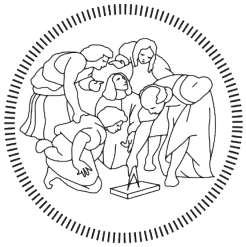


POLITECNICO
MILANO 1863

Network Measurement and Data Analysis Lab

Final Project 2023-2024

Optical Failure Localization

Moein Taherinezhad - 10935476
Giulia Gerometta - 10628256
Niloufar Sasannia - 10888547

July 8, 2024

Table of Contents

1 Problem presentation

This project aims to localize soft failures in an optical network comprising four nodes connected in series, creating three links (Figure 1.1). Each node has a monitor at its input port that provides OSNR (Optical Signal-to-Noise Ratio) samples. A fault can happen at any of the three links, and our objective is to determine the faulty link. Hence, we use the OSNR values to determine key features, and train machine-learning algorithms to localize the fault. Moreover, to be aware of the correctness of ML-based decisions, we used Explainable AI (XAI). XAI techniques allow uncovering the reasoning behind the ML model to a human expert, making ML models more trustable and, hence, more likely to be adopted practically.

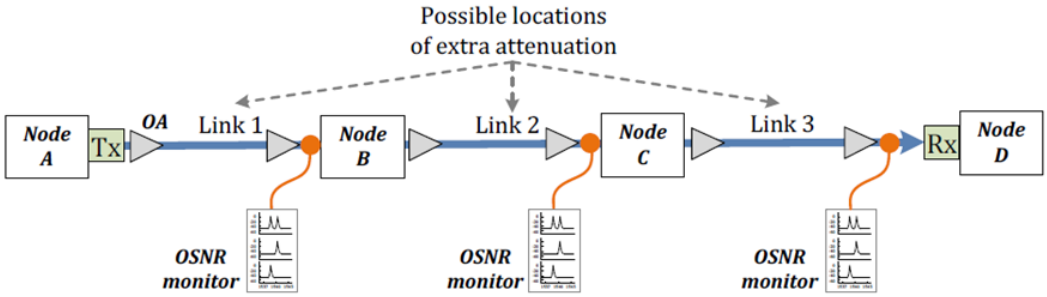


Figure 1.1: Optical network configuration

2 Data analysis and visualization

Our dataset comprises nine files containing observed data from the three monitors. Hence, we will standardize the OSNR values to reduce bias, boosting our model's predictive performance. A particular file contains data of a fault at a specific link x (with x = 1, 2, and 3) observed by monitor y (with y = B, C, and D).

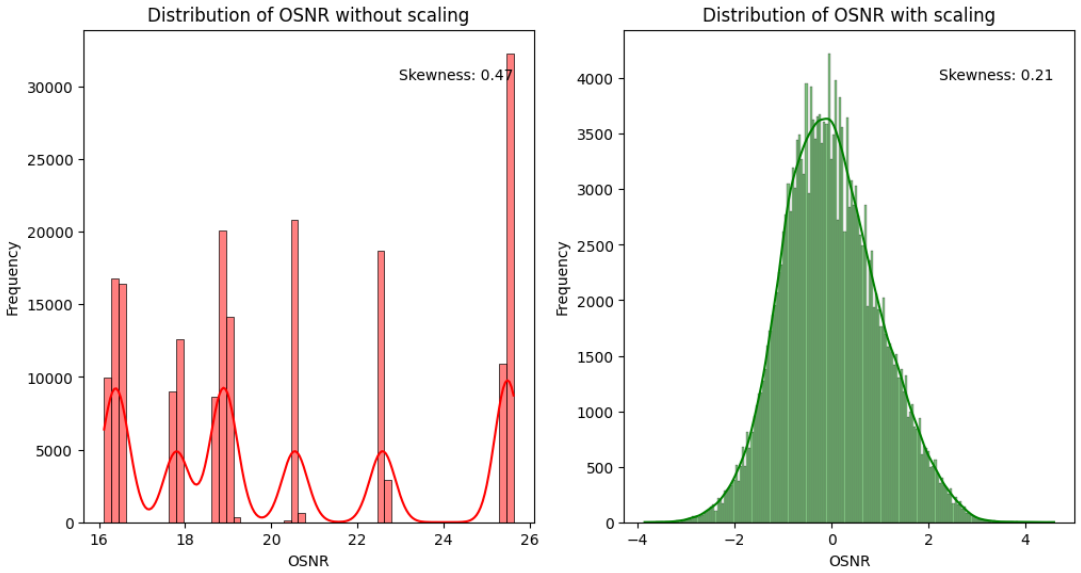


Figure 2.1: Histograms of OSNR values with and without scaling

Once we have assembled the complete data frame, we can analyze our dataset. Observing the OSNR distributions (Figure 2.1), we can see multiple peaks in the plot with absolute OSNR values: each link has unique characteristics, leading to varying OSNR measures. The distribution with distinct peaks suggests that the data clusters around specific OSNR values rather than being smoothly distributed. On the other hand, the scaled dataset closely approximates a normal distribution. Indeed, we can appreciate the classical bell-shaped plot of a normal distribution. The scaling process transformed the raw OSNR data from a multimodal moderately skewed distribution into a near-normal distribution (skewness equal to 0.21). Skewness is a statistical measure that describes the asymmetry of a distribution around its mean. Our dataset's skewness values (0.21 and 0.47) reveal that both distributions are moderately symmetrical since both have a slight positive skew. However, the scaled dataset provides a more homogeneous data distribution. Moreover, statistical analysis and machine learning algorithms assume normally distributed data. Indeed, since we decided to use logistic regression, XGBoost, and LSTM machine learning models for classification, a scaled dataset ensures better performance and faster convergence during training.

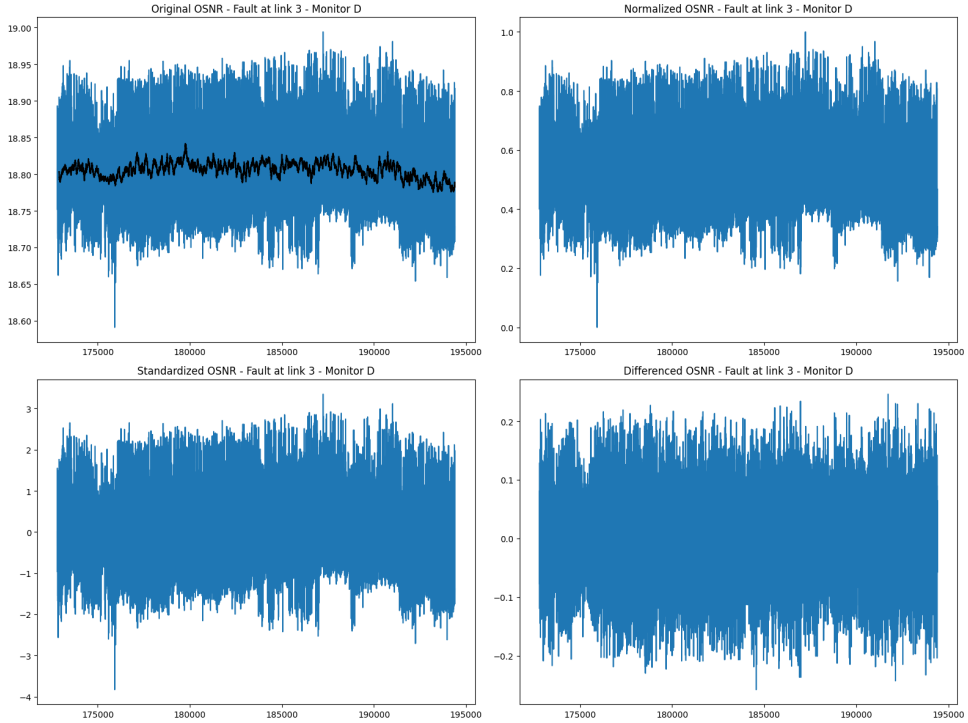


Figure 2.2: Normalization vs Standardization vs Differencing

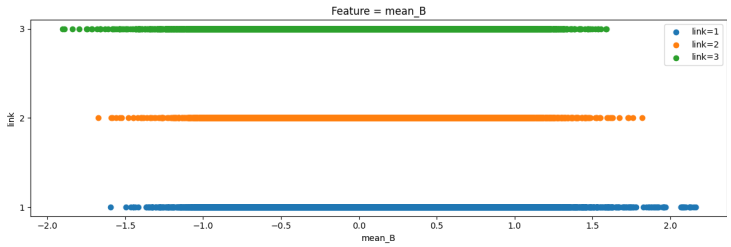
At this point, we analyze the impact of Normalization, Standardization, and Differencing techniques on a subset of our dataset (fault at link 3 observed by monitor D). The top left plot represents the original OSNR values without any preprocessing. The top right plot shows normalized OSNR measures. As we can see, normalization retains the shape and relative distribution of the original data. The bottom left plot displays standardized OSNR values. It highlights outliers and deviations from the mean more clearly. Moreover, standardization is usually employed in machine learning algorithms that assume normally distributed data. The bottom right plot illustrates the OSNR values after differencing. However, differencing removes trends and seasonality, highlighting changes from one point to the next. Indeed, differencing appears more robust against outliers. These charts show an outlier that significantly impacts the dataset. Only standardization and normalization maintain the trend. Since we need to apply machine learning techniques, we will employ standardization.

3 Feature extraction and Analysis

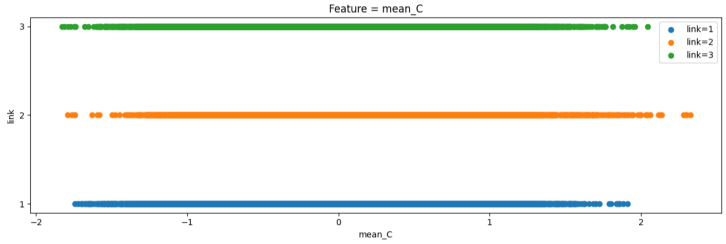
At this point, we extract statistical features (Mean OSNR, Max OSNR, Min OSNR, Peak-to-Peak OSNR, Skewness, Kurtosis) from OSNR values grouped into same-size windows. Windows are of duration

window_size, including *window_size* consecutive OSNR samples, spaced of a specific value from one to another. Hence, we built our dataset computing features from OSNR values using a sliding window approach. In this work, we will explore different window sizes with different scaling.

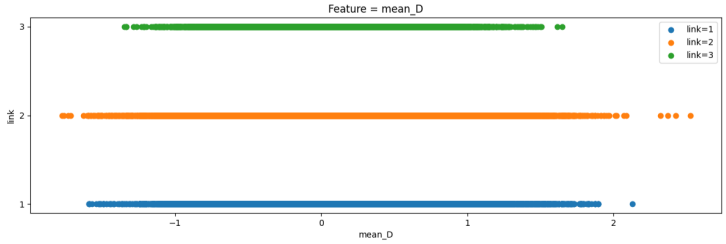
Therefore, we perform feature analysis to help us understand the dataset structure for feature selection suitable for machine learning applications. We plotted scatter plots for each feature, with feature values on the x-axis and the link on the y-axis. In general, features having non-overlapping plots will indicate a good discriminative power. On the other hand, a feature scatter plot representing coinciding values in the three links suggests that this feature is not valuable for classification. Moreover, even if the presence of outliers can introduce a discriminative value for a specific link, it also introduces noise. Indeed, a wider spread indicates a vaster variability in that feature. While features with a more concentrated distribution imply more consistency in the values. Here, the ideal case would be a concentrated distribution of the single features in a specific link but with a non-overlapping scatter plot from one link to another.



(a) Mean of the OSNR values collected from monitor at node B.



(b) Mean of the OSNR values collected from monitor at node C.



(c) Mean of the OSNR values collected from monitor at node D.

Figure 3.1: Visualization of data distributions using scatter plots for each feature

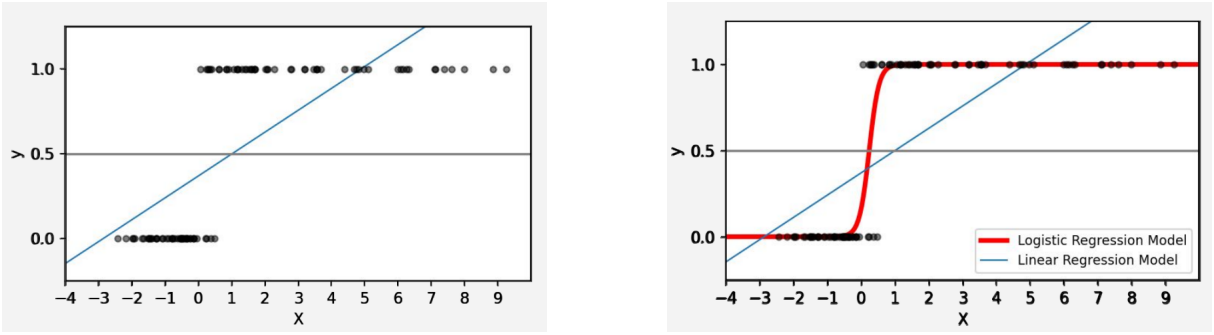
In Figure 3, we gathered just the mean features computed from data collected from monitors B, C, and D, respectively. For most features, scatter plots of the different links overlap, meaning that a single feature is not discriminative for classification. For this reason, we must use machine-learning algorithms to deliver a more comprehensive analysis, including all the features. Indeed, multiple features collectively contribute to the classification. Machine learning models can consider all the features simultaneously to produce accurate predictions.

4 ML Background

We modeled our failure-localization problem as a supervised multi-class classification problem. Hence, in this section, we will explain briefly the ML models we selected to localize faults.

4.1 Logistic Regression

In this project, we used a logistic regression supervised machine learning algorithm with k-fold cross-validation to classify the faulty link. Logistic regression is a statistical method employed in binary classification. In opposition to linear regression, which uses a linear function to discriminate classes (Figure 4.1a), logistic regression is based on the logistic (sigmoid) function (Figure 4.1b). Logistic regression can be applied to solve multiclass problems. In particular, we used multinomial logistic regression. This approach handles multiple classes with a single model, providing a probability distribution over all classes. Moreover, we used the k-fold cross-validation technique to evaluate the performance of the logistic regression model. This methodology proposes to split the data into k subsets (folds). Hence, the model is trained k times, using a different fold as the validation set and the remaining folds as the training set. At this point, we average the performance metrics over all k iterations to provide a more robust estimate of model performance.



(a) Linear Regression

(b) Logistic Regression

Figure 4.1

4.2 XGBoost

Decision Trees, used for regression and classification, are characterized by excellent interpretability. However, Decision Trees, compared to linear/logistic regression methodologies, suffer from significantly low prediction accuracy limitations. XGBoost combines many Decision Trees to overcome accuracy limitations. XGBoost (Extreme Gradient Boosting) is a powerful and efficient implementation of gradient boosting algorithms designed for speed and performance. XGBoost boosting technique sequentially adds Decision Trees to form a strong learner. Each subsequent model corrects the errors of the previous models.

4.3 LSTM

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) usually employed in sequence prediction problems. LSTM networks aim to resolve the limitations in traditional RNNs to remember information for long periods. Theoretically, classic RNNs can retain long-term dependencies in the input sequences. However, when training a classic RNN using back-propagation, the long-term back-propagated gradients can vanish (they can tend to zero), causing the model to stop learning. RNNs using LSTM partially solve this problem. The LSTM network creates an additional module in the NN that learns when to remember and when to forget pertinent information. Hence, the network effectively learns which data might be needed later in a sequence and when that information is no longer needed. In particular, LSTM networks have memory cells that remember values over arbitrary time intervals. Each cell consists of three gates regulating the information flow into and out of the cell: input, forget, and output gates. The input gate controls the extent to which new information flows into the cell. The forget gate determines what portion of the information from the previous time step should be discarded. The output gate regulates the information that will be passed to the next time step. In conclusion, LSTM networks can capture long-term temporal dependencies in sequential data. We exploit this LSTM property to identify failure patterns over time in optical networks.

4.4 Explainable AI

Machine learning models are complex black boxes, and considering just the model's high accuracy is not enough. Operators want to understand ML reasoning to trust its decisions in critical scenarios. Explainable AI moves beyond interpretable AI, providing insights into how the ML model arrived at the result. The explanations can be local or global, explaining a decision for a specific data instance or decisions towards a particular class, respectively. Here, we are interested in global explanations since we want to investigate the model's predictions for each class. In particular, we apply XAI to understand the reasoning of ML models (Logistic regression, XGBoost) in the failure localization problem. We used the SHAP framework to estimate the contribution of each feature (input) to model prediction (output). SHAP is based on Shapley value, a game theory concept that quantifies the contributions of different players to the total payoff. In this context, we have features instead of players in a game. For each model's prediction, Shapley values represent the contribution of each input feature to that specific prediction. Hence, for a given instance where a failure is predicted, you can see which OSNR features (and from which monitor) drive the model towards a specific prediction.
In the following sections, concerning the four different scenarios, we will use a bar plot to give a global SHAP summary showing the impact of each feature on the model's output across the entire test set. Then, we will generate a SHAP summary plot for each link. The global SHAP summary plot shows insights into the impact of each feature on the model's output. This plot arranges on the y-axis the features ordered by their effect on the prediction. A bigger SHAP value, hence the higher the feature is on the y-axis, denotes a more significant impact on the model's output. Each bar is divided into colored segments representing the influence of the feature on the three different classes (the three links). The SHAP summary plot combines feature importance with feature values to explain the ML model's behavior specifically for the monitor in a specific link. Features are listed vertically from top to bottom in order of their impact on the model's output. The horizontal axis holds the Shapley value scale. Each point of the summary plot is a Shapley value for a given feature in a given data sample. Red dots indicate that a high value of a feature contributes to the prediction that a fault happens at a link (positive SHAP) or not (negative SHAP). Blue dots denote that a low feature value contributes to predicting whether a fault occurred or not at a specific link.

5 Scenarios

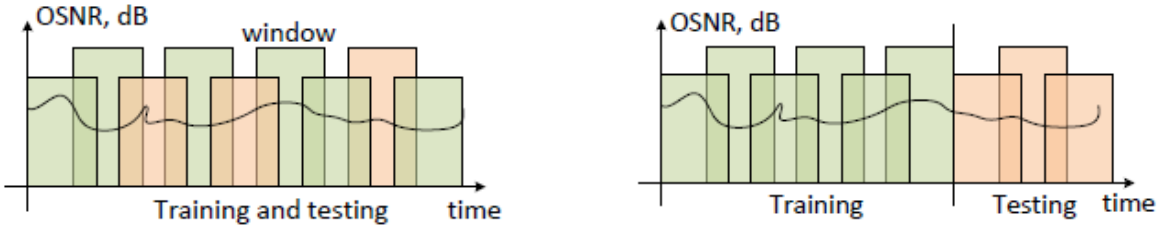
In this section, we will analyze four scenarios: using data from a single monitor with random and in-time splits and multiple monitors with random and in-time splits (summary in Table 5.1).

Table 5.1

Monitor/Splitting	**Random**	**In-time**
Single	Scenario 1	Scenario 2
Multiple	Scenario 3	Scenario 4

A random split divides the dataset randomly into train and test sets, as in figure 5.1a. In this case, each data point has an equal chance of being included in either the training or testing set, regardless of its position in the sequence. The random split method assumes that all the data points are independent and identically distributed. Hence, it does not take into account temporal dependency. On the other hand, the in-time split approach divides the dataset in temporal order, as in figure 5.1b. In our case (and in most cases), we used earlier data points for training and later data points for testing.

Since our dataset has data taken every second from each monitor, with this method, we can respect the chronological order of the data. In our case, a time split could be necessary to investigate if the failures and OSNR values are temporally dependent. Indeed, it may happen that past failures influence future failures.



(a) Random splitting

(b) In time splitting

Figure 5.1: Train test splitting

For brevity, we will report only the multiple monitor scenarios (3-4) since they have a comprehensive view of the whole network under investigation. Moreover, even if we also experimented with a single monitor setup, it was already demonstrated by Karandin et al.[1] that additional monitors not only improve fault-localization accuracy but also significantly enhance the explainability of the ML model reasoning. Hence, we will focus on the analysis with multiple monitors only. Moreover, we will document only results with the highest accuracy, hence the ones with a window size of 100 OSNR values with a spacing of 25 data points. In Table 5.2, we collected all the best test accuracy results of all the experiments we carried out.

Table 5.2

Model	**Split type**	**Window size**	**Accuracy**
Logistic Regression	Random	10	0.667
Logistic Regression	Random	100	0.925
XGBoost	Random	10	0.976
XGBoost	**Random**	**100**	**0.988**
LSTM	Random	10	0.685
LSTM	Random	100	0.679
Logistic Regression	In-time	10	0.599
Logistic Regression	In-time	100	0.795
XGBoost	In-time	10	0.874
XGBoost	In-time	100	0.855
LSTM	In-time	10	0.687
LSTM	In-time	100	0.789

As we can see from Table 5.2, our models perform better with random splitting, meaning that our dataset does not have significant temporal dependencies or trends. Random splitting mixes data from different time periods, ensuring a more homogeneous and representative training set. Moreover, it reduces the temporal bias, lowering the risk the model is learning from a specific subset of data. The biased training set may contain trends and patterns relevant to a certain time period.The best accuracy result falls in the XGBoost, window size = 100, random splitting case.

5.1 Scenario 3 - Multiple monitor, random split

5.1.1 Logistic Regression

Accuracy, in this case, has a high value (0.925). Indeed, as we can see from the confusion matrix (Figure 5.2), the majority of classes are well-predicted, and there is no significant prediction error.

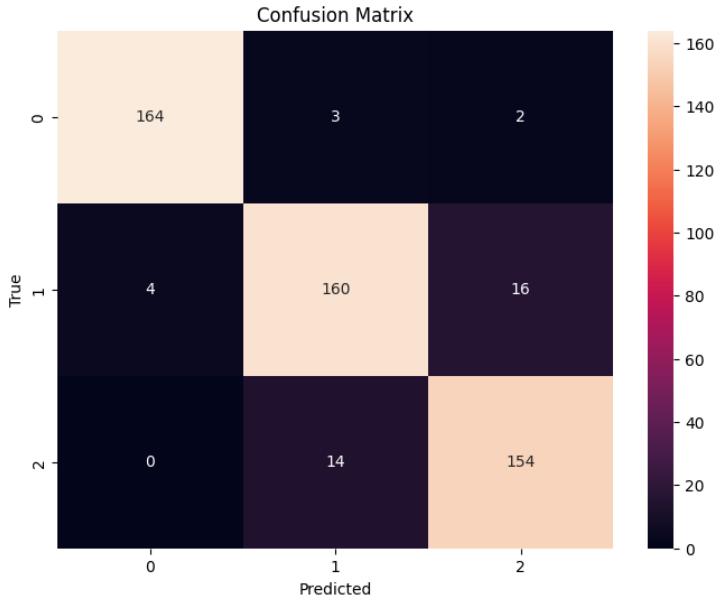


Figure 5.2: Confusion Matrix

This SHAP global summary plot in Figure 5.3 provides insights into the feature relevance for our best logistic regression model with random splitting. The top features, hence the most important ones, are *std_B*, *kurtosis_B*, and *peak_to_peak_B*. These latter features are the most influential across all classes (failures at links 1, 2, and 3, respectively). Hence, the values collected at monitor B, placed at the end of the first link, are the most significant. Since features collected from monitor B dominate, we can conclude that this monitor's data is crucial in predicting failures across all links. The colored bars (blue, green, and pink) show the impact of the feature in predicting a failure in links 1, 2, and 3, respectively. As we expected, *std_B*, *kurtosis_B*, and *peak_to_peak_B* have a substantial impact on predicting failures in the first link. However, they show a significant influence also in foreseeing failures on the second and third links. We can conclude that features derived from monitor B are paramount in the detection of failures on all links. We can guide network monitoring and maintenance enhancing efforts on this monitor, giving closer attention to it.
Figure 5.4a, 5.4b and 5.4c show the summary plots produced by SHAP to explain the decision in fault localization on links 1, 2, and 3, respectively. The summary plot incorporates feature importance with its SHAP value The horizontal axis holds the SHAP value scale, while the vertical axis has features organized by their importance. Features that drive fault localization in a specific link have many points with high SHAP values and are placed at the top. In Figure 5.4a, 5.4b and 5.4c, we can see that *std_B* is the key feature for all three links. However, from Figure 5.4a, we know that a high value of it contributes to the prediction that the fault is at the first link. While for links 2 and 3, we can say the opposite. Hence, a high value of *std_B* does not contribute to the localization of a fault at the second or third link. As we can see from the plots, the prediction is very similar for links 2 and 3. The discrimination between the two resides in the *peak_to_peak_C* feature that is crucial in the localization of fault in the third link.

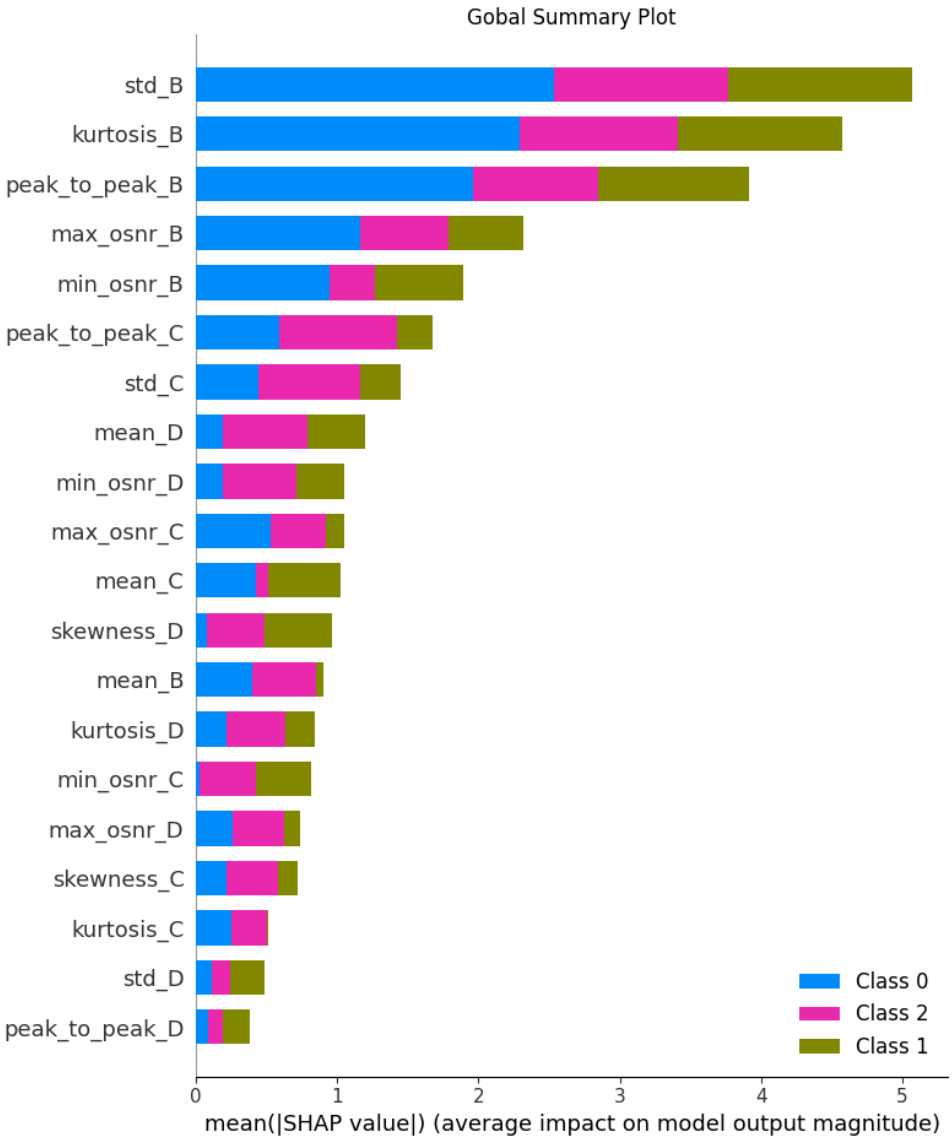
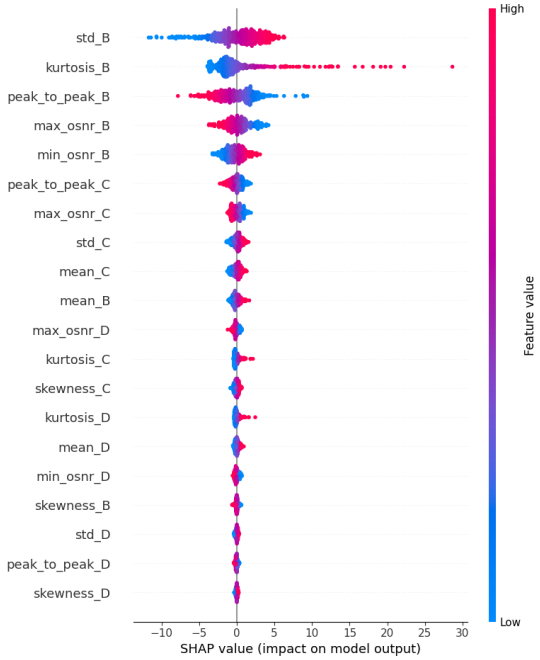
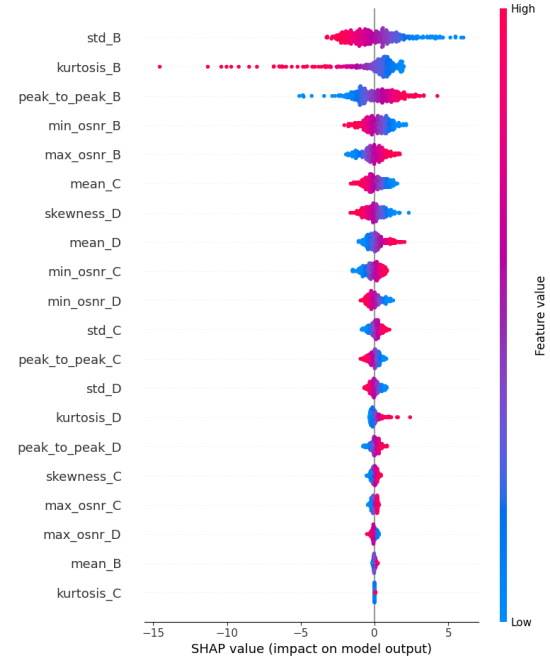


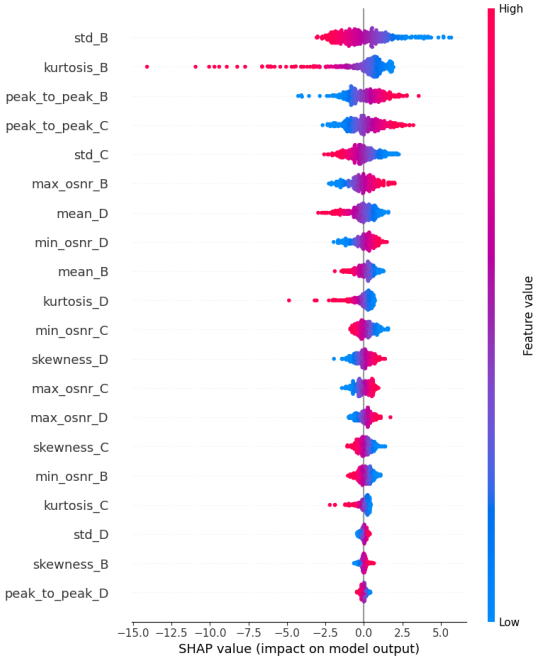
Figure 5.3: Global SHAP Summary plot



(a) Link 1



(b) Link 2



(c) Link 3

Figure 5.4: SHAP Summary plot

5.1.2 XGBoost

Our XGBoost model with random splitting has an accuracy of 0.988. Indeed, as we can see from the confusion matrix in Figure 5.5, most of the classes are well-predicted with just a few mispredictions.

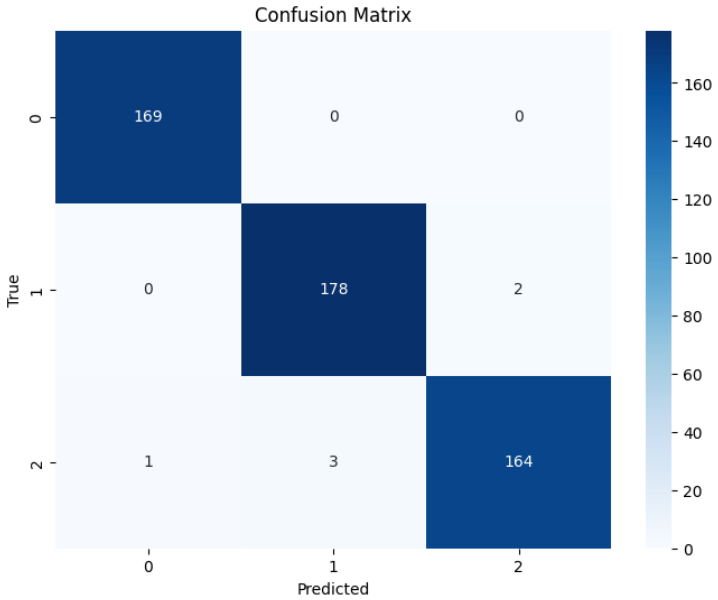


Figure 5.5: Confusion Matrix

The SHAP global summary plot in Figure 5.6 highlights different dominant features from the previous case. Indeed, the most important features here are *peak_to_peak_B* and *peak_to_peak_C*, which are the values of peak_to_peak collected at monitor B (placed in link 1) and at monitor C (placed in link 2), respectively. Other impactful features are *kurtosis_B*, *mean_D*, *skewness_D*, *skewness_B* and *min_osnr_D*. In this case, features from all monitors play significant roles, indicating that this model considers information from multiple links to make accurate predictions. In particular, the *peak_to_peak_B* feature significantly influences class 0 (i.e., failures at link 1). While for class 1 (i.e., failures at link 2), impactful features belong to different monitors (*kewness_D* and *peak_to_peak_B* and). Finally, class 2 (i.e., failures at link 3) shows considerable influence from features collected from the last node (*mean_D*, *skewness_D*, and *min_osnr_D*). As intuition would suggest, in this case, the most influencing features are related to OSNR measured by the monitor located at the end of the faulty link. In particular, our model predicts faults at the first link with OSNR values collected from the monitor at the end of this link (monitor B). We can say the same for the faults at the last link, which come from values collected at monitor D. However, the most relevant features to localize fault at link 2 belong to monitors B and D. Hence, to classify failure at link 2, the model relies on measurements from monitors placed at the previous and next link from the fault, suggesting that our model is building a proof of contradiction.
In this case, the SHAP summary plot explaining predictions of faults in the first link (5.7a, 5.7b, and 5.7c) shows clearly that high values of *peak_to_peak_B* do not contribute to prediction, while low values of the same feature represent a failure in the first link. In predicting third-link failures, low values of *mean_D* and *kurtosis_D* play a crucial role. At the same time, low values of *peak_to_peak_C* indicate that the fault did not happen in the third link. As mentioned, our model seems to be building a proof of contradiction. In this case, our model predicts faults at the second link based on features collected on the monitor in the following link. In particular, a high value of *skewness_D* indicates a failure is happening at the third link, but a high value of this latter feature is a crucial indicator of faults in the second link. Moreover, while a high value of *min_osnr_D* indicates a fault in the third link, it is also essential to point out that a failure is not at the second link.

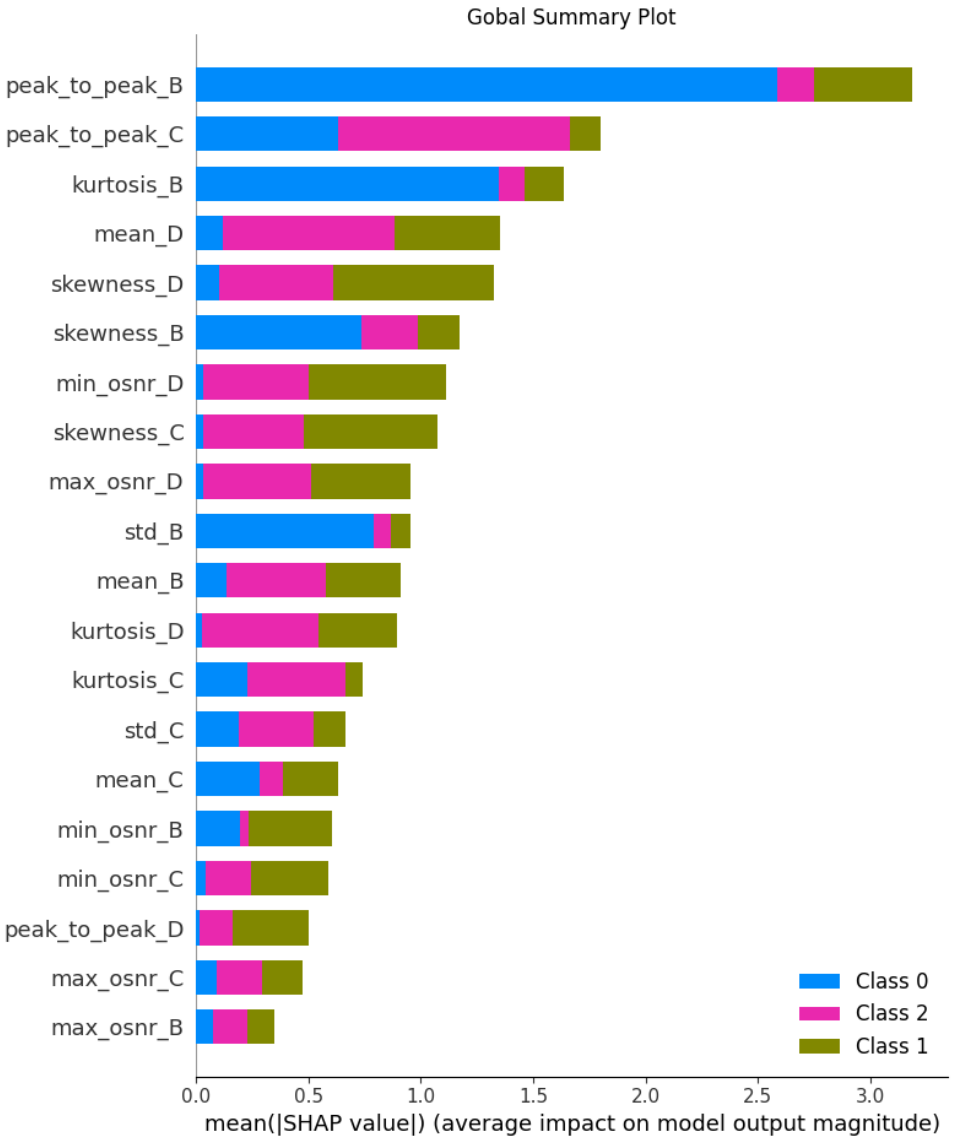
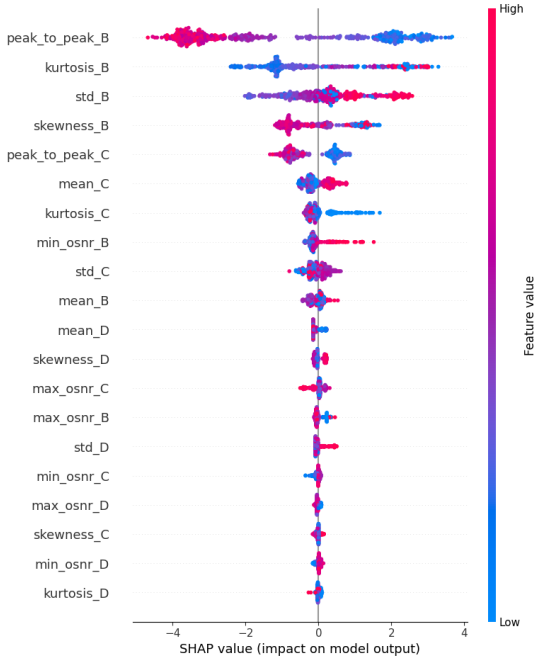
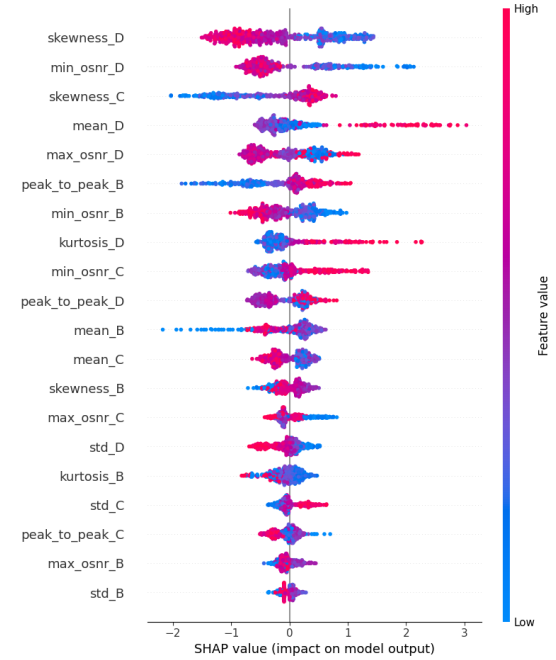


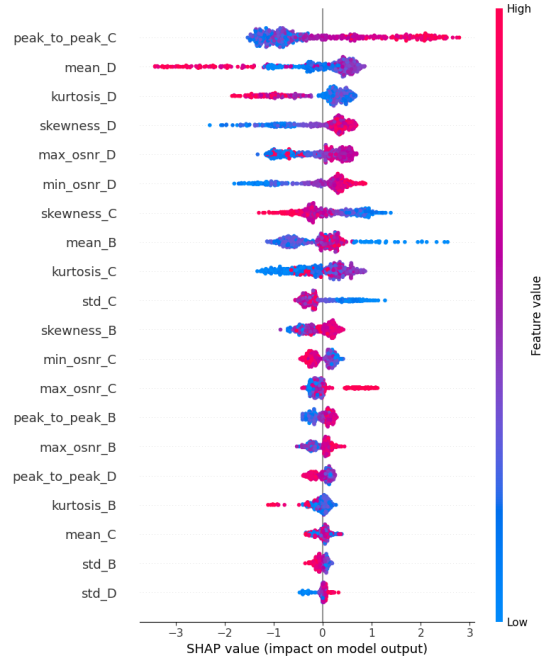
Figure 5.6: Global SHAP Summary plot



(a) Link 1



(b) Link 2



(c) Link 3

Figure 5.7: SHAP Summary plot

5.1.3 LSTM

We also experimented with the LSTM model. However, the results were not exciting. Indeed, we obtained an accuracy of 0.679, which is moderately low in comparison with the previous models. Indeed, we decided not to further investigate the model explanation since we would not employ the LSTM model in this context.

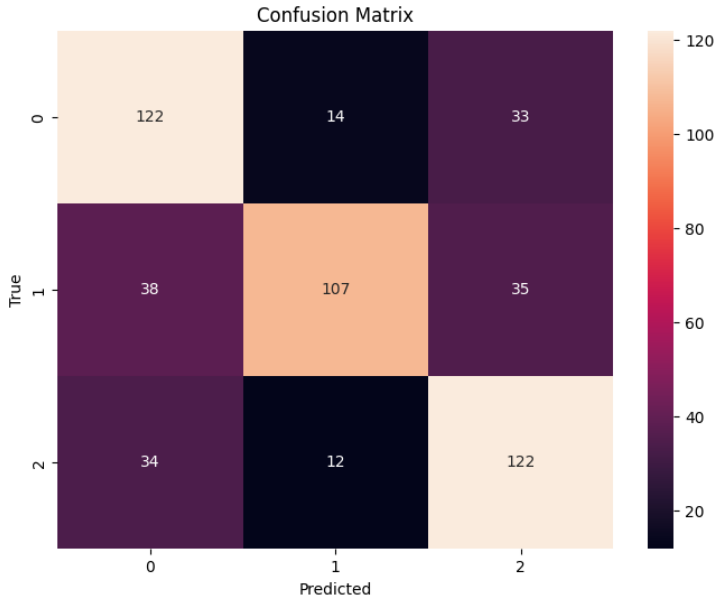


Figure 5.8: Confusion Matrix

5.2 Scenario 4 - Multiple monitor, in-time split

5.2.1 Logistic Regression

In this case, the model's accuracy is 0.795. Indeed, from the confusion matrix in Figure 5.9, we can see that the majority of the classes are well-predicted. However, the matrix shows impactful mispredictions of class 1; the model predicted 55 data entries, originally belonging to class 2, to be of class 1. In other words, our model predicts 55 failures in the third link to be failures of the second link.

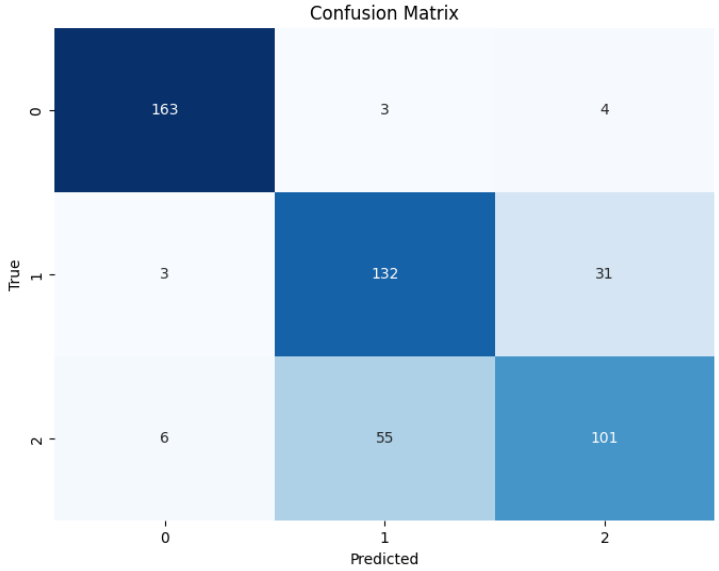


Figure 5.9: Confusion Matrix

Figure 5.10 shows the SHAP global summary plot explaining the logistic regression model trained with in-time splitting. It is very similar to the random splitting one. Indeed, the top features (*std_B*, *kurtosis_B*, and *peak_to_peak_b*) consistently appear among the top features for both splitting methods.

Both plots show that features from monitor B (link 1) are crucial for predicting failures in all classes (links 1, 2, and 3). However, in the random splitting case, the impact of the top features is slightly less pronounced compared to the in-time splitting plot.
Figure 5.11a, 5.11b and 5.11c represent the SHAP summary plot for each link. Also, in this case, it is very similar to the random splitting one. We can see that the plots explaining the second and third link failure predictions behave very similarly, while the first plot has the opposite behavior.

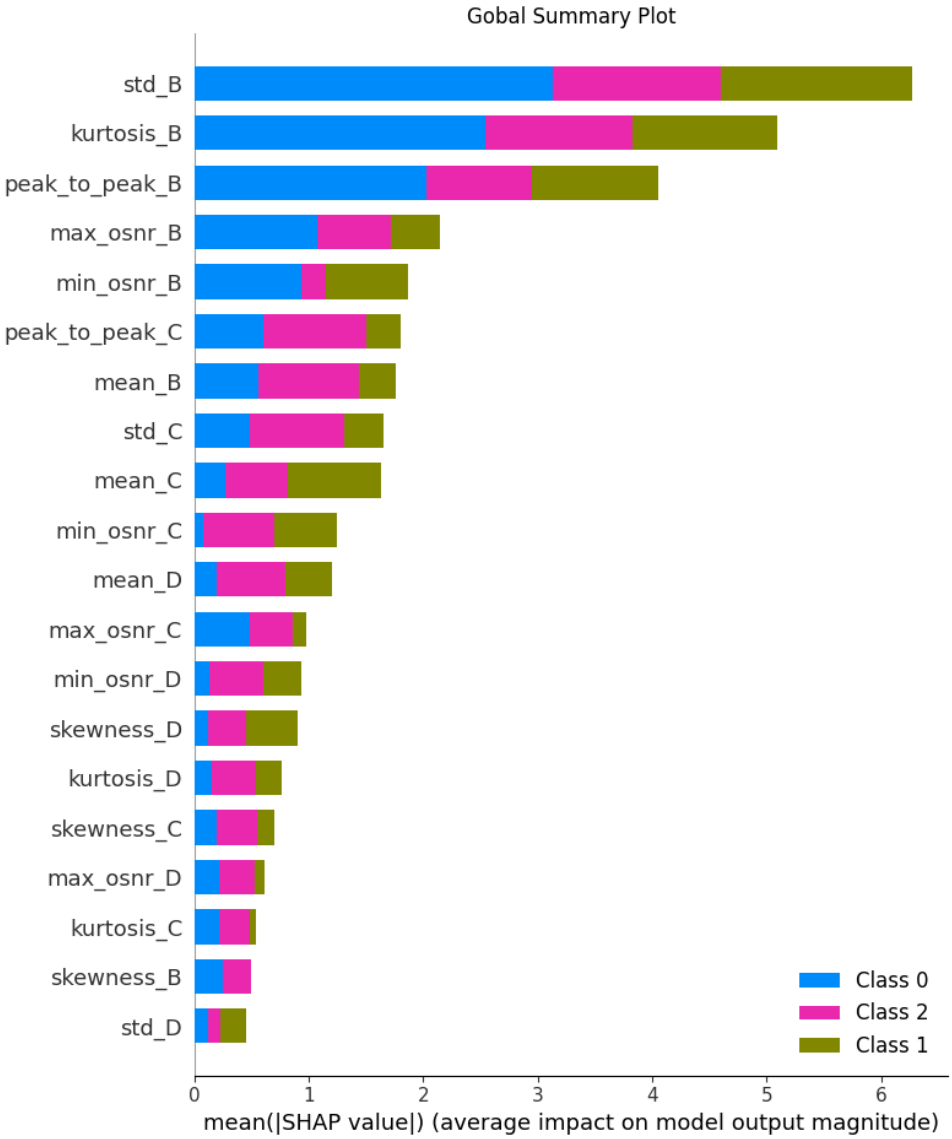
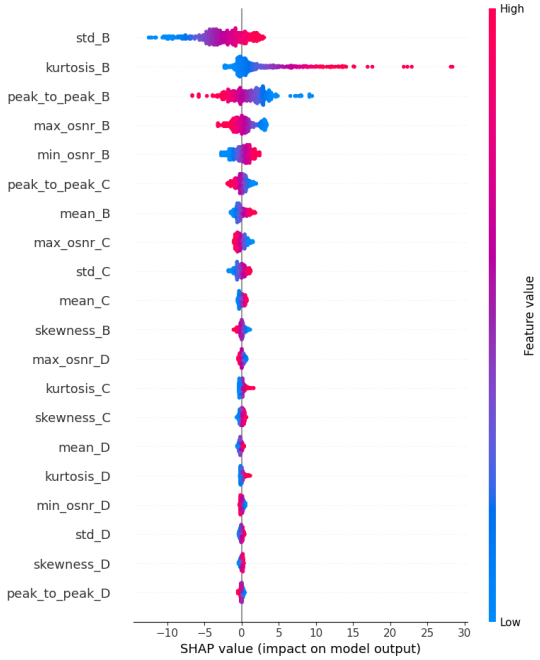
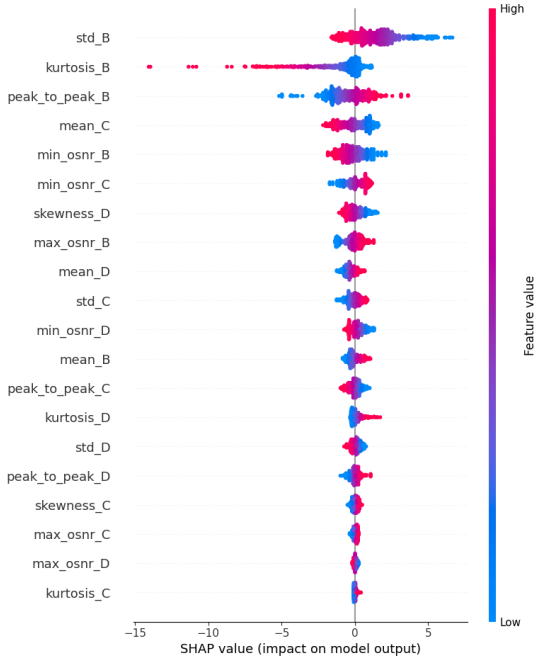


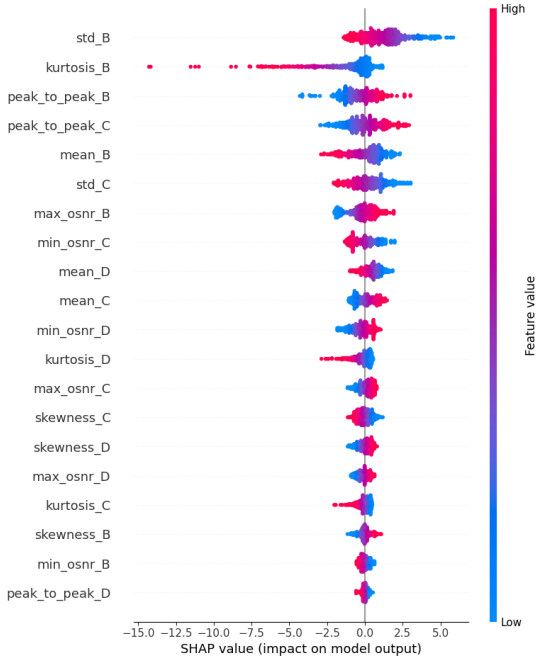
Figure 5.10: Global SHAP Summary plot



(a) Link 1



(b) Link 2



(c) Link 3

Figure 5.11: SHAP Summary plot

5.2.2 XGBoost

The XGBoost model confirms its superiority against the others when applied in the context of failure localization. Again, we have the highest accuracy, but it is lower than in the case of random splitting. As we can see from the confusion matrix in Figure 5.12, it correctly predicts most of the failures at the first link, while it creates some confusion in predicting failures on the other two links.

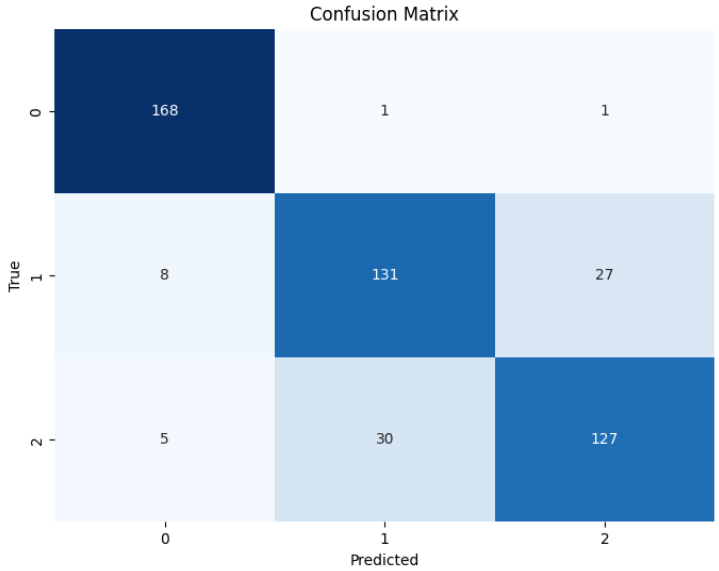


Figure 5.12: Confusion Matrix

The SHAP global summary plot explaining XGBoost with in-time splitting (5.13) is similar to the plot explaining the same model but with random splitting. What changes is not always the overall contribution of a feature in predicting classes, but there are differences in the weight it has. For example, the *peak_to_peak_B* feature is again the most relevant feature to predict a failure in the first link, but its dominance is slightly less pronounced compared to the random splitting case. And again, while in the random splitting, the *skewness_D* feature was a little relevant in predicting failures in the first link, we can see that in the case of in-time splitting, it has no contribution.
Figures 5.14a, 5.14b, and 5.14c represent the SHAP summary plot explaining features' contribution to failures' prediction in each link. While the plots in Figures 5.14a and 5.14b are similar to the random splitting ones, the plot in Figure 5.14c presents some discrepancies. Before, low values of *mean_D* were representative of a fault in the link, while in this case, they indicate that there is not a failure in the third link. The other significative features behave likewise. However, since the model's accuracy is much better when employing random splitting, this diverse behavior tells us where the two models diverge.

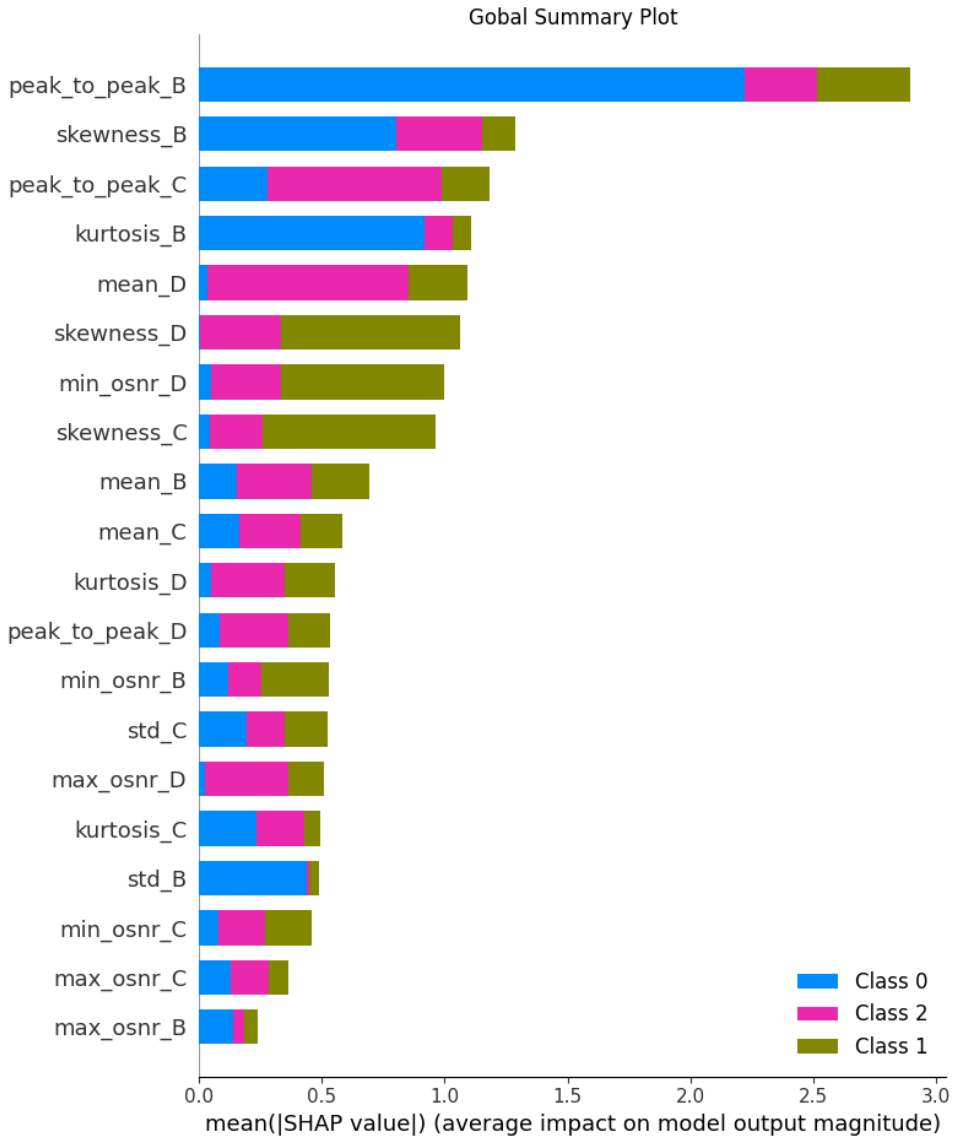
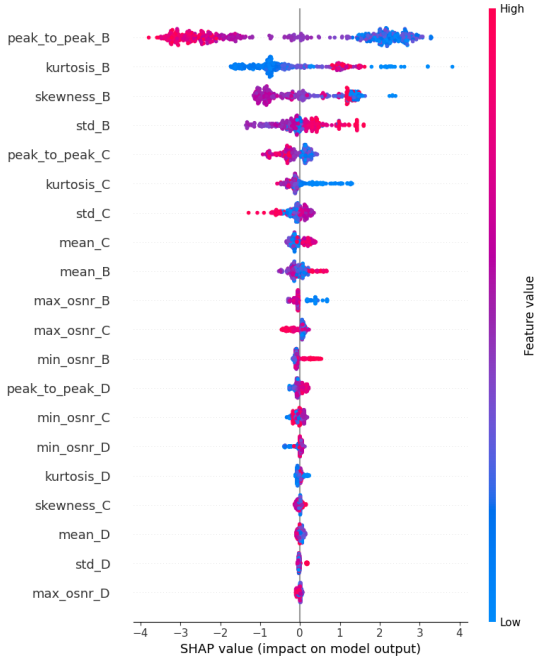
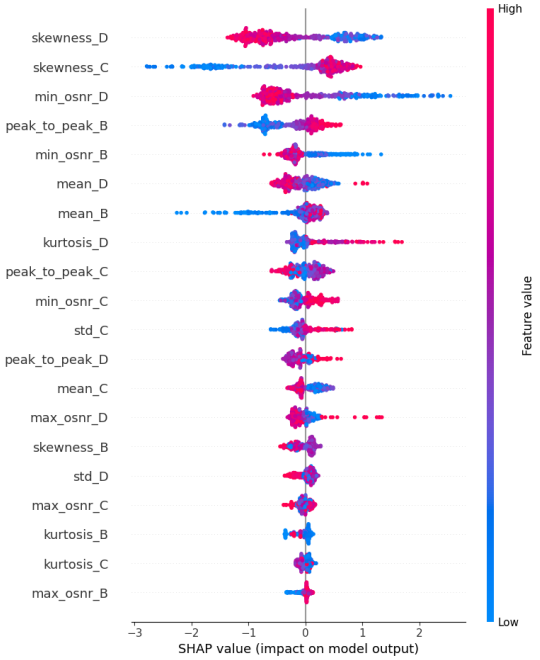


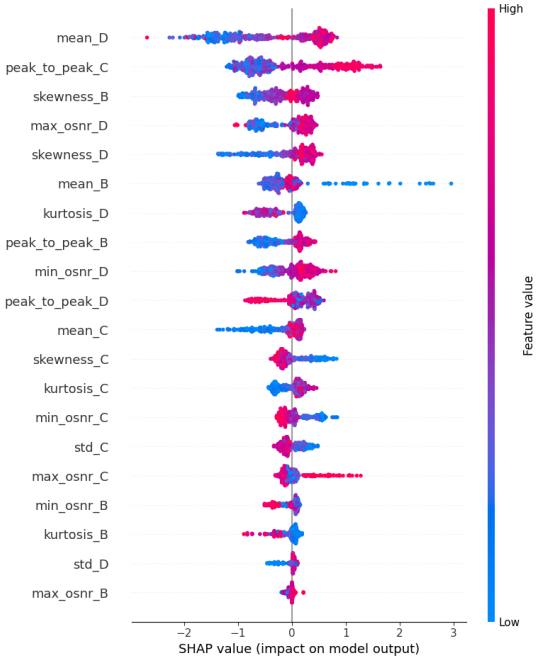
Figure 5.13: Global SHAP Summary plot



(a) Link 1



(b) Link 2



(c) Link 3

Figure 5.14: SHAP Summary plot

5.2.3 LSTM

The in-time splitting seems to work better for the LSTM model. Indeed, LSTM models are designed to capture temporal dependencies in sequential data. However, as we can see from the previous accuracy results of ML models within in-time splitting, the features in our dataset do not have significant time-dependent patterns. Hence, the accuracy of our LSTM model is again low. From the confusion matrix in Figure 5.15, we can see that the model mispredicted a significant number of failures at the third link as if they were in the first.

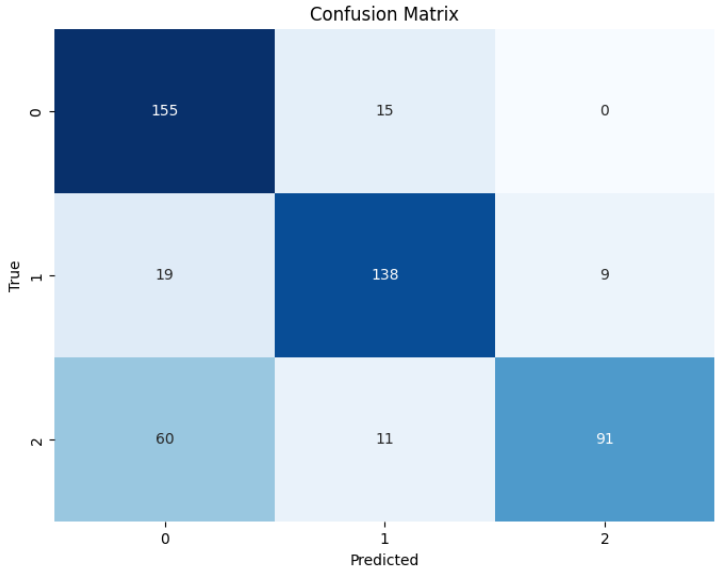


Figure 5.15: Confusion Matrix

6 Conclusions

Our work showcases machine learning and XAI employment in the fault localization problem, analyzing the reasoning of different machine learning models trained on OSNR measurements. In particular, we analyzed how three different machine learning algorithms perform. Moreover, we experimented with two separate train-test splittings (random and in-time) to investigate temporal dependencies in our dataset. In conclusion, we explained our best-performing model's decisions with Explainable AI. In particular, our analysis highlighted that the three-based model XGBoost is the best-performing model for this problem. Moreover, our experiments demonstrated that dataset random splitting works better than in-time splitting: this is true for Logistic Regression and XGBoost only. These latter models output a nearly perfect performance with random splitting. Contrary to our expectations, the LSTM model did not perform as well as we thought. Since LSTM is designed to capture temporal dependencies in sequential data and our best-performing model works better with random splitting, we can conclude that our dataset lacks of time-dependent patterns. Furthermore, we explained Logistic Regression and XGBoost models with Explainable AI. XAI is paramount to increase the sense of trust towards the model, encouraging the adoption of machine learning tools for fault management.