PRESENTATION

Both the datasets that were given in this project were characterized by rows each one representing a path in which there were the lengths in km of the fiber in the spans, then a block containing the number of channels per each link and finally a block containing the SNR of the link. In the data preprocessing we calculated the mean, the variance and the standard deviation of the elements of these three blocks, then we calculated some features for the paths as the number of fiber spans, the overall length, the length of the longest span, the number of links, the total number of channels of all the links and the maximum number of channels on a link; subsequently we put all these features inside a matrix that has been used for further analysis. We have plotted the relationships between each one of these features and the SNR, finding that the number of fiber spans and the light path length are inversely proportional to SNR and at the same time the other features are not strongly related with it.For the regression and and the classification part we used LGBM; LightGBM constructs decision trees differently from other boosting algorithms. It uses a technique called **Leaf-wise Growth** instead of the traditional **Level-wise Growth**. In other words, it expands the leaf node that provides the greatest gain, rather than expanding all nodes at the same depth. This approach allows for deeper trees and more complex models. Like other gradient boosting algorithms, LightGBM iteratively minimizes a loss function. At each iteration, it builds a new tree that tries to correct the errors made by the previous trees. This is done by using the gradients of the loss function to update the tree weights.

LightGBM has native support for categorical features. It uses a method called **Gradient-based One-Side Sampling (GOSS)**, which intelligently selects a subset of training data to reduce computational costs without significantly compromising model quality.

First of all for the regression part we have splitted the dataset in train and test then we used quantile regression (that is a kind of probabilistic regression in which once you define the quantile you’ll obtain a curve under which you have the quantile percentage of the elements) just setting the parameter objective to ‘quantile’. We performed two kinds of analysis: with and without optimized hyperparameters. In the first case we used several hyperparameters:

* n\_estimators: number of boosting rounds (trees) to be used in the model, the more they are the higher is the computation time and the performances
* max\_depth: the maximum depth of a tree
* learning\_rate: it scales the contribution of each tree added to the model. By using a lower learning rate, the algorithm takes smaller steps towards the optimal solution, which helps in smoothing out the learning process and avoiding overfitting.
* Num\_leaves: maximum number of end nodes in each tree
* Min\_child samples: minimum number of samples to form a leaf
* Subsample: fraction of datas to be used for fitting individual trees
* Colsample\_bytree: fraction of features to be used for fitting individual trees
* Reg\_alpha and reg\_lambda: represent the weights to be assigned for penalties

and we found the best ones using randomsearch crossvalidation, where are taken some random combinations of the parameters and is performed the crossvalidation using 5 folds. In the second case we just used some default hyperparameters. Finally we compared the results and we found that with the optimization of the parameters the number of incorrected MFs is 22 and without it is 26.

In addition to this we tried to use also linear regression and the KNN as classifiers, in particular for KNN we used the best number of neighbours found using gridsearch crossvalidation that tries all the possible combinations of hyperparameters and does the crossvalidation over 5 folds taking the mean value of the mean squared error. As expected the results of the two simpler models are worse than the ones obtained using lgbm.

Then we also tried to use the European model to train the model and the german dataset to test it, in this case the results that we obtained are quite good in terms of accuracy, precision, recall and wrongly assigned MFs; but at the same time doing the operation swapping the two datasets is not so good, this could be related to the fact that the german daset is smaller than the European one, then we could also say that the European network has longer links and great variance, so training with the European model should be better, at the same time the german dataset has smaller links and a lower variance, so if we use the german dataset for training the model it could be difficult to recognize longer links.

At the end we tried also to extract some informations using explainable AI and with this plot, we can see that for the European dataset, the value of the predicted SNR depends very highly on the feature "Total Len", something expected for a lightpath.

Also, as concluded from the graphs, the second and third more impactful features were "Total Channels" (means some interference on the lighpath takes place) and the "# of Spans" (means the introduction of more Optical Amplifiers that introduce more AES noise that degrades the SNR).

For the german dataset instead, the plot still shows

that the feature "Total Len" is the one that dominates as the most important aspect for the prediction of the SNR, as expected.

Futhermore, the "# of Links" feature gets more importance for this dataset (it's also a feature corresponding to the possible interference in the lightpath).

The "# of Spans" is still a feature with a considerable influence, reasonable because of the influence of the increased number of Optical Amplifier that implies the higher number of spans.

In the second part of the code we used LGBM directly as a classifier so the final output in this case is a categorical value and we can directly associate each path to a specific modulation. In this section we did exactly the same kind of operations that we did for the first part, so we analysed the performances of LGBM with/without hyperparameters, KNN with the optimization of the number of neighbours and also a linear regressor; the final plot shows us that also in the case the LGBM with the best hyperparameters gives the best results.

Also in this case we tried to extract some additional informations using explainable AI and we obtained for the European dataset, that using these plots for each MF, we can see the impact of the features in the prediction of each class. As expected, the "Total Len" is the most influential feature on every MF.

As commented before, the "Max Span Len" feature seems to be important only for one MF (16QAM), and not in a really significant manner; so maybe this is one feature that could be neglected in following trials of the algorithm.

In addition, for the highest of MF (64QAM) it can be appreciated that more features, like "Total Channels" and "Max Channels in a Link", take a more considerable effect on the predction. This is due to the fact that, gaining high values of SNR (that allows the use of this MF) seem to take great impact of the interference factor on the lightpath, which these features seem to describe.

For the german dataset instead, the "Total Len" is as expected the feature with the most impact on the model (by quite a lot). Obviously, as longer the lightpath, the less the SNR will be.

The second and third most impactful are the "# of Links" (more chance of interference) and "# of Spans" (More amplifiers used, so more AES error), which was also concluded from the graphs of the features on the Data preprocessing section of the notebook.

The lest influential feature is the one of "Max Span Len", as in can occur that there is a lenghty span but the lightpath has only 1 span, this wouldn't be of high impact.

On the last plot can be seen that on the first two types of MFs (8QAM and 16QAM) we found that the the "Total Len" is indeed the most important feature to determine if they are the one used.

For 32QAM, "# of Links" as well as "Total Len" are by far the two most important deciding features for this type of MF.

However, for 64QAM the feature of "Max Channels in a Link" takes the place as the most impactful feature. So, the interference provoqued by the links represent the deciding factor of choosing this MF. On the other MFs, this feature was the less important.

Accuracy: (total number of correctly classified example)/(total number of examples)

Precision: (true positives)/(true positive + false positive : number of actual positive)

Recall : (true positive)/(number of actual positive: true positive +false negative)

F1 score : 2[(precision X recall)/(precision + recall)]

Why to use decision three algorithms instead of neural networks?

Decision trees provide clear, visual representations of decision rules.

Generally faster to train compared to neural networks, especially for small to medium-sized datasets.

Require less computational power and can be trained on standard hardware

Perform well on small datasets without significant overfitting