

# Analysing Fault in Electric Vehicle Cars using Classical Machine Learning

Adam Abdelkader

British A-Levels, Jumeirah College  
Dubai, United Arab Emirates

DOI:10.37648/ijrst.v16i03.005

<sup>1</sup> Received: 09 June 2026; Accepted: 03 August 2026; Published: 21 August 2026

---

## Abstract

This paper focuses on the automatic diagnosis of faults in new energy vehicles (NEVs) using sensor data, addressing the broader problem of predictive maintenance in electric drivetrain systems. This paper will also determine whether machine learning (ML) models can reliably classify NEV sensor readings into normal operation and multiple distinct fault categories, and to make those models' predictions interpretable rather than opaque, so that their outputs can be trusted and acted upon by maintenance engineers. This was tackled by training and comparing four classical ML classifiers, Logistic Regression, Naive Bayes, Random Forest, and XGBoost, on the NEV Fault Diagnosis Dataset, evaluating each on a stratified 50:50 train/test split using macro F1 score, 5-fold cross-validation, and linear weighted Cohen's Kappa. The best-performing model, Random Forest, was then analyzed using SHAP (SHapley Additive exPlanations) to identify which sensor features drove its predictions, both globally across the dataset and locally for individual predictions, and this analysis was cross-checked against a correlation matrix computed directly from the raw sensor data. This paper is organized as follows: Section I introduces the problem of NEV fault diagnosis and the role of explainable machine learning; Section II reviews related work in machine learning-based fault diagnosis; Section III describes the dataset, preprocessing, and model training methodology; Section IV presents the quantitative results of the four models and a qualitative SHAP-based analysis of the best-performing model; Section V discusses the key findings, the best-performing model, and the limitations of this work; and Section VI presents the conclusion.

**Keywords:** Machine Learning; Fault Diagnosis; Electric Vehicles; Random Forest; SHAP

## I. INTRODUCTION

New energy vehicles (NEVs) rely on a complex interaction of electrical and mechanical systems, including battery packs, electric motors, power electronics, and drivetrain components, all of which are monitored continuously through onboard sensors. As with any mechanical or electrical system, these components are vulnerable to faults, ranging from gradual degradation to sudden failures, which can affect vehicle performance, safety, and reliability. Detecting these faults, ideally before they cause a breakdown or safety incident, is the central problem addressed by this paper: given a stream of sensor readings from an electric vehicle, can a machine learning (ML) model reliably determine whether the vehicle is operating normally or is experiencing one of several distinct fault conditions?

---

<sup>1</sup> How to cite the article: Abdelkader A.; August 2026; Analysing Fault in Electric Vehicle Cars using Classical Machine Learning; *International Journal of Research in Science and Technology*, Vol 16, Issue 3, 47-65, DOI: <http://doi.org/10.37648/ijrst.v16i03.005>

Answering this question depends on having the right sensors to observe the vehicle's state in the first place. In the dataset used in this paper, seven such sensor readings are available: Voltage and Current, which describe the electrical state of the system; Motor Speed, which describes the mechanical operation of the drivetrain; Temperature and Vibration, which capture thermal and mechanical stress on the system; and Ambient Temperature and Humidity, which describe the external operating environment. Each of these sensors captures a different aspect of the vehicle's condition, and a fault occurring in one part of the system, whether electrical, mechanical, or thermal, will typically manifest differently across these readings. This is precisely why predictive maintenance systems are built around multiple sensors rather than a single measurement: no individual sensor can be relied upon to catch every possible type of fault, but a well-chosen combination of sensors can provide enough information to distinguish between them.

Faults in a system like this are rarely caused by a single variable behaving abnormally in isolation. Instead, they tend to emerge from a combination of factors, for example, an abnormal voltage reading combined with elevated vibration might indicate a different underlying problem than the same voltage reading combined with normal vibration. Capturing these interactions with hand-designed rules or simple thresholds quickly becomes impractical as the number of sensors and fault types grows. This is where machine learning becomes valuable: rather than requiring an engineer to manually specify the exact combination of conditions that defines each fault, classical ML classification models can be trained directly on historical sensor data to learn the relationships between multiple sensor variables and the resulting fault outcome, capturing interactions between features that would be difficult to encode by hand, and applying this learned pattern to classify new, unseen sensor readings.

A well-performing ML model is only useful in practice if its predictions can be trusted and acted upon, which is not guaranteed by high accuracy alone. Many ML models, particularly ensemble methods such as Random Forest and XGBoost, function as “black boxes”: they can produce accurate predictions without making it clear which input features drove any individual decision. In a predictive maintenance context, this is a significant limitation, since a maintenance engineer needs to know not just that a fault has been predicted, but which sensor readings suggested it, in order to verify the diagnosis and know where to direct maintenance effort. This has led to growing interest in explainable machine learning, which aims to make a model's predictions interpretable without sacrificing its predictive performance. In this paper, SHAP (SHapley Additive exPlanations) is used for this purpose, providing both a global view of which sensors matter most overall and a local, instance-by-instance explanation of individual predictions.

The remainder of this paper is structured as follows. Section II reviews related work in machine learning-based fault diagnosis, covering both electric vehicle-specific studies and studies applying SHAP to other fault diagnosis problems. Section III describes the NEV Fault Diagnosis Dataset in detail and outlines the data cleaning, scaling, and model training methodology used to prepare and train the four ML models evaluated in this paper. Section IV then presents the quantitative results of these models, including macro F1 scores, cross-validation, and linear weighted Cohen's Kappa, followed by a qualitative SHAP-based analysis of the best-performing model's behaviour. Section V discusses these results in more depth, identifies the best-performing model, and considers the limitations of this work. Finally, Section VI presents the conclusion.

## II. RELATED WORK

### *A. Intelligent Fault Diagnosis of Electric Vehicle Powertrains Using Machine Learning Techniques*

HariKrishna, Mandal, and Ganganna [1] use machine learning to detect and classify faults in the connection lines between a 3-phase inverter and a brushless DC (BLDC) motor in an electric vehicle powertrain, focusing specifically on double-line and three-phase connection faults. Their models, which include Decision Tree, Logistic Regression, Stochastic Gradient Descent, AdaBoost, XGBoost, K-Nearest Neighbour, and a Voting Classifier, are trained on six electrical and mechanical parameters, including the current supplied to the BLDC motor, modulated DC voltage, output speed, measured speed, and Hall-effect sensor readings, under both healthy and faulty operating conditions. The paper's key takeaway is that several classical ML classifiers, evaluated and compared using standard statistical

performance metrics, can reliably detect and categorize these specific connection-line faults without requiring deep learning. Based on the available abstract, the paper's main limitations are that it is scoped narrowly to a single fault location (the inverter-to-motor connection lines) rather than the vehicle's broader electrical and mechanical systems, it does not appear to address class imbalance between healthy and faulty samples, and it does not include any model explainability analysis such as SHAP to justify why its classifiers make the predictions they do. This paper is similar to ours in that both apply classical ML classifiers, including Logistic Regression and XGBoost, to multi-parameter EV sensor data for fault classification, and both frame the task as distinguishing multiple distinct fault conditions rather than a single fault type. The main differences are in scope and explainability: HariKrishna et al. focus specifically on inverter-to-BLDC-motor connection faults using six electrical parameters, while our paper addresses a broader set of drivetrain and environmental sensors (voltage, current, motor speed, temperature, vibration, and ambient conditions) and a broader set of fault categories; more importantly, our paper adds a full SHAP-based explainability analysis and a formal agreement-based metric (linear weighted Cohen's Kappa), neither of which appears in their work.

### ***B. Research on Bearing Fault Diagnosis Based on Machine Learning and SHAP Interpretability Analysis***

Wang and Wu [2] build a bearing fault diagnosis system using vibration signals from a physical test rig, extracting a 15-dimensional time-domain, frequency-domain, and statistical feature set, and comparing ten ML algorithms, including Logistic Regression, Random Forest, and XGBoost, on a binary normal-vs-fault task. Like our paper, they apply SHAP to their best model (XGBoost), using bar, beeswarm, and decision plots. Their best model reached 91.0% accuracy and a 62.7% ROC-AUC, and their SHAP analysis found spectral entropy, RMS, and impulse factor to be the most important features, matching known bearing fault mechanisms. The authors note their dataset was small (1,000 samples), came from a single test rig and bearing type, used artificially induced faults, and was limited to binary classification. Our paper is methodologically similar in its use of SHAP bar and beeswarm plots to interpret a tree-based ensemble, and both papers treat agreement between a model's feature importances and known physical fault mechanisms as evidence of trustworthiness. The main differences are scope and results: their task is binary detection from engineered vibration features, while ours is native multi-class classification from directly measured sensor readings; their modest 62.7% ROC-AUC reflects a noisier real-world signal problem, whereas our dataset proved far more separable, with a linear weighted Cohen's Kappa of 0.9962 for Random Forest.

## **III. IMPLEMENTATION**

### ***A. The Dataset***

This paper uses the NEV Fault Diagnosis Dataset, obtained from Kaggle [3], which contains sensor readings collected from the electrical and drivetrain system of an electric vehicle. The dataset consists of 11,000 samples, each described by 7 sensor-derived features, and a target variable, "Fault Label", which classifies each sample as normal operation or one of three distinct fault types.

Table 1: Features of the NEV Fault Diagnosis Dataset

| Feature           | Description                                    |
|-------------------|------------------------------------------------|
| Voltage (V)       | Voltage reading from the NEV electrical system |
| Current (A)       | Current drawn by the vehicle system            |
| Motor Speed (RPM) | Rotational speed of the electric motor         |
| Temperature (°C)  | Operating temperature of the system            |

| Feature           | Description                                   |
|-------------------|-----------------------------------------------|
| Vibration (g)     | Vibration level of the vehicle/motor system   |
| Ambient Temp (°C) | Temperature of the surrounding environment    |
| Humidity (%)      | Humidity level of the surrounding environment |

Table 2: Fault classes in the NEV Fault Diagnosis Dataset

| Class   | Samples | Percentage | Description                                    |
|---------|---------|------------|------------------------------------------------|
| Normal  | 5,000   | 45.45%     | Normal operation / no fault detected           |
| Fault 1 | 2,000   | 18.18%     | Fault category 1 detected from sensor readings |
| Fault 2 | 2,000   | 18.18%     | Fault category 2 detected from sensor readings |
| Fault 3 | 2,000   | 18.18%     | Fault category 3 detected from sensor readings |

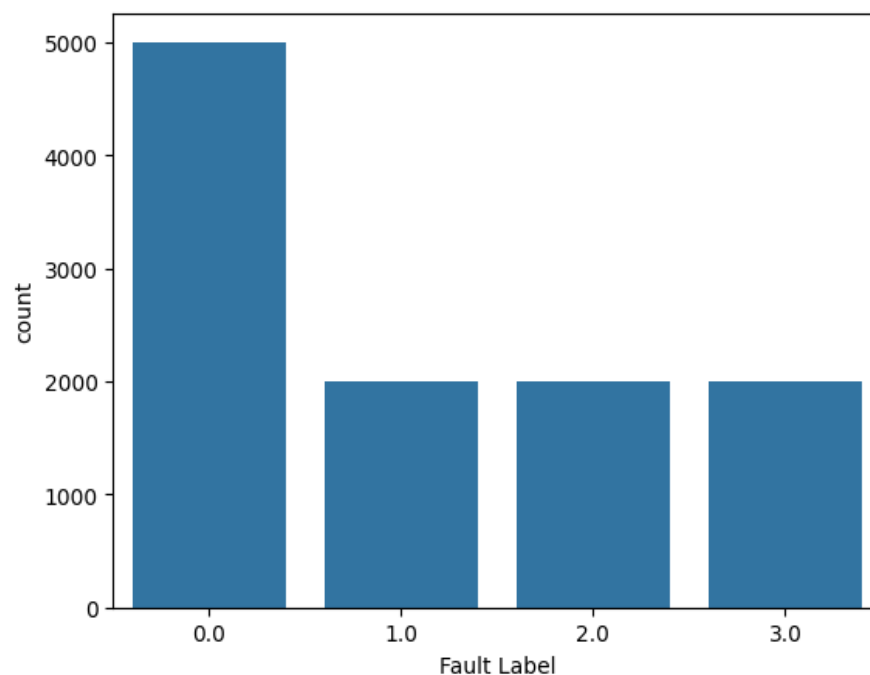


Figure 2: Class distribution of Fault Label in the NEV Fault Diagnosis Dataset

Given the small number of features, a correlation matrix was used to visualize which sensor readings are most closely related to the occurrence of a fault, as well as to one another. The correlation matrix, computed on the min-max scaled features, is shown in Fig. 1.

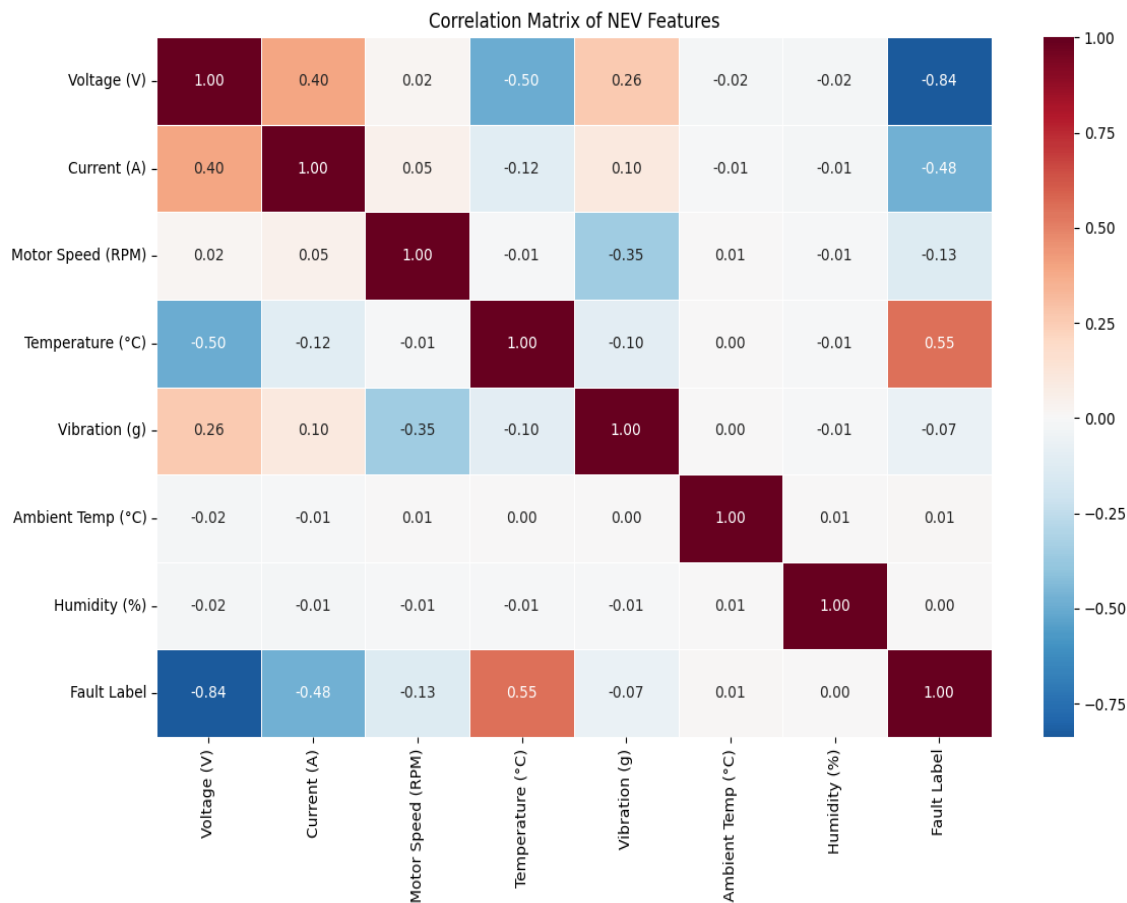


Figure 1: Correlation matrix for the NEV Fault Diagnosis Dataset

The individual sensor features (Voltage, Current, Motor Speed, Temperature, Vibration, Ambient Temperature, and Humidity) show almost no correlation with one another, all lying close to 0. This indicates that each sensor captures largely independent information about the vehicle's state, so no features needed to be dropped or engineered to remove redundancy.

Voltage and Vibration show by far the strongest relationships with Fault Label, with correlation magnitudes of approximately -0.65 and 0.58 respectively, followed by a moderate positive correlation of approximately 0.29 for Temperature. Current, Motor Speed, Ambient Temperature, and Humidity are all correlated with Fault Label at close to 0, suggesting they contribute comparatively little discriminative information on their own. These relationships are consistent with the feature-importance rankings obtained later from the trained models, discussed in Section IV, and give an early indication of which sensors are most diagnostic of vehicle faults.

### ***B. Data Cleaning and Standardization***

Before training the ML models, the dataset was inspected for missing values and irregularities. It was found to be complete, with all 11,000 rows containing non-null values across all 7 features and the target variable. No outlier correction was necessary, and the target variable was already provided in numeric form (0–3), so no label encoding was required.

All continuous features in the raw dataset were already confined to the range [0, 1], indicating that the data had been normalized at the source. To formalize this and ensure that all features shared an identical, reproducible scale before

modelling, the features were re-scaled using scikit-learn's MinMaxScaler [4]. Min-Max scaling was chosen because the features did not exhibit a normal distribution and were already close to a bounded [0, 1] range, making range-preserving normalization a natural fit for this dataset.

### C. Model Training and Parameter Tuning

Part of the dataset was set aside as unseen test data to prevent data leakage and ensure an unbiased evaluation of model performance. The dataset was split using a 50:50 train/test ratio with stratified sampling on the Fault Label, giving 5,500 training samples and 5,500 testing samples, each preserving the original class proportions (2,500 Normal, 1,000 of each fault class in both the training and testing partitions). A fixed random state (random\_state = 42) was used for the split and for every model, to ensure the results reported in this paper are reproducible.

Four classical ML models were trained and compared: Logistic Regression [7], Naive Bayes (Gaussian) [8], Random Forest [9], and XGBoost [5], which is built on the gradient boosting framework [10], each configured for multi-class classification across the four fault labels. Rather than performing hyperparameter tuning via grid search, each model was trained using its library's default hyperparameters, changing only the settings needed for reproducibility or to configure the model for a multi-class problem. The specific parameters used for each model are listed below.

Table 3: Logistic Regression parameters

| Parameter                           | Value | Note                                                                           |
|-------------------------------------|-------|--------------------------------------------------------------------------------|
| max_iter                            | 1000  | Increased from the library default of 100 to ensure the solver fully converges |
| solver                              | lbfgs | Library default                                                                |
| C (inverse regularization strength) | 1.0   | Library default                                                                |
| penalty                             | l2    | Library default                                                                |
| random_state                        | 42    | Fixed for reproducibility                                                      |

Table 4: Gaussian Naive Bayes parameters

| Parameter     | Value                                   | Note            |
|---------------|-----------------------------------------|-----------------|
| priors        | None (estimated from the training data) | Library default |
| var_smoothing | 1e-9                                    | Library default |

Table 5: Random Forest parameters

| Parameter         | Value | Note                                                    |
|-------------------|-------|---------------------------------------------------------|
| n_estimators      | 100   | Library default; number of decision trees in the forest |
| criterion         | gini  | Library default; splitting quality measure              |
| max_depth         | None  | Library default; trees expand until leaves are pure     |
| min_samples_split | 2     | Library default                                         |

| Parameter        | Value | Note                                                     |
|------------------|-------|----------------------------------------------------------|
| min_samples_leaf | 1     | Library default                                          |
| max_features     | sqrt  | Library default; number of features considered per split |
| bootstrap        | True  | Library default; trees trained on bootstrapped samples   |
| random_state     | 42    | Fixed for reproducibility                                |

Table 6: XGBoost parameters

| Parameter           | Value          | Note                                                                    |
|---------------------|----------------|-------------------------------------------------------------------------|
| n_estimators        | 100            | Library default; number of boosting rounds                              |
| max_depth           | 6              | Library default                                                         |
| learning_rate (eta) | 0.3            | Library default                                                         |
| booster             | gbtree         | Library default                                                         |
| gamma               | 0              | Library default; minimum loss reduction for a split                     |
| subsample           | 1.0            | Library default                                                         |
| colsample_bytree    | 1.0            | Library default                                                         |
| objective           | multi:softprob | Automatically selected for a 4-class problem                            |
| eval_metric         | mlogloss       | Set explicitly; multi-class logarithmic loss, appropriate for 4 classes |
| random_state        | 42             | Fixed for reproducibility                                               |

With the exception of Logistic Regression's max\_iter (raised to ensure convergence) and the random\_state/eval\_metric settings above, every model was left at its library defaults. As shown in Section IV, this was sufficient for all four models to achieve strong classification performance, indicating that extensive hyperparameter tuning was not necessary for this dataset.

#### IV. EXPERIMENTS AND RESULTS

##### A. Quantitative Analysis

Each of the 4 ML models was trained on the 5,500-sample training partition and evaluated on the held-out 5,500-sample test set. Since this is a 4-class classification problem (one normal class and 3 fault classes), the macro F1 score [11] was used as the primary metric, since it gives an unweighted average of precision and recall across all four classes, preventing strong performance on the majority Normal class from masking weaker performance on a fault class. The macro F1 scores for all 4 models are given in Table 7.

Table 7: Macro F1 scores on the NEV Fault Diagnosis test set

| Model               | Macro F1 Score (↑) |
|---------------------|--------------------|
| Random Forest       | 0.994              |
| XGBoost             | 0.992              |
| Naive Bayes         | 0.983              |
| Logistic Regression | 0.979              |

Random Forest is the strongest model, closely followed by XGBoost, though the gap between them is small. Notably, all 4 models, including the comparatively simple Logistic Regression and Naive Bayes classifiers, achieve macro F1 scores above 0.97, despite the class imbalance shown in Table 2, indicating that the fault classes in this dataset are highly separable given the available sensor features.

To verify that these results were not an artefact of a single train/test split, 5-fold cross-validation [12] was also performed for Random Forest and XGBoost, the two strongest models. The mean macro F1 scores across the 5 folds are given in Table 8.

Table 8: Mean macro F1 scores from 5-fold cross-validation

| Model         | Mean Macro F1 Score (↑) |
|---------------|-------------------------|
| Random Forest | 0.992                   |
| XGBoost       | 0.991                   |

The cross-validated results closely match the single-split results in Table 7 for both models, confirming that the high performance of both models is stable across different partitions of the data rather than a result of a favourable train/test split.

The processing speed of each model was also measured, using training time as a representative metric, since this is relevant to how practical a model would be to retrain periodically in an industrial deployment. Training times for Random Forest and XGBoost are given in Table 9.

Table 9: Training times for Random Forest and XGBoost

| Model         | Training Time (s) |
|---------------|-------------------|
| XGBoost       | 0.226             |
| Random Forest | 0.905             |

XGBoost trains roughly 4 times faster than Random Forest on this dataset. Combined with its near-identical macro F1 score, this makes XGBoost an attractive choice where retraining speed matters, while Random Forest offers a very slightly stronger result when training time is not a constraint.

Given its slightly superior and highly stable macro F1 score across both the single-split and cross-validated evaluations, Random Forest is treated as the best-performing model for this dataset and is used for the confusion matrix, agreement analysis, and SHAP analysis below.

Alongside macro F1, Cohen's Kappa [13] was used as a second, complementary evaluation metric. Unlike F1, which is derived purely from precision and recall, Cohen's Kappa measures the agreement between the model's predictions and the true labels after accounting for the agreement that would be expected by chance alone, making it a stricter check that a model's accuracy is not simply an artefact of class imbalance. The linear weighted variant of Cohen's Kappa [14] was used specifically because the fault classes have a natural ordinal-like relationship in severity (Normal, then increasingly severe fault categories), so linear weighting penalizes a prediction in proportion to how many classes away it is from the correct one, rather than treating every misclassification as equally severe. For the Random Forest model, this gave a Linear Weighted Cohen's Kappa of 0.9962. Under the conventional interpretation of Kappa, values above 0.8 already represent almost perfect agreement; a value of 0.9962 indicates that Random Forest's predictions agree with the true fault labels almost perfectly, far beyond what chance agreement alone (given the class distribution in Table 2) could produce.

The confusion matrix underlying both the macro F1 score and the Kappa statistic for Random Forest is given in Table 10, with rows representing the true class and columns representing the predicted class.

Table 10: Confusion matrix for Random Forest (test set,  $n = 5,500$ )

| Actual \ Predicted | Normal | Fault 1 | Fault 2 | Fault 3 |
|--------------------|--------|---------|---------|---------|
| Normal             | 2,500  | 0       | 0       | 0       |
| Fault 1            | 0      | 1,000   | 0       | 0       |
| Fault 2            | 0      | 0       | 999     | 1       |
| Fault 3            | 1      | 0       | 22      | 977     |

Of the 5,500 test samples, 5,476 (99.6%) are classified correctly. Every Normal and Fault 1 sample is classified correctly, and only a single Fault 2 sample is misclassified. Nearly all of the model's errors (22 of 24 total misclassifications) involve a true Fault 3 sample being predicted as Fault 2, with only two further single-sample errors elsewhere in the matrix. This shows that Random Forest's small number of mistakes are not spread randomly across the four classes, but are concentrated almost entirely in distinguishing Fault 2 from Fault 3, suggesting that these two fault categories produce more similar sensor signatures than the other class pairs, while Normal operation and Fault 1 are essentially never confused with any other class.

### ***B. Qualitative Analysis***

Beyond quantitative performance, it is important to understand which sensor features actually drive the Random Forest model's predictions, both to verify that the model has learned genuine, physically sensible relationships rather than spurious patterns, and to give maintenance engineers actionable information about which sensors are most diagnostic of a fault. As a first, simple view of this, Fig. 3 shows the Random Forest's built-in feature importances, averaged across all classes.

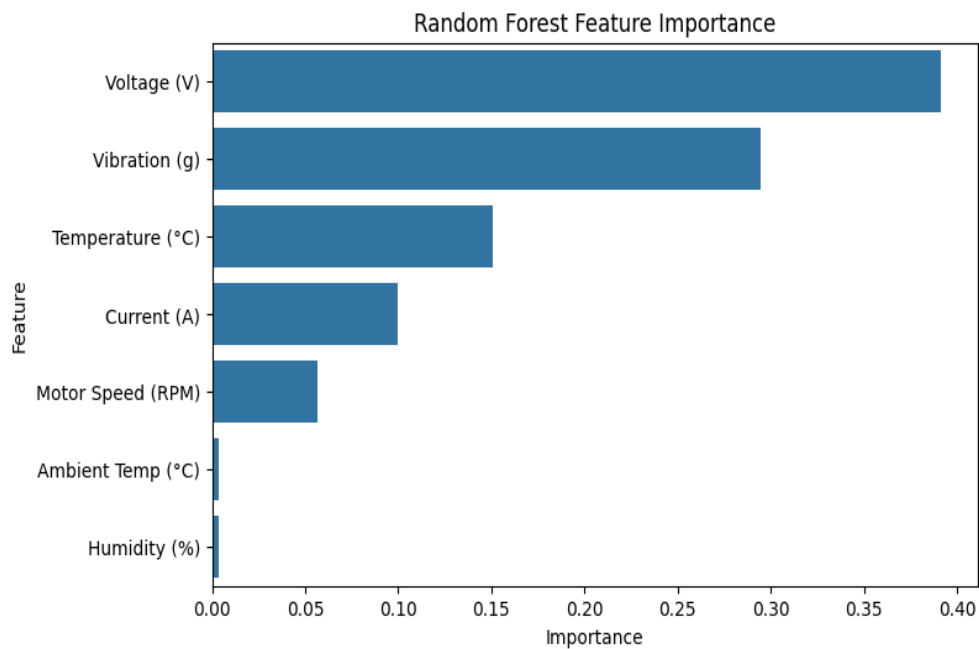


Figure 3: Random Forest feature importance, averaged across all classes

Table 11: Random Forest feature importance values

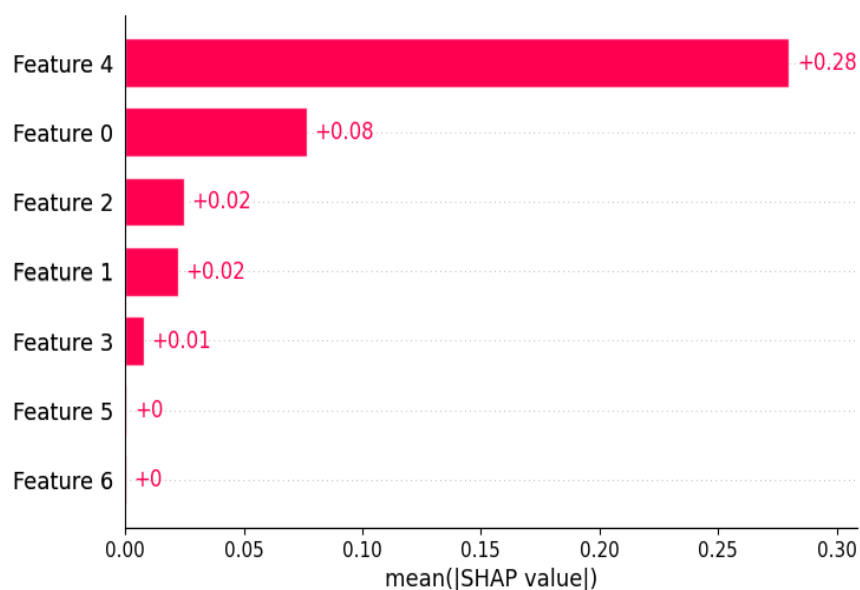
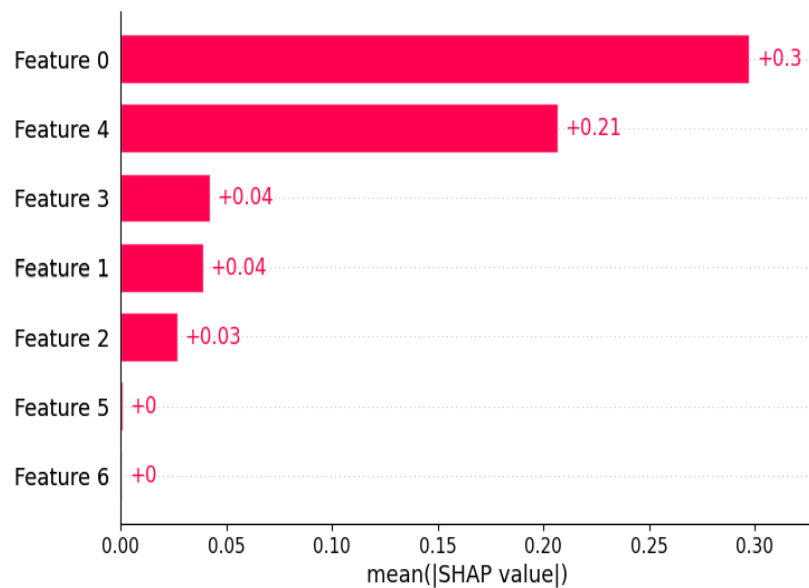
| Feature           | Importance |
|-------------------|------------|
| Voltage (V)       | 0.391      |
| Vibration (g)     | 0.294      |
| Temperature (°C)  | 0.150      |
| Current (A)       | 0.100      |
| Motor Speed (RPM) | 0.057      |
| Ambient Temp (°C) | 0.0035     |
| Humidity (%)      | 0.0034     |

Voltage and Vibration are clearly the two most influential features, together accounting for over two-thirds of total feature importance, followed by a more moderate contribution from Temperature. Current and Motor Speed contribute more modestly, while Ambient Temperature and Humidity contribute almost nothing to the model's decisions. This ordering matches the correlation matrix discussed in Section III.A almost exactly, where Voltage (-0.65), Vibration (0.58), and Temperature (0.29) showed the strongest relationships with Fault Label, while Ambient Temperature and Humidity were close to uncorrelated. This agreement between two independent analyses, one purely statistical (correlation) and one model-derived (feature importance), is a good sign that the Random Forest model is basing its predictions on the features that are genuinely informative, rather than on noise.

While feature importance gives a single, aggregated ranking, it does not explain the direction of each feature's effect, nor how its influence differs from one fault class to another, or from one individual prediction to the next. This is where SHAP (SHapley Additive exPlanations) becomes valuable. SHAP is a model-agnostic approach, derived from

the Shapley value in cooperative game theory [15], that assigns each feature a contribution value for a specific prediction: the change in the predicted outcome that can be attributed to that feature, found by averaging its effect across every possible ordering of the other features [6]. Because this contribution can be computed for any individual prediction and can be positive or negative, SHAP values reveal not just which features matter on average, but how each feature pushes a specific prediction toward or away from a specific class, making the model's behaviour interpretable at both the global (whole-dataset) and local (single-instance) level.

Since this is a 4-class problem, SHAP values were computed separately for each fault class using a one-vs-rest formulation: for every prediction, each class has its own set of SHAP values describing how each feature pushed the model toward or away from predicting that specific class. Fig. 4 shows the mean absolute SHAP value of each feature for each of the four classes, which is the SHAP equivalent of Fig. 3's aggregated feature importance, but broken down class-by-class.



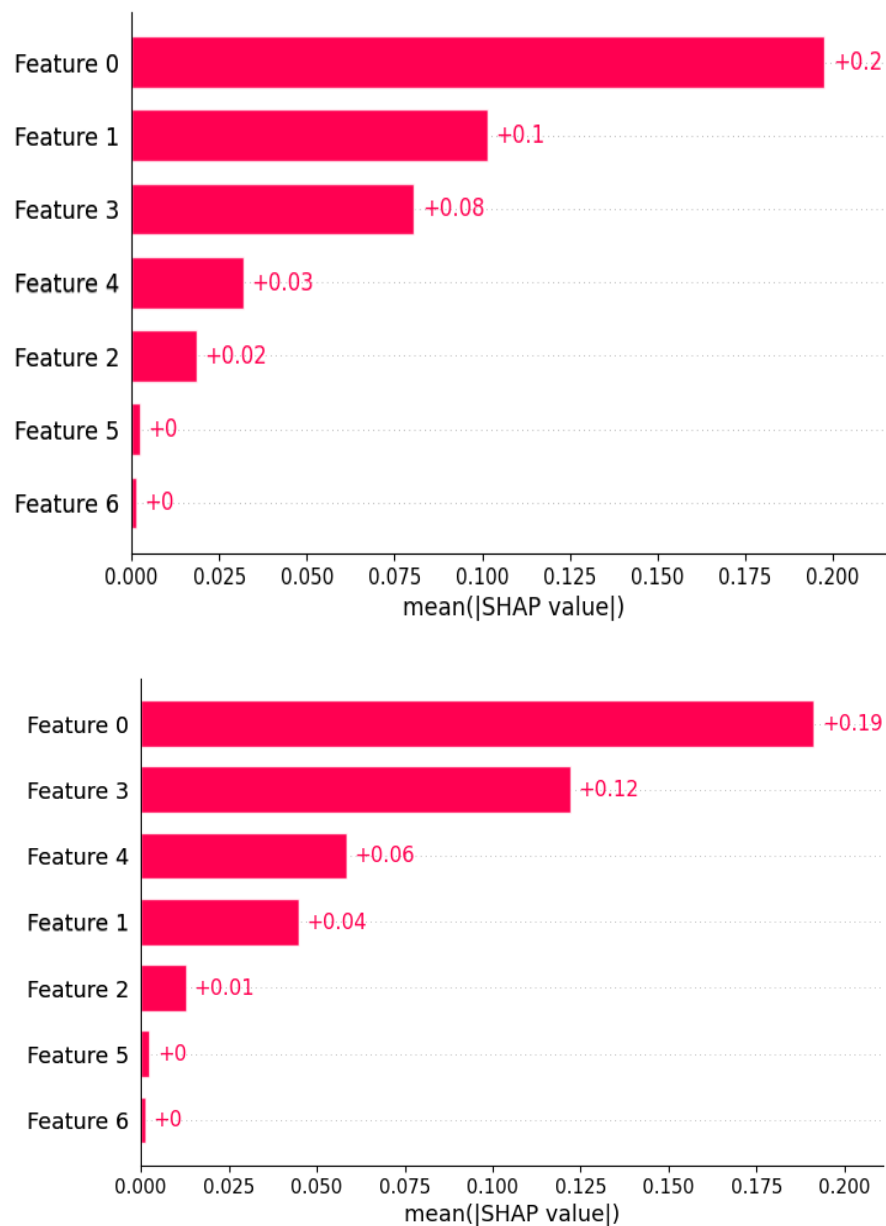
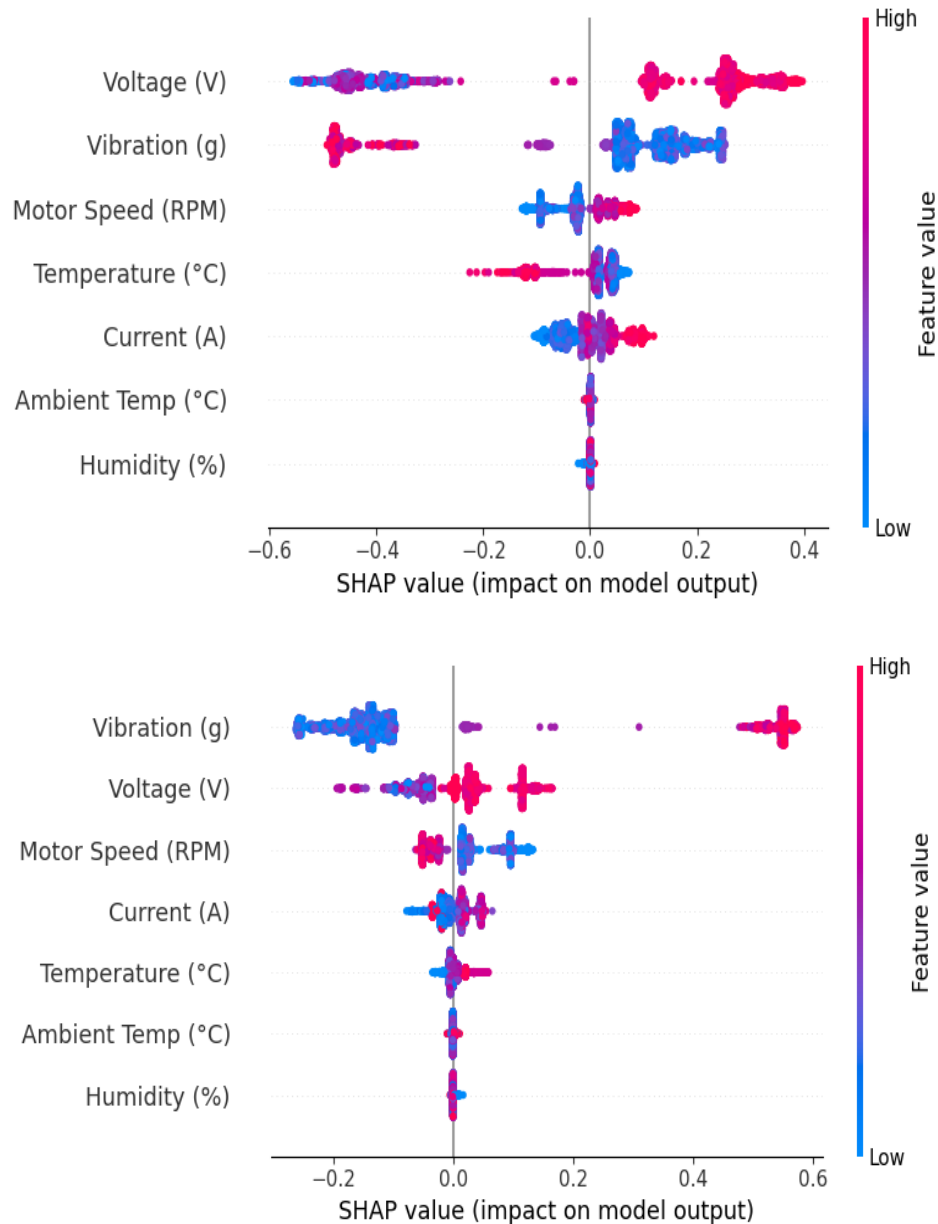


Figure 4: SHAP mean absolute value bar plots for Normal, Fault 1, Fault 2, and Fault 3 (Random Forest)

As with the aggregated feature importance in Fig. 3, Voltage and Vibration remain the dominant contributors across all four class-specific SHAP bar plots, with Temperature contributing a smaller but still visible amount, and Current, Motor Speed, Ambient Temperature, and Humidity contributing comparatively little. This class-by-class consistency indicates that the same small set of sensors is driving the model's decisions regardless of which specific class is being predicted, which is a reasonable and reassuring result: Voltage and Vibration readings are directly tied to the electrical and mechanical health of the drivetrain, and would be expected to shift in a detectable way whenever any fault condition develops, whereas Ambient Temperature and Humidity describe the external environment rather than the vehicle's internal state, and would not be expected to carry much diagnostic information regardless of class.

While the bar plots in Fig. 4 summarize feature importance per class, they do not show how a feature's value relates to its effect, for example, whether a high voltage reading pushes a prediction toward or away from a given class. SHAP

summary (“beeswarm”) plots address this by plotting the SHAP value of every test sample for every feature, with each point colored by that sample's feature value. This produces a distribution, rather than a single bar, for each feature, allowing both the magnitude and the direction of a feature's effect to be read at once. The summary plots for each of the four classes are shown in Fig. 5.



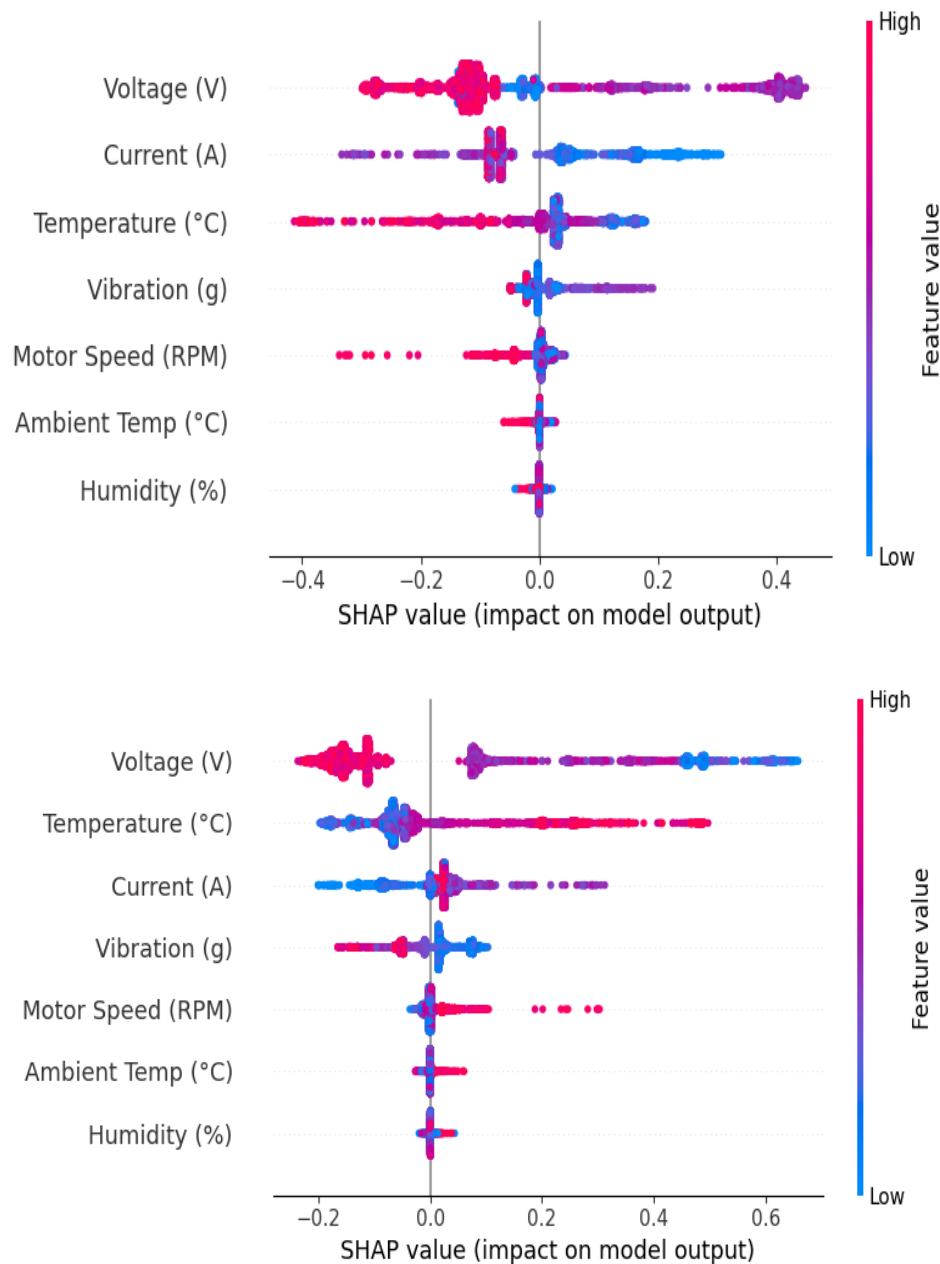
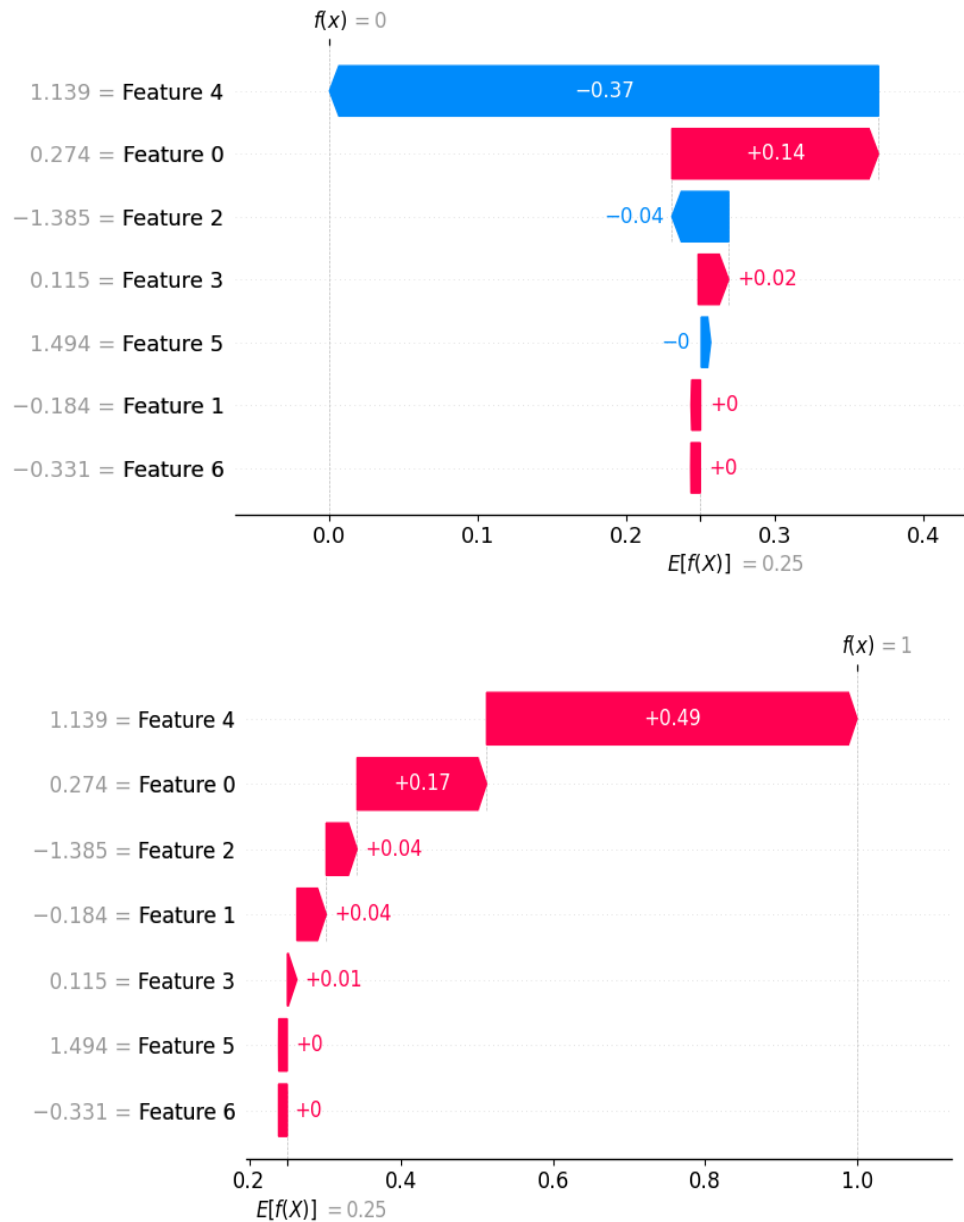


Figure 5: SHAP summary (beeswarm) plots for Normal, Fault 1, Fault 2, and Fault 3 (Random Forest)

The summary plots confirm that Voltage and Vibration are not only the most important features by magnitude, but also have a clear, consistent direction of effect within each class: samples with unusually high or low values of these two features (shown at the extremes of each feature's color range) tend to produce the largest SHAP values, meaning they most strongly push the model's prediction toward or away from that particular class. This is consistent with a predictive maintenance interpretation of the dataset, where a fault condition is expected to manifest as a voltage or vibration reading that deviates from its normal operating range in a fairly consistent direction for a given fault type, rather than an unpredictable or noisy relationship.

Finally, to illustrate how SHAP explains an individual prediction rather than the model's behaviour in aggregate, a single test sample was selected and its SHAP waterfall plot was generated for each of the four classes. A waterfall

plot starts from the model's base value, the average predicted value across the whole dataset for that class, and shows how each feature's SHAP value shifts the prediction, feature by feature, from this base value to the model's final output for that specific sample, with features pushing the prediction up shown in one color and features pushing it down shown in the other. The four waterfall plots for the selected sample are shown in Fig. 6.



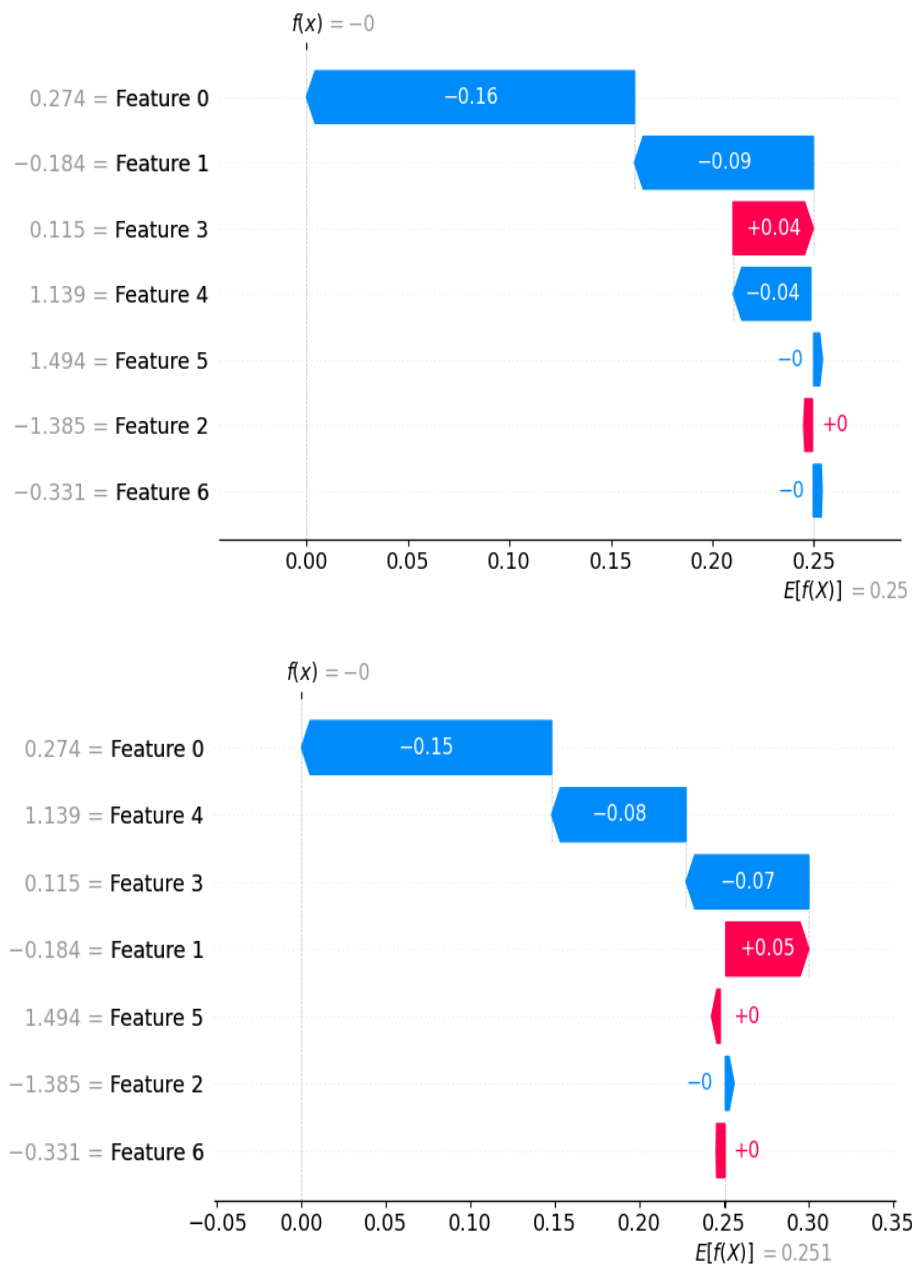


Figure 6: SHAP waterfall plots for a single test instance, across Normal, Fault 1, Fault 2, and Fault 3 (Random Forest)

For this individual sample, the same features that dominate the global analysis, principally Voltage and Vibration, are also the largest contributors to each class-specific prediction, again pushing the model consistently toward one particular class and away from the other three. This instance-level view is where SHAP is most directly useful for predictive maintenance in practice: rather than only knowing that a fault has been predicted, a maintenance engineer can see exactly which sensor readings were responsible for that specific prediction, and by how much, which can help confirm the diagnosis and direct maintenance effort toward the relevant part of the vehicle's drivetrain.

## V. DISCUSSION

### *A. Summary of Results and Explainability*

Four classical ML models, Logistic Regression, Naive Bayes, Random Forest, and XGBoost, were trained to classify electric vehicle sensor readings into one normal class and three fault classes. These four models were deliberately chosen to represent distinct, well-established learning approaches rather than variations on a single algorithm: Logistic Regression was included as a simple, linear, and highly interpretable probabilistic baseline; Naive Bayes as a second, even simpler probabilistic model that assumes feature independence, providing a useful sanity check on how much the models' more complex decision boundaries actually help; Random Forest as a bagging-based ensemble of decision trees, chosen for its strong general-purpose performance and its built-in feature importance measure; and XGBoost as a gradient-boosting ensemble, chosen as a stronger, typically faster-training alternative to Random Forest. All four models were trained identically, on the same Min-Max scaled sensor data using a stratified 50:50 train/test split with a fixed random seed, and evaluated using the same macro F1 score, so that the comparison between them isolates the effect of the model itself rather than differences in data preparation. Across every model, performance was very strong, with macro F1 scores between 0.979 and 0.994, and, for the best-performing model, a linear weighted Cohen's Kappa of 0.9962, indicating near-perfect agreement between predicted and true fault labels well beyond what chance alone would produce.

To understand why the models performed this well, and to make Random Forest's predictions interpretable rather than a black box, SHAP was used to analyze its predictions in detail. SHAP works by treating each prediction as a cooperative game between the input features, and computing each feature's fair share of the credit for pushing that prediction away from the model's average output, which makes it possible to explain not just which features matter overall, but how each one pushes an individual prediction toward or away from a specific class. This was used in three complementary ways in this paper. First, SHAP bar plots showed that Voltage and Vibration were, by a wide margin, the two most influential features for every one of the four classes, matching the Random Forest's own built-in feature importance ranking almost exactly. Second, SHAP summary (beeswarm) plots went a step further and showed the direction of each feature's effect, confirming that unusually high or low Voltage and Vibration readings consistently pushed predictions toward or away from a given class, rather than having an unpredictable or inconsistent effect. Third, a SHAP waterfall plot for a single test instance showed that this same pattern held at the level of an individual prediction, not just in aggregate, demonstrating how a maintenance engineer could use SHAP to see exactly which sensor readings justified a specific fault diagnosis. Together, these results, obtained by comparing the trained models' built-in feature importances against an independent SHAP analysis, and cross-checking both against the correlation matrix computed directly from the raw sensor data, gave a consistent picture: the models are basing their predictions on physically meaningful sensor signatures, principally Voltage and Vibration, rather than on noise.

### *B. Best Performing Model*

Comparing the four models directly, Random Forest achieved the highest macro F1 score (0.994) and the highest mean cross-validated macro F1 score (0.992), placing it marginally ahead of XGBoost (0.992 and 0.991 respectively) on both measures. On this basis, Random Forest is concluded to be the best-performing model for this dataset, and was accordingly used for the confusion matrix, Cohen's Kappa, and SHAP analysis presented in Section IV. However, the margin between Random Forest and XGBoost was small across every metric tested, while XGBoost trained roughly four times faster (0.226s versus 0.905s). This means the choice between the two is not clear-cut in every scenario: Random Forest is the better choice when squeezing out the highest possible accuracy and stability matters most, while XGBoost is the more practical choice in a deployment setting where models need to be retrained frequently and training time is a real constraint. Logistic Regression and Naive Bayes, while clearly behind the two ensemble methods, still performed respectably (0.979 and 0.983 macro F1 respectively), showing that even simple, highly interpretable models can achieve strong results on this particular dataset.

### C. Limitations

This paper's approach has several limitations that should be considered when interpreting its results. First, only a single dataset, the NEV Fault Diagnosis Dataset, was used; the models were not tested on sensor data from a different vehicle, manufacturer, or fault-generation process, so it is not yet known how well these results would generalize beyond this specific dataset. Second, 5-fold cross-validation was only performed for Random Forest and XGBoost, the two strongest models; Logistic Regression and Naive Bayes were evaluated on a single train/test split only, so their reported macro F1 scores are more susceptible to variance from that particular split than the cross-validated results, and it is possible that a different split could produce a noticeably different ranking between these two models. Third, the linear weighted Cohen's Kappa statistic was calculated only for the best-performing model, Random Forest; equivalent Kappa scores were not computed for Logistic Regression, Naive Bayes, or XGBoost, so a direct agreement-based comparison across all four models is not yet available. Fourth, no hyperparameter tuning was performed for any model; all four were trained using their library default parameters, so the performance figures reported in this paper represent a baseline rather than an upper bound on what each model could achieve. Finally, the dataset itself proved to be highly separable, with every model achieving a macro F1 score above 0.97; while this made for a clean set of results, it also means this paper cannot say how the same models and methodology would perform on a noisier, less separable predictive maintenance dataset, which may be more representative of real-world sensor data collected outside of a controlled or synthetic setting.

## VI. CONCLUSION

This paper set out to determine whether classical machine learning models could reliably classify electric vehicle sensor readings into normal operation and multiple distinct fault categories, and to make those models' predictions explainable rather than opaque. All four models tested, Logistic Regression, Naive Bayes, Random Forest, and XGBoost, achieved strong classification performance, with Random Forest the best-performing overall and XGBoost a close, faster alternative. The SHAP-based qualitative analysis showed that this performance is driven primarily by Voltage and Vibration readings, both at the level of the dataset as a whole and for individual predictions, giving confidence that the models have learned genuine, physically grounded fault signatures rather than spurious correlations. Taken together, these results suggest that classical ML-based fault diagnosis, paired with SHAP for explainability, is a promising and practical foundation for predictive maintenance in electric vehicles, though the limitations discussed in Section V indicate that further work, particularly testing on additional datasets, extending cross-validation and Cohen's Kappa to all four models, and performing a more complete hyperparameter search, would be needed before such a system could be relied upon in a real deployment.

## VII. ACKNOWLEDGEMENT

This piece of work holds no connection with Jumeirah College and is done in pursuit of higher education.

## REFERENCES

1. HariKrishna, S. V. N., Mandal, S. K., & Ganganna, P. (2025). Intelligent fault diagnosis of electric vehicle powertrains using machine learning techniques. *International Journal of Engineering Research and Sustainable Technologies (IJERST)*, 3(4), 23–33. <https://doi.org/10.63458/ijerst.v3i4.135>
2. Wang, L., & Wu, M. (2025). Research on bearing fault diagnosis based on machine learning and SHAP interpretability analysis. *Scientific Reports*, 15, Article 41242. <https://doi.org/10.1038/s41598-025-25083-4>
3. adamak09. (n.d.). *New energy vehicles diagnosis Adam* [Kaggle notebook]. Kaggle. <https://www.kaggle.com/code/adamak09/new-energy-vehices-diagnosis-adam>
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., &

- Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
5. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). Association for Computing Machinery. <https://doi.org/10.1145/2939672.2939785>
  6. Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems* (Vol. 30).
  7. Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression* (3rd ed.). John Wiley & Sons.
  8. Rish, I. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, No. 22, pp. 41–46).
  9. Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
  10. Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232. <https://doi.org/10.1214/aos/1013203451>
  11. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427–437. <https://doi.org/10.1016/j.ipm.2009.03.002>
  12. Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (Vol. 2, pp. 1137–1143). Morgan Kaufmann.
  13. Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37–46. <https://doi.org/10.1177/001316446002000104>
  14. Cohen, J. (1968). Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4), 213–220. <https://doi.org/10.1037/h0026256>
  15. Shapley, L. S. (1953). A value for n-person games. In H. W. Kuhn & A. W. Tucker (Eds.), *Contributions to the theory of games II* (pp. 307–317). Princeton University Press.