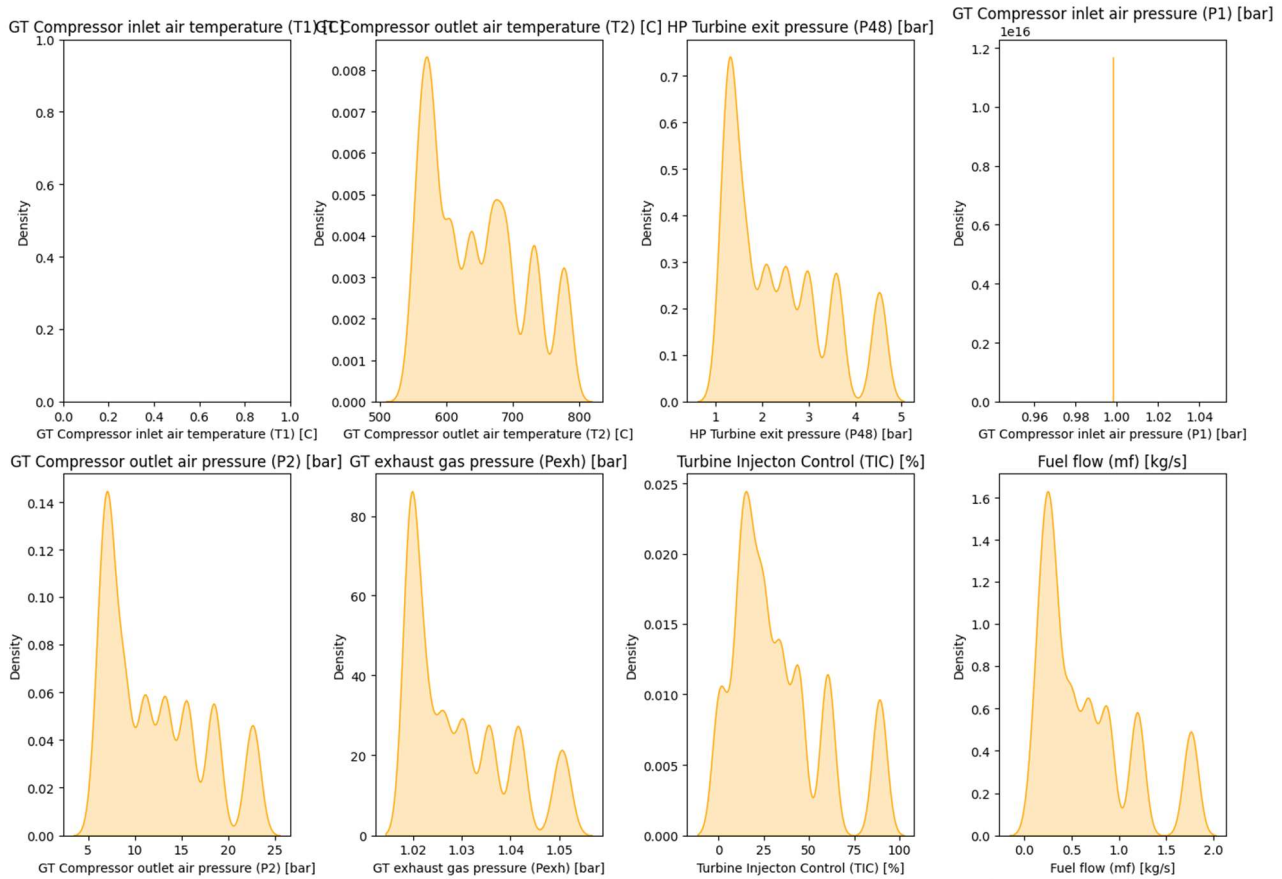
```

figure_engine.figure(figsize=(14,10))

for i, col in enumerate(df.columns[8:16]):
    figure_engine.subplot(2, 4, i+1)
    sea_engine.kdeplot(df[col], fill=True, color="orange")
    figure_engine.title(col)

figure_engine.tight_layout()
figure_engine.show()

```



```

figure_engine.subplot(1,2,1)
sea_engine.kdeplot(df["GT Compressor decay state coefficient"], fill=True, color="blue")
figure_engine.title("Compressor Decay Distribution")

figure_engine.subplot(1,2,2)
sea_engine.kdeplot(df["GT Turbine decay state coefficient"], fill=True, color="red")
figure_engine.title("Turbine Decay Distribution")

figure_engine.tight_layout()
figure_engine.show()

```

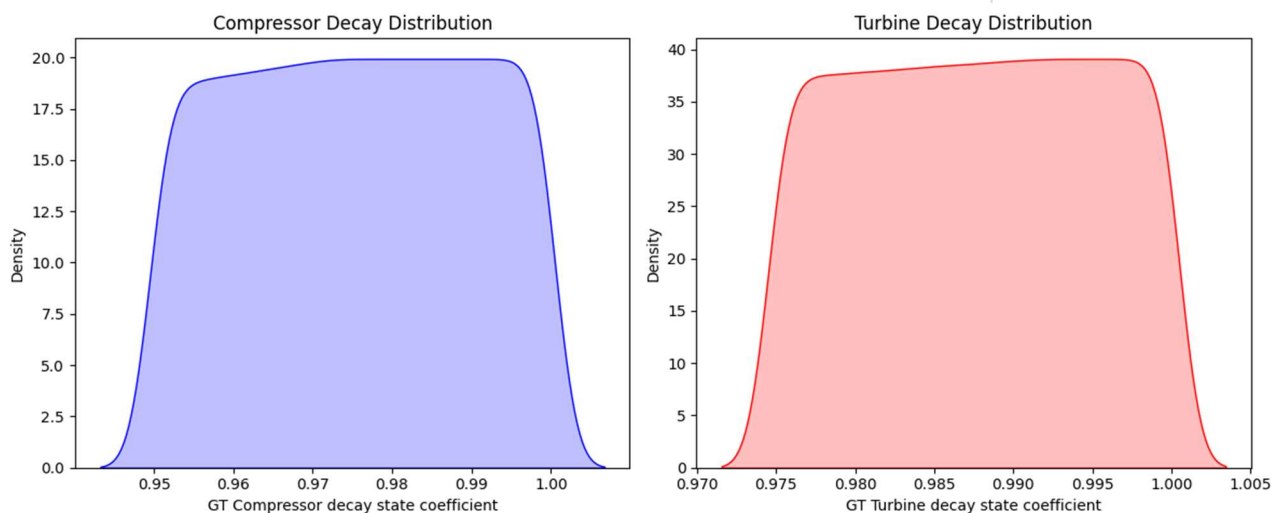


Figure 33 KDE distributions after outlier removal across features and targets

```
[ ] ▶
X = df.drop(columns=[
    "GT Compressor decay state coefficient",
    "GT Turbine decay state coefficient"
])

y_comp = df["GT Compressor decay state coefficient"]
y_turb = df["GT Turbine decay state coefficient"]

[ ]
X_train, X_test, y_comp_train, y_comp_test = train_test_split(
    X, y_comp, test_size=0.2, random_state=42
)

_, _, y_turb_train, y_turb_test = train_test_split(
    X, y_turb, test_size=0.2, random_state=42
)

[ ]
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

[ ]
joblib.dump(scaler, "scaler.pkl")

▼ ['scaler.pkl']
```

Figure 34 Train-test split, StandardScaler fitting, transformation, and joblib serialisation

Appendix 3: Model Development, Cross-Validation and Hyperparameter Tuning

```
def evaluate_model(model, X_train, X_test, y_train, y_test, name="Model"):
    model.fit(X_train, y_train)
    pred = model.predict(X_test)

    # Regression Metrics
    r2 = r2_score(y_test, pred)
    rmse = np.sqrt(mean_squared_error(y_test, pred))
    mae = mean_absolute_error(y_test, pred)

    print(f"\n {name} REGRESSION RESULTS ")
    print("R2 :", r2)
    print("RMSE :", rmse)
    print("MAE :", mae)

    # Actual vs Predicted Plot
    figure_engine.figure(figsize=(6, 5))
    figure_engine.scatter(y_test, pred, alpha=0.3, color="steelblue", s=10)
    figure_engine.plot(
        [y_test.min(), y_test.max()],
        [y_test.min(), y_test.max()],
        "r--", linewidth=1.5, label="Perfect Prediction"
    )
    figure_engine.xlabel("Actual")
    figure_engine.ylabel("Predicted")
    figure_engine.title(f"Actual vs Predicted - {name}")
    figure_engine.legend()
    figure_engine.tight_layout()
    figure_engine.show()

    # Residual Plot
    residuals = y_test - pred
    figure_engine.figure(figsize=(6, 4))
    figure_engine.scatter(pred, residuals, alpha=0.3, color="darkorange", s=10)
    figure_engine.axhline(0, color="red", linestyle="--", linewidth=1.5)
    figure_engine.xlabel("Predicted Values")
    figure_engine.ylabel("Residuals")
    figure_engine.title(f"Residual Plot - {name}")
    figure_engine.tight_layout()
```

Figure 35 evaluate_model function showing regression metrics, actual versus predicted plot, and residual plot

```
lr = LinearRegression()
lr_results = evaluate_model(
    lr, X_train, X_test,
    y_comp_train, y_comp_test,
    name="Linear Regression"
)
```

```

▶ rf = RandomForestRegressor(n_estimators=200, random_state=42)
rf_results = evaluate_model(
    rf, X_train, X_test,
    y_comp_train, y_comp_test,
    name="Random Forest"
)

```

```

▶ xgb = XGBRegressor(n_estimators=200, random_state=42)
xgb_results = evaluate_model(
    xgb, X_train, X_test,
    y_comp_train, y_comp_test,
    name="XGBoost"
)

```

```

▶ lr_turb = LinearRegression().fit(X_train, y_turb_train)
rf_turb = RandomForestRegressor(n_estimators=200, random_state=42).fit(X_train, y_turb_train)
xgb_turb = XGBRegressor().fit(X_train, y_turb_train)

```

```

lr_results_turb = evaluate_model(
    lr_turb,
    X_train, X_test,
    y_turb_train, y_turb_test,
    name="Linear Regression - Turbine"
)

```

```

rf_results_turb = evaluate_model(
    rf_turb,
    X_train, X_test,
    y_turb_train, y_turb_test,
    name="Random Forest - Turbine"
)

```

```

xgb_results_turb = evaluate_model(
    xgb_turb,
    X_train, X_test,
    y_turb_train, y_turb_test,
    name="XGBoost - Turbine"
)

```

Figure 36 Baseline model training for compressor and turbine targets across all three architectures

```
def regression_cv(model, X, y, name="Model"):
    kf = KFold(n_splits=5, shuffle=True, random_state=42)

    r2_scores, rmse_scores, mae_scores = [], [], []

    X = np.array(X)
    y = np.array(y)

    for train_idx, val_idx in kf.split(X):

        X_tr, X_val = X[train_idx], X[val_idx]
        y_tr, y_val = y[train_idx], y[val_idx]

        model.fit(X_tr, y_tr)
        y_pred = model.predict(X_val)

        r2_scores.append(r2_score(y_val, y_pred))
        rmse_scores.append(np.sqrt(mean_squared_error(y_val, y_pred)))
        mae_scores.append(mean_absolute_error(y_val, y_pred))

    print(f"\n {name} REGRESSION CV RESULTS ")
    print("R2      :", np.mean(r2_scores))
    print("RMSE    :", np.mean(rmse_scores))
    print("MAE     :", np.mean(mae_scores))

    return {
        "r2": np.mean(r2_scores),
        "rmse": np.mean(rmse_scores),
        "mae": np.mean(mae_scores)
    }

cv_lr = regression_cv(LinearRegression(), X_train, y_comp_train, "Linear Regression")
cv_rf = regression_cv(RandomForestRegressor(n_estimators=200, random_state=42), X_train, y_comp_train, "Random Forest")
cv_xgb = regression_cv(XGBRegressor(), X_train, y_comp_train, "XGBoost")

Linear Regression REGRESSION CV RESULTS
R2      : 0.839442647808537
RMSE    : 0.005866356012316989
MAE     : 0.004611228058287753

Random Forest REGRESSION CV RESULTS
R2      : 0.9938795818971637
RMSE    : 0.001144605557317122
MAE     : 0.0005865790852428931

XGBoost REGRESSION CV RESULTS
R2      : 0.9915310302097657
RMSE    : 0.0013451420415283878
MAE     : 0.0009521333160893281

cv_lr_t = regression_cv(LinearRegression(), X_train, y_turb_train, "Linear Regression")
cv_rf_t = regression_cv(RandomForestRegressor(n_estimators=200, random_state=42), X_train, y_turb_train, "Random Forest")
cv_xgb_t = regression_cv(XGBRegressor(), X_train, y_turb_train, "XGBoost")

Linear Regression REGRESSION CV RESULTS
R2      : 0.9095914184990675
RMSE    : 0.0022503280278965247
MAE     : 0.0016926740281333545

Random Forest REGRESSION CV RESULTS
R2      : 0.9871710708309299
RMSE    : 0.0008438887310525233
MAE     : 0.0003895453062575443

XGBoost REGRESSION CV RESULTS
R2      : 0.9817512241456618
RMSE    : 0.0010108531432397806
MAE     : 0.0006735594465549272
```

Figure 37 Five-fold cross-validation function and results for all models across both targets

```
def error_distribution(model, X_train, X_test, y_train, y_test, name):
    model.fit(X_train, y_train)
    pred = model.predict(X_test)

    errors = y_test - pred

    figure_engine.figure(figsize=(7,4))
    sea_engine.histplot(errors, kde=True, bins=30)
    figure_engine.title(f"Error Distribution - {name}")
    figure_engine.show()

    print(name)
    print("Mean Error:", np.mean(errors))
    print("Std Error :", np.std(errors))
```

```
error_distribution(lr, X_train, X_test, y_comp_train, y_comp_test, "Linear Regression- Compressor")
error_distribution(rf, X_train, X_test, y_comp_train, y_comp_test, "Random Forest- Compressor")
error_distribution(xgb, X_train, X_test, y_comp_train, y_comp_test, "XGBoost- Compressor")
```

Figure 38 Error distribution function and residual histogram outputs for all six baseline models

```
xgb_params = {
    "n_estimators": [200, 400, 600],
    "learning_rate": [0.01, 0.05, 0.1],
    "max_depth": [4, 6, 8],
    "subsample": [0.7, 0.8, 1.0],
    "colsample_bytree": [0.7, 0.8, 1.0]
}
```

```
xgb_grid = GridSearchCV(
    XGBRegressor(random_state=42),
    xgb_params,
    cv=3,
    scoring="r2",
    n_jobs=-1,
    verbose=1
)
```

```
xgb_grid.fit(X_train, y_comp_train)
```

```
best_xgb = xgb_grid.best_estimator_
print("Best XGB Params:", xgb_grid.best_params_)
```

Fitting 3 folds for each of 243 candidates, totalling 729 fits
Best XGB Params: {'colsample_bytree': 1.0, 'learning_rate': 0.05, 'max_depth': 8, 'n_estimators': 600, 'subsample': 0.7}

```
xgb_grid_turb = GridSearchCV(
    XGBRegressor(random_state=42),
    xgb_params,
    cv=3,
    scoring="r2",
    n_jobs=-1,
    verbose=1
)
```

```
xgb_grid_turb.fit(X_train, y_turb_train)
```

```
best_xgb_turb = xgb_grid_turb.best_estimator_
print("Best XGB Params (Turbine):", xgb_grid_turb.best_params_)
joblib.dump(best_xgb_turb, "xgb_turbine.pkl")
```

Fitting 3 folds for each of 243 candidates, totalling 729 fits
Best XGB Params (Turbine): {'colsample_bytree': 1.0, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 600, 'subsample': 0.7}
['xgb_turbine.pkl']

```

rf_params = {
    "n_estimators": [100, 200, 400],
    "max_depth": [None, 10, 20],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4]
}

rf_grid = GridSearchCV(
    RandomForestRegressor(random_state=42),
    rf_params,
    cv=3,
    scoring="r2",
    n_jobs=-1
)

rf_grid.fit(X_train, y_comp_train)

best_rf = rf_grid.best_estimator_
print("Best RF Params:", rf_grid.best_params_)

```

```

... Best RF Params: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 200}

```

Figure 39 GridSearchCV hyperparameter grids and fitting for all models — compressor and turbine targets

```
before = panda_core.DataFrame({
    "Linear Regression": lr_results,
    "Random Forest": rf_results,
    "XGBoost": xgb_results
}).T

after = panda_core.DataFrame({
    "Linear Regression": lr_results_tuned,
    "Random Forest": rf_results_tuned,
    "XGBoost": xgb_results_tuned
}).T

print("BEFORE TUNING Compressor")
display(before)

print("AFTER TUNING Compressor")
display(after)
```

```
before_turb = panda_core.DataFrame({
    "Linear Regression": lr_results_turb,
    "Random Forest": rf_results_turb,
    "XGBoost": xgb_results_turb
}).T

after_turb = panda_core.DataFrame({
    "Linear Regression": lr_turb_results_tuned,
    "Random Forest": rf_turb_results_tuned,
    "XGBoost": xgb_turb_results_tuned
}).T

print("BEFORE TUNING Turbine")
display(before_turb)

print("AFTER TUNING Turbine")
display(after_turb)
```

Figure 40 Before-and-after tuning comparison tables and R-squared improvement plots

```

▶ lr_results_tuned = evaluate_model(
    best_lr, X_train, X_test, y_comp_train, y_comp_test,
    name="Tuned Linear Regression - Compressor"
)

rf_results_tuned = evaluate_model(
    best_rf, X_train, X_test, y_comp_train, y_comp_test,
    name="Tuned Random Forest - Compressor"
)

xgb_results_tuned = evaluate_model(
    best_xgb, X_train, X_test, y_comp_train, y_comp_test,
    name="Tuned XGBoost - Compressor"
)

```

```

▶ lr_turb_results_tuned = evaluate_model(
    best_lr_turb,
    X_train, X_test,
    y_turb_train, y_turb_test,
    name="Tuned Linear Regression - Turbine"
)

rf_turb_results_tuned = evaluate_model(
    best_rf_turb,
    X_train, X_test,
    y_turb_train, y_turb_test,
    name="Tuned Random Forest - Turbine"
)

xgb_turb_results_tuned = evaluate_model(
    best_xgb_turb,
    X_train, X_test,
    y_turb_train, y_turb_test,
    name="Tuned XGBoost - Turbine"
)

```

Figure 41 Tuned model evaluation outputs for all six model instances

```

▶ from google.colab import drive
drive.mount('/content/drive')

# Set the folder path in your drive where models will be saved
save_path = "/content/drive/MyDrive/naval_propulsion_models/"

# Create the folder if it doesn't exist
import os
os.makedirs(save_path, exist_ok=True)

... Mounted at /content/drive

```

```

▶ # Untuned models – Compressor
joblib.dump(lr, save_path + "lr_compressor.pkl")
joblib.dump(rf, save_path + "rf_compressor.pkl")
joblib.dump(xgb, save_path + "xgb_compressor_base.pkl")

# Untuned models – Turbine
joblib.dump(lr_turb, save_path + "lr_turbine.pkl")
joblib.dump(rf_turb, save_path + "rf_turbine.pkl")
joblib.dump(xgb_turb, save_path + "xgb_turbine_base.pkl")

# Tuned models – Compressor
joblib.dump(best_lr, save_path + "best_lr_compressor.pkl")
joblib.dump(best_rf, save_path + "best_rf_compressor.pkl")
joblib.dump(best_xgb, save_path + "best_xgb_compressor.pkl")

# Tuned models – Turbine
joblib.dump(best_lr_turb, save_path + "best_lr_turbine.pkl")
joblib.dump(best_rf_turb, save_path + "best_rf_turbine.pkl")
joblib.dump(best_xgb_turb, save_path + "best_xgb_turbine.pkl")

# Scaler
joblib.dump(scaler, save_path + "scaler.pkl")

print("All models saved to Google Drive successfully.")
print("Location:", save_path)

... All models saved to Google Drive successfully.
Location: /content/drive/MyDrive/naval_propulsion_models/

```

Figure 42 Model serialisation to local storage and Google Drive using joblib

Appendix 4: SHAP Explainability and Learning Curve Analysis

```

▶ feature_names = X.columns.tolist()

# Compressor SHAP
explainer_comp = shap_engine.TreeExplainer(best_xgb)
shap_values_comp = explainer_comp(X_test)
shap_engine.summary_plot(shap_values_comp, X_test, feature_names=feature_names)
shap_engine.plots.waterfall(shap_values_comp[0])
shap_engine.plots.bar(shap_values_comp)

# Turbine SHAP
explainer_turb = shap_engine.TreeExplainer(best_xgb_turb)
shap_values_turb = explainer_turb(X_test)
shap_engine.summary_plot(shap_values_turb, X_test, feature_names=feature_names)
shap_engine.plots.waterfall(shap_values_turb[0])
shap_engine.plots.bar(shap_values_turb)

```

Figure 43 SHAP TreeExplainer for compressor target with beeswarm, waterfall, and bar plots

```

▶ feature_names = X.columns.tolist()

# Compressor SHAP
explainer_comp = shap_engine.TreeExplainer(best_xgb)
shap_values_comp = explainer_comp(X_test)
shap_engine.summary_plot(shap_values_comp, X_test, feature_names=feature_names)
shap_engine.plots.waterfall(shap_values_comp[0])
shap_engine.plots.bar(shap_values_comp)

# Turbine SHAP
explainer_turb = shap_engine.TreeExplainer(best_xgb_turb)
shap_values_turb = explainer_turb(X_test)
shap_engine.summary_plot(shap_values_turb, X_test, feature_names=feature_names)
shap_engine.plots.waterfall(shap_values_turb[0])
shap_engine.plots.bar(shap_values_turb)

```

Figure 44 SHAP TreeExplainer for turbine target with beeswarm, waterfall, and bar plots

```

) # Learning Curve Analysis
def plot_learning_curve(model, X, y, name="Model"):
    train_sizes, train_scores, val_scores = learning_curve(
        model, X, y, cv=5, scoring="r2",
        train_sizes=np.linspace(0.1, 1.0, 10), n_jobs=-1
    )
    figure_engine.figure(figsize=(8, 5))
    figure_engine.plot(train_sizes, np.mean(train_scores, axis=1), marker="o", label="Train R²", color="steelblue")
    figure_engine.plot(train_sizes, np.mean(val_scores, axis=1), marker="o", label="Validation R²", color="darkorange")
    figure_engine.title(f"Learning Curve - {name}")
    figure_engine.xlabel("Training Set Size")
    figure_engine.ylabel("R² Score")
    figure_engine.legend()
    figure_engine.grid()
    figure_engine.tight_layout()
    figure_engine.show()

plot_learning_curve(LinearRegression(), X_train, y_comp_train, "Linear Regression - Compressor")
plot_learning_curve(RandomForestRegressor(n_estimators=200, random_state=42), X_train, y_comp_train, "Random Forest - Compressor")
plot_learning_curve(XGBRegressor(n_estimators=200, random_state=42), X_train, y_comp_train, "XGBoost - Compressor")

plot_learning_curve(LinearRegression(), X_train, y_turb_train, "Linear Regression - Turbine")
plot_learning_curve(RandomForestRegressor(n_estimators=200, random_state=42), X_train, y_turb_train, "Random Forest - Turbine")
plot_learning_curve(XGBRegressor(n_estimators=200, random_state=42), X_train, y_turb_train, "XGBoost - Turbine")

```

Figure 45 Learning curve function definition and its six calls across all models and targets

Appendix 5: Streamlit Dashboard Implementation

Note: The following snippets are representative excerpts from app.py. Replace each placeholder with the corresponding screenshot taken directly from your Streamlit application file.

Figure A5.1. Dashboard page configuration, header block, and model loading at startup

```

python
import streamlit as st
import joblib

```

```
import numpy as np
import pandas as pd
import shap
import matplotlib.pyplot as plt

st.set_page_config(
    page_title="Naval Propulsion Digital Twin",
    page_icon="ship",
    layout="wide"
)

40 st.markdown("""
    <div style='background-color:#0a1628; padding:20px; border-radius:10px;'>
    <h1 style='color:white; text-align:center;'>
    Naval Propulsion System Digital Twin Dashboard
    </h1>
    <p style='color:#a0aec0; text-align:center;'>
    XGBoost Regression with SHAP Explainability
    </p>
    </div>
    """, unsafe_allow_html=True)

67 # Load trained models and scaler
xgb_comp = joblib.load("best_xgb_compressor.pkl")
xgb_turb = joblib.load("best_xgb_turbine.pkl")
scaler = joblib.load("scaler.pkl")
```

Figure A5.2. Prediction Dashboard tab with input sliders, four scenario presets, and Run Prediction session state logic

```
python
125 tab1, tab2, tab3, tab4, tab5 = st.tabs([
    "Prediction Dashboard",
    "Live Monitoring",
    "SHAP Explainability",
    "EDA",
    "Fleet Simulation"
])

with tab1:
    st.subheader("Propulsion Health Prediction")

    # Scenario presets
    presets = {
        "Healthy": {
            "Lever position (lp)": 5,
            "Ship speed (v) [knots]": 12,
            "GT shaft torque (GTT) [kN m]": 150,
            "GT rate of revolutions (GTn) [rpm]": 1400,
            "Gas Generator rate of revolutions (GGn) [rpm]": 9000,
            "Starboard Propeller Torque (Ts) [kN]": 120,
            "Port Propeller Torque (Tp) [kN]": 118,
            "HP Turbine exit temperature (T48) [C]": 520,
            "GT Compressor inlet air temperature (T1) [C]": 15,
            "GT Compressor outlet air temperature (T2) [C]": 320,
            "HP Turbine exit pressure (P48) [bar]": 2.8,
```

```

    "GT Compressor inlet air pressure (P1) [bar]": 1.01,
    "GT Compressor outlet air pressure (P2) [bar]": 12.5,
    "Gas Generator rate of revolutions (GGn) [rpm].1": 9000,
    "Fuel flow (mf) [kg/s]": 0.5,
    "GT Compressor outlet air temperature (T2) [C].1": 320
},
"Moderate Degradation": { ... },
"Severe Degradation": { ... },
"Custom Manual": { ... }
}

selected_preset = st.selectbox("Select Scenario", list(presets.keys()))
preset_values = presets[selected_preset]

# Input sliders for all 16 features
input_data = {}
col1, col2 = st.columns(2)
feature_list = list(preset_values.keys())

for i, feature in enumerate(feature_list):
    with col1 if i % 2 == 0 else col2:
        input_data[feature] = st.slider(
            feature,
            min_value=0.0,
            max_value=1000.0,
            value=float(preset_values[feature])
        )

# Run Prediction button with session state
if st.button("Run Prediction"):
    input_array = np.array(list(input_data.values())).reshape(1, -1)
    input_scaled = scaler.transform(input_array)

    pred_comp = xgb_comp.predict(input_scaled)[0]
    pred_turb = xgb_turb.predict(input_scaled)[0]

    st.session_state["pred_comp"] = pred_comp
    st.session_state["pred_turb"] = pred_turb

    st.success(f"Compressor Decay Coefficient: {pred_comp:.6f}")
    st.success(f"Turbine Decay Coefficient: {pred_turb:.6f}")

```

Figure A5.3. SHAP Explainability tab rendering waterfall, bar, and beeswarm plots via st.pyplot

```

python
with tab3:
    st.subheader("SHAP Feature Explainability")

    if "pred_comp" in st.session_state:
        input_array = np.array(list(input_data.values())).reshape(1, -1)
        input_scaled = scaler.transform(input_array)

        # Compressor SHAP
        explainer_comp = shap.TreeExplainer(xgb_comp)
        shap_values_comp = explainer_comp(input_scaled)

        st.write("Compressor Waterfall Plot")

```

```

18 fig, ax = plt.subplots()
shap.plots.waterfall(shap_values_comp[0], show=False)
st.pyplot(fig)
plt.clf()

st.write("Compressor Global Bar Plot")
fig, ax = plt.subplots()
shap.plots.bar(shap_values_comp, show=False)
st.pyplot(fig)
plt.clf()

# Turbine SHAP
explainer_turb = shap.TreeExplainer(xgb_turb)
shap_values_turb = explainer_turb(input_scaled)

st.write("Turbine Waterfall Plot")
fig, ax = plt.subplots()
shap.plots.waterfall(shap_values_turb[0], show=False)
st.pyplot(fig)
plt.clf()

else:
    st.info("Run a prediction first to generate SHAP explanations.")

```

Figure A5.4. Live Monitoring tab with real-time trend charts, health index calculation, and auto-refresh toggle

```

python
with tab2:
    st.subheader("Live Monitoring")

31 auto_refresh = st.toggle("Enable Auto-Refresh", value=False)

36 if "history" not in st.session_state:
    st.session_state["history"] = []

if "pred_comp" in st.session_state:
    pred_comp = st.session_state["pred_comp"]
    pred_turb = st.session_state["pred_turb"]

# Health index calculation (Z-score normalised composite)
health_index = (pred_comp + pred_turb) / 2

89 timestamp = pd.Timestamp.now().strftime("%H:%M:%S")
st.session_state["history"].append({
    "time": timestamp,
    "comp_decay": pred_comp,
    "turb_decay": pred_turb,
    "health_index": health_index
})

97 history_df = pd.DataFrame(st.session_state["history"])

st.metric("Compressor Decay", f"{pred_comp:.6f}")
st.metric("Turbine Decay", f"{pred_turb:.6f}")
st.metric("Health Index", f"{health_index:.6f}")

```

Trend charts

```
st.line_chart(history_df.set_index("time")[["comp_decay", "turb_decay"]])
st.line_chart(history_df.set_index("time")[["health_index"]])
```

Timestamped history log

```
st.dataframe(history_df, use_container_width=True)
```

if auto_refresh:

```
import time
time.sleep(3)
st.rerun()
```