CM50268 Consultancy Project

University of Bath - Department of Computer Science

April 9, 2025

1 Introduction

Industrial components degrade over time, thus requiring accurate integrity predictions to prevent failures. This report makes use of historical degradation data comprised of component characteristics over a fixed time period, with the aim to predict future integrity values.

1.1 Data

The data consists of 75 K components, with both irregular and noisy integrity measurements over a 45 day period. Each row contains unique component identifiers, real-valued characteristics of the component, day and integrity. Integrity is in the range $[0, 100]$. The first 50 K components are well-populated while the subsequent 25 are increasingly sparse.

1.2 Objectives

Specifically, this report seeks to achieve the following objectives:

1. accurately model and characterise degradation rate based on historical data
2. make the best "prediction" of component integrity, especially in the case where a given component has little associated data
3. to better understand potential causes of degradation, based on the additional component characteristics

2 Baseline Model

2.1 Formulation

Following client guidance, the baseline model follows exponential decay:

$$y_i(t) = u_i \exp\left(-\frac{v_i t}{100}\right) + \epsilon \tag{1}$$

i defines the K^{th} component, where:

- u_i corresponds to the intercept or, in other words, the initial integrity and is said to be within the range $[80, 100]$
- v_i is the decay rate of the component with expected range $[1, 10]$
- t refers to the day on which a measurement was taken
- ϵ denotes the measurement noise which is anticipated to be normally distributed with unknown standard deviation

We adapt the proposed model in (1) to leverage hierarchical Bayesian principles, namely to separate evidence from components such that we can make informed predictions for components that lack significant historical data.

2.2 Implementation

To adapt equation (1) for hierarchical Bayesian modelling, we use the Python probabilistic programming library, NumPyro. Our model separates the intercept and decay rate parameters for each component, using the plate functionality. We draw component-specific parameters from population-level

distributions as follows:

$$u_i \sim \mathcal{N}(u_\mu, u_\sigma) \tag{2}$$
$$v_i \sim \mathcal{N}(v_\mu, v_\sigma) \tag{3}$$

with population-level hyperpriors:

$$u_\mu \sim \mathcal{N}(90, 10) \tag{4}$$
$$u_\sigma \sim \text{HalfNormal}(3) \tag{5}$$
$$v_\mu \sim \mathcal{N}(5, 5) \tag{6}$$
$$v_\sigma \sim \text{HalfNormal}(2) \tag{7}$$

and simultaneously we model the measurement noise over a log scale as:

$$\log \epsilon \sim \text{Uniform}(-1, 2) \tag{8}$$
$$\epsilon = 10^{\log \epsilon} \tag{9}$$

We parametrised our distributions for intercept and decay rate such that most probability density resides in the anticipated regions, $u \in [80, 100], v \in [1, 10]$, while building in the expectation of occasional outliers. This results in the following model diagram, generated using the NumPyro render_model() function (figure 1).

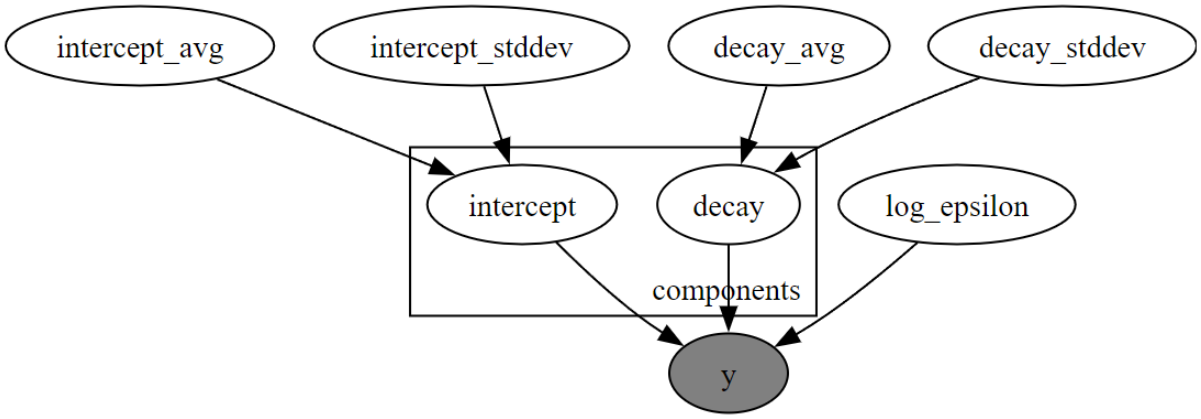

Figure 1: Baseline model diagram

We run our model with the Markov Chain Monte Carlo (MCMC) algorithm using the No-U-Turn Sampler (NUTS) kernel. We run with 500 warm-up iterations, 2000 sampling iterations and four chains to draw samples from the posterior distribution. We were able to verify the convergence of our sampling using the plot_trace() method, evidenced in figure 2. The plots show stationarity, an important characteristic that increases confidence of our samples being representative of the posterior distribution. Simultaneously, we experience 0 divergences and $\hat{R}$ scores of 1.00 for all sampled parameters, further confirming reliability of our MCMC results wherein our chains converged.

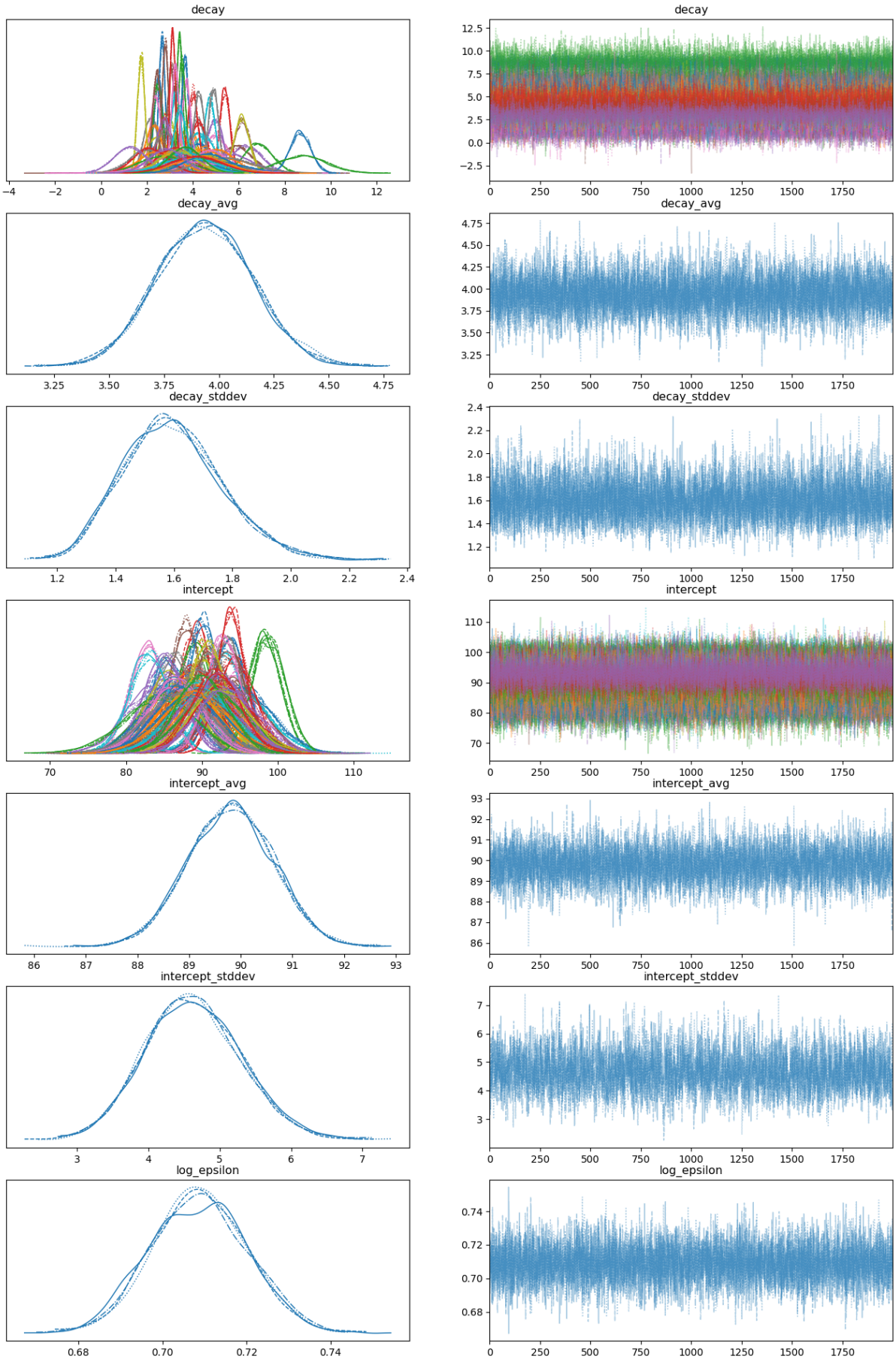

Figure 2: Baseline trace plots

2.3 Results

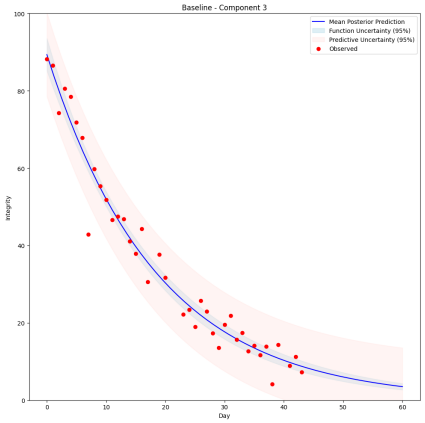

Figure 3: Baseline component 3

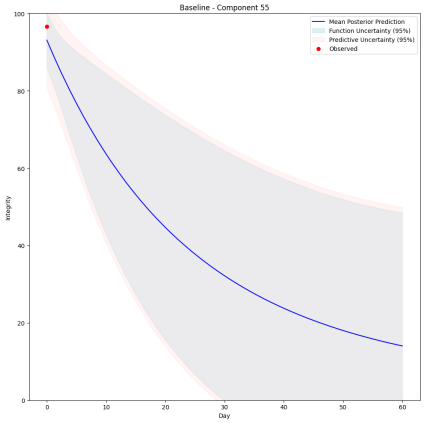

Figure 4: Baseline component 55

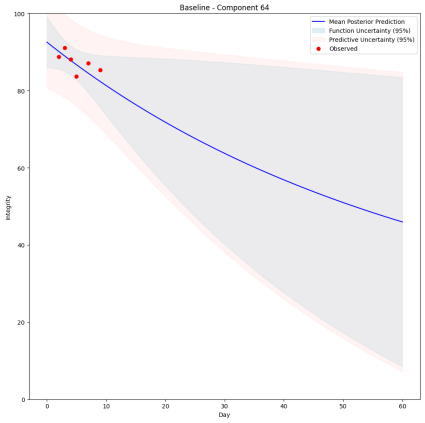

Figure 5: Baseline component 64

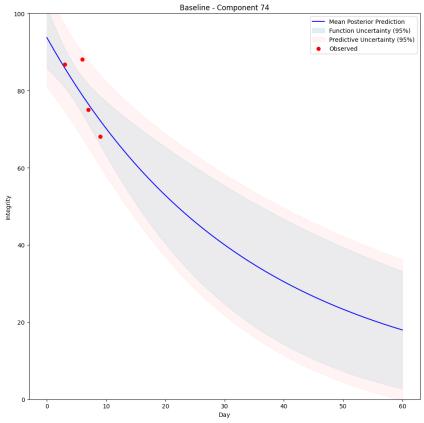

Figure 6: Baseline component 74

Figure 7: Baseline plots over components with varying measurement quantities

The predictive plots generated are consistent with expectations. For components with limited historical data, our (mean) posterior predictions leverage evidence from other components to make informed predictions. However, in doing so it introduces significantly greater uncertainty in the function estimate because greater emphasis is placed on inter-component relationships as opposed to direct historical evidence. This increases the further ahead you attempt to predict, which intuitively makes sense - there is more room for factors to contribute to the component degradation. This is clear in the comparison between component 3 which has historical data over a larger time period, and component 55 which only has one early integrity measurement and thus significantly wider error bars. Components 64 and 74 demonstrated the reduction in uncertainty incurred when introducing extra measurements.

3 Enhanced Model

3.1 Formulation

In our enhanced model, the client suggested the incorporation of 5 component characteristics denoted x_{ji}. This extra contextual information provides further factors which could contribute to the degradation, and allows us to refine our predictions by weighting the characteristics with w_j. We can now write the

predictive function as:

$$f_i(t) = u_i \exp\left(-\left(v_i + \sum_{j=1}^{5} w_j x_{ji}\right) t/100\right) \tag{10}$$

A key piece of advice by the client was that, in introducing weighted characteristics, the expected magnitude of decay rate would be reduced.

3.2 Implementation

To adapt the enhanced model formula for hierarchical Bayesian modeling, we again leverage the NumPyro probabilistic programming library. The enhanced model builds upon our baseline by incorporating the component characteristics, while maintaining the hierarchical structure.

As in our baseline model, we draw component-specific parameters from population-level distributions with modified population-level hyperpriors to account for the expected reduction in decay rate magnitude:

$$v_\mu \sim \mathcal{N}(3, 2) \tag{11}$$
$$v_\sigma \sim \text{HalfNormal}(2) \tag{12}$$

For the characteristic weights, we employ:

$$w_j \sim \mathcal{N}(0, 5) \quad \text{for } j \in 1, 2, 3, 4, 5 \tag{13}$$

All other hyperpriors are defined as in our baseline model. For the characteristic weights, we chose a zero-centred Gaussian prior with a moderate standard deviation to allow the model to learn both positive and negative effects of each characteristic on degradation. The summation of weighted characteristics can be optimised by plating the weights and computing the dot product with x, making use of NumPyro's vectorised operations. Weights are sampled at the population level, since they are common across components. This results in the following model diagram (figure 8).

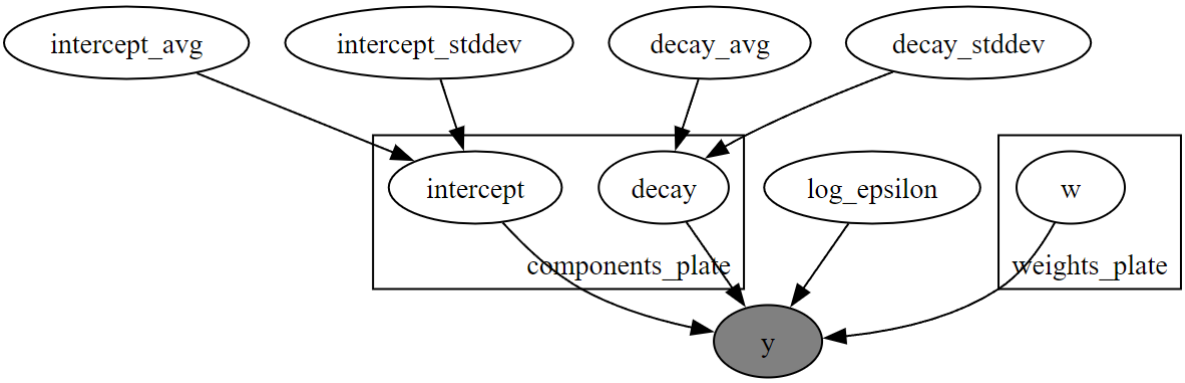

Figure 8: Enhanced model diagram

We use the same combination of NUTS and MCMC to run our model. The convergence is again verified through trace plots (figure 9) and $\hat{R}$ values of 1.00 for all parameters, indicating reliable posterior sampling.

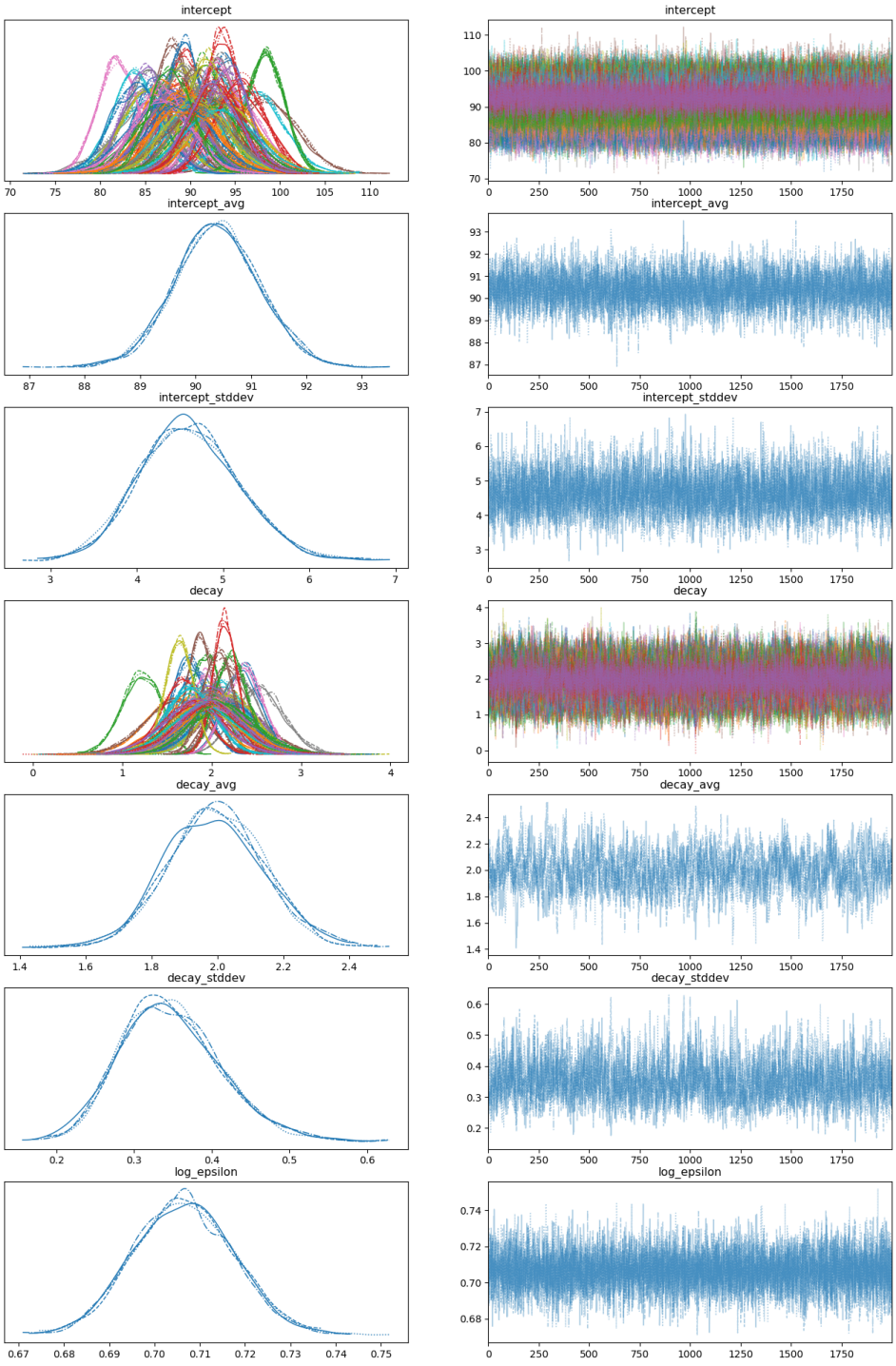

Figure 9: Enhanced model trace plots

3.2.1 Blind Test Predictions

We generate predictions of the probability being less than or equal to 30% at day 30 for components indexed 50 through 74. To do so, we model each component's enhanced predictive function as a normal distribution with μ = mean posterior predictor and σ = standard deviation of posterior predictor, where the predictor is computed with $t = 30$. We then simply compute $p(\textit{integrity} \leq 30)$. These can be found in the attached *predictions.csv* file.

3.3 Results

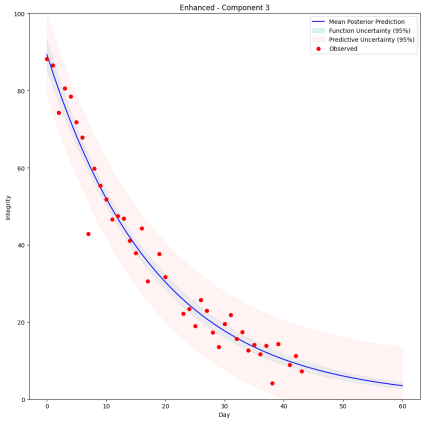

Figure 10: Enhanced component 3

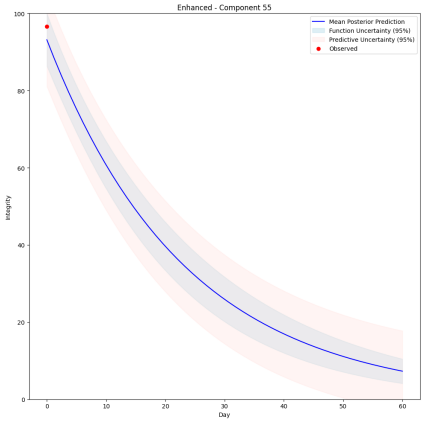

Figure 11: Enhanced component 55

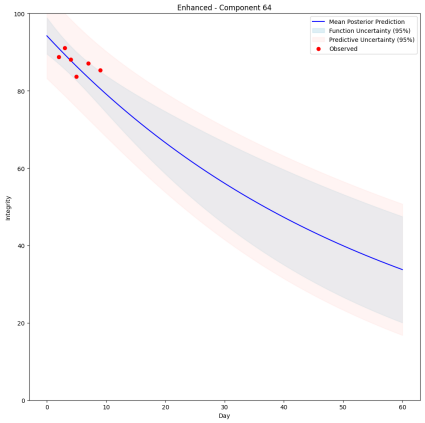

Figure 12: Enhanced component 64

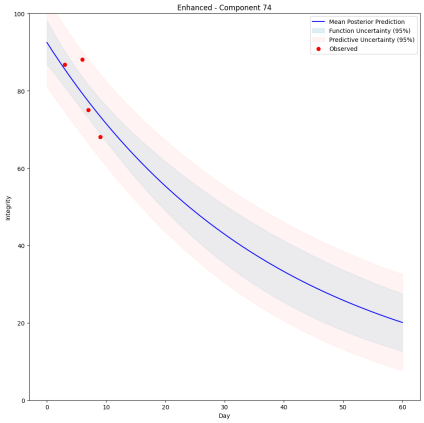

Figure 13: Enhanced component 74

Figure 14: Enhanced plots over components with varying measurement quantities

In adding weighted characteristics, we observe a significant reduction in function uncertainty over the same components (figure 14). This is due to the additional contextual information regarding the component, likely correlated with component properties. Thus, the model can better learn to relate components based on these additional properties leading to increasingly informed predictions. We observe that even for component 55, with one integrity measurement, we have relatively narrow uncertainty bars.

3.3.1 Relevant component characteristics

The enhanced implementation allows us to not only predict component degradation with greater accuracy and reduced uncertainty, but to quantify the impact of component characteristics on integrity

predictions. We do so using the weight trace plots (figure 15). This shows that characteristic 3 (2 in the legend due to array indexing) is the largest contributor to integrity degradation, by considerable distance. The higher weighting correlates to greater contribution of X3 to the prediction relative to the other characteristics. In terms of weight magnitude, it is approximately four times larger than other characteristic weights. Simultaneously characteristic 5 has a narrow distribution centred at around -0.05, suggesting negligible impact to integrity degradation.

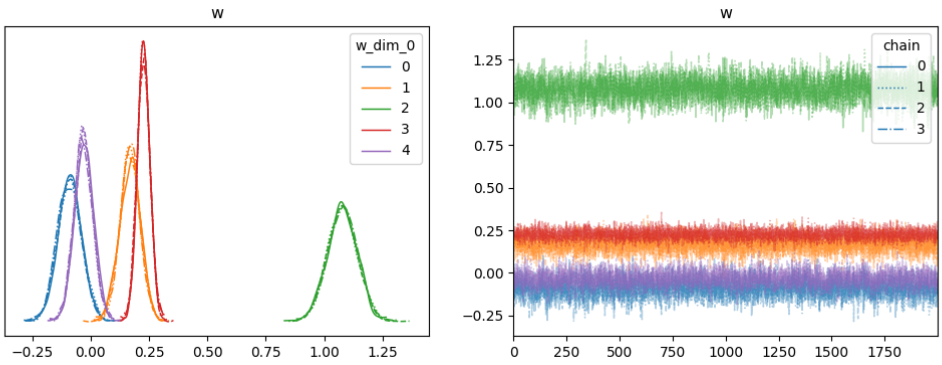

Figure 15: Enhanced model weight trace plots

4 Black-box model

In this section I will demonstrate the implementation of a black-box model, namely ElasticNet, while addressing queries related to alternate approaches, such as neural networks and Support Vector Machines.

4.1 ElasticNet

We demonstrate the application of a "black-box" approach using ElasticNet. This is fundamentally linear regression with a combination of L_1 and L_2 regularisation, offering a balance between penalising large weights and introducing sparsity. We implement ElasticNet using *scikit-learn*. To do so, we engineer the following features:

- Initial integrity
- Integrity gradient over the first 10 days. We set this to 0 if a component has insufficient measurements to calculate a gradient.
- Mean integrity over first 10 days.
- Standard deviation of integrity over first 10 days.

Component characteristics x are also appended to our feature selection.

The corresponding target variable is the closest integrity measurement to the 30^{th} day, from $t \in [23, 37]$. This allows us to introduce extra measurements to boost our training set. If a component has no measurements in this range, we exclude it from the training set. We then split this data into 80:20 training and validation splits. We plot true-predicted values in figure 16.

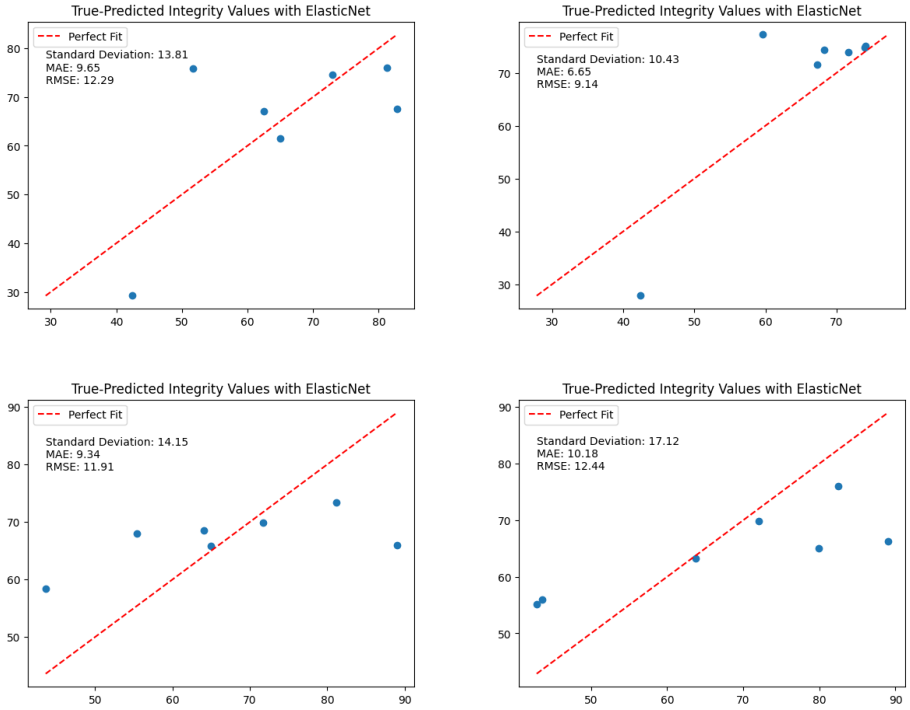

Figure 16: Black-box true-predicted plots for validation set, for four random train-val splits

In our plots, we observe some linearity in true-predicted values although this is weakly correlated. Furthermore, there is significant variability introduced due to the train-validation splits. This is due to the limited samples available.

We perform inference on components indexed 50 through 74, which lack measurements for our target of day 30. We compare black-box predictions to our enhanced model in figure 17.

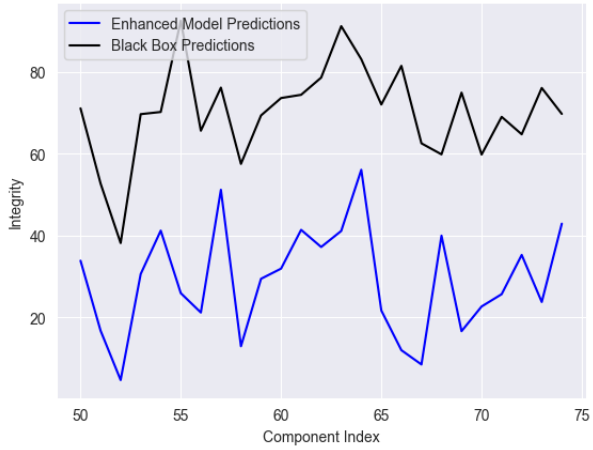

Figure 17: Comparison of black-box and enhanced model predictions for day 30 over components 50 through 74

Evidently, our black-box approach tends to predict significantly higher than the enhanced model. On initial observation, some predictions seem unreasonable - from components with measurements it is highly unlikely that a component has an integrity of 80% after 30 days. Interestingly, both predictors observe a similar trend across components, however showing that some underlying pattern in the data is learned by our black-box approach.

4.2 Neural Networks

"The Bayesian hierarchical approach sounds quite complex. Can you not simply build a black-box model that maps from some of our usable data to directly output a prediction? e.g. just train a neural network"

While neural networks have been shown to excel at numerous tasks, they are dependent on extensive datasets to learn the complex underlying patterns. In our case, we are limited to 879 time entries spread across 75 K components - many of which are sparse. It is unlikely that a neural network would be able to to learn from such limited data. One may question how our approach instead can still "learn" in this case. Using your guidance, we are able to impose mathematical formulae in our Bayesian model to model physical parameters and thus incorporate domain expertise. Hierarchical in nature, we can share information across components to provide robust predictions even for components lacking measurements. Thus, we link back to objectives (1) and (2) - we believe that a neural network would fail to achieve these, thus falling short of your requirements.

Simultaneously, the black-box nature of neural networks offers limited interpretability. While one could look at their weights, there is little indication with respect to what each weight is aligned with, i.e. component characteristics. However, the incorporation of our weighted characteristics allows us to observe their relative impact on integrity predictions, thus offering reasoning as to the real-world implications. This directly allows us to achieve objective (3).

4.3 Support Vector Machines

"I've seen some work in the past that made use of a popular model called the Support Vector Machine. It is said to be very accurate, and if I remember correctly, was deployed in many applications at the time. Can you explain why you do not use this model in your analysis, and instead use Bayesian techniques?"

As a foundational model, Support Vector Machines (SVMs) frequently achieve top performance in benchmarks. Moreover, they are theoretically better suited to smaller datasets than neural networks. However, feature engineering is required to adapt our dataset for use in SVMs, as in our ElasticNet implementation. In doing so, rich information is lost, thus impacting the model's ability to make grounded predictions. Simultaneously, they are not hierarchical in nature, meaning that one may either choose to combine all component data into one model, or train one model for each component. In either approach, our predictions cannot effectively learn inter-component relationships in the same manner as our Bayesian approach. Further to this, as in neural networks, one is unable to build in these mathematical constraints from domain knowledge efficiently. Therefore, we argue that they would be inadequate for objective (2), not providing the "best" integrity predictions.

Again, for objective (3) they offer little interpretability nor understanding of degradation causes. While one can obtain feature importance in SVMs through various techniques such as permutation importance, depending on the kernel choice, there is little quantification as to the physical impact of each characteristic x, for example. Comparatively, our Bayesian approach offers direct insight into the role of each characteristic in integrity predictions through our imposed weights.

Simultaneously, they only offer point estimates such that we cannot offer uncertainty measurements which are vital to risk assessments, which could be of utmost importance in the use of these components.

5 Conclusion

To conclude, this project successfully developed a Bayesian hierarchical framework to model component degradation, achieving all three client objectives. The enhanced model offered narrower uncertainty

intervals, improving our ability to predict future integrity values. We demonstrated that these intervals remain significantly smaller, even for components with little associated data, thanks to the incorporation of component characteristics. Simultaneously, trace plots offered insight into the contributions of different characteristics, providing increased understanding into degradation causes. Moreover, we justify our model selection over alternative black-box approaches - emphasised through our ElasticNet example.