Failure-cause identification with different failures location

Network Data Analysis Lab Project

Authors

Luca Marsella

Pasquale Romano

POLIMI - Politecnico di Milano

Academic year 2021/2022

Analysis

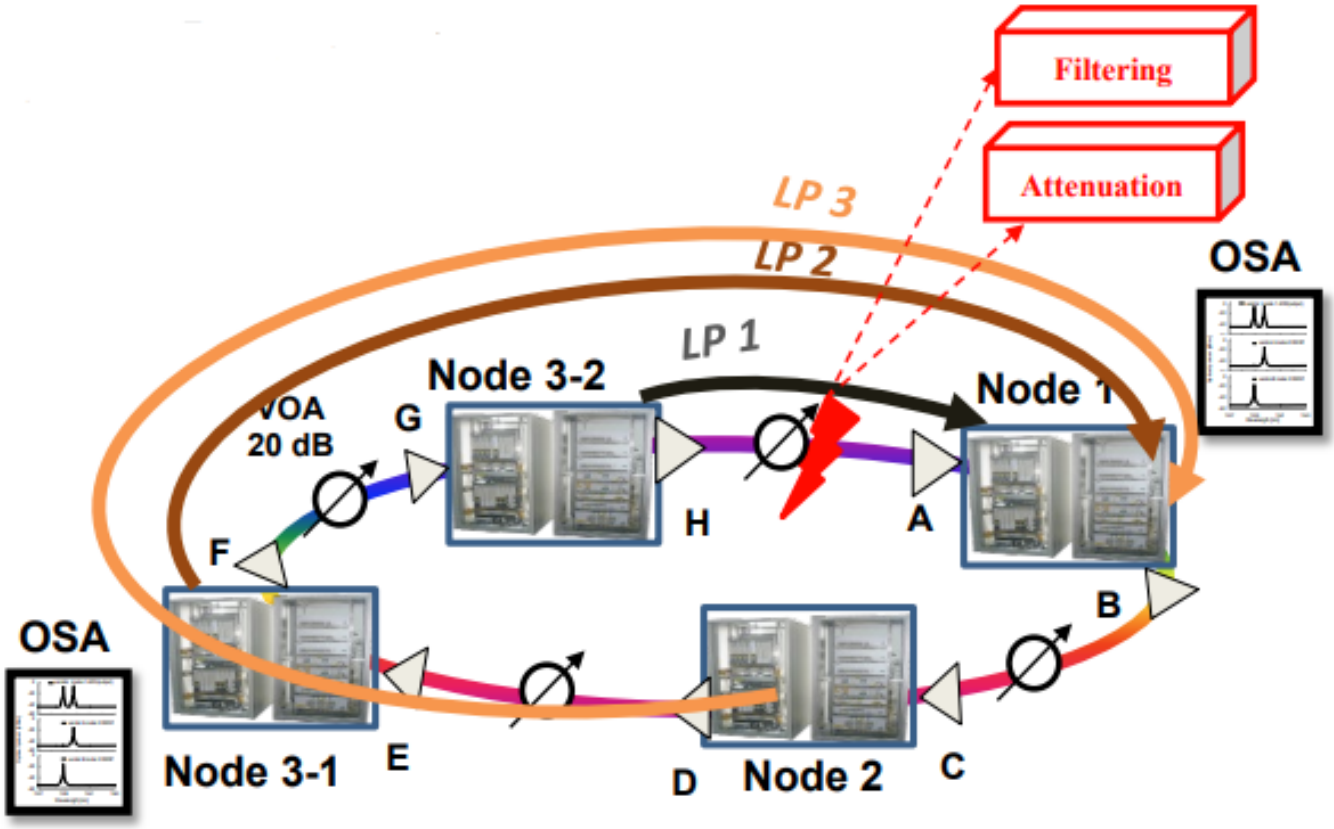
Project

The aim of this project is to give different lightpaths scenarios (different lightpath lengths, with different types of failures at different locations) to learn models that, given an "OSNR-window" observed at the receiver, predict what is the failure cause on the lightpath.

We can have two types of failure:

- Attenuation (dB)
- Filtering (GHz)

Testbed



Dataset

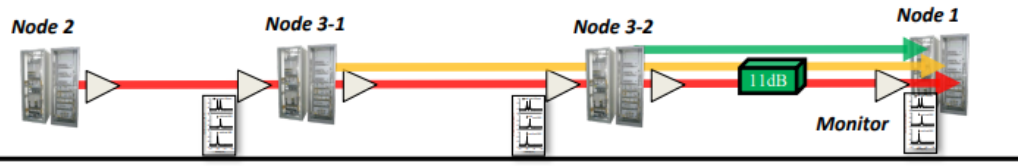
The scheme below represents different scenarios in which two types of errors (attenuation or/and filtering) may occur.

The dataset used for the analysis is represented in the below table:

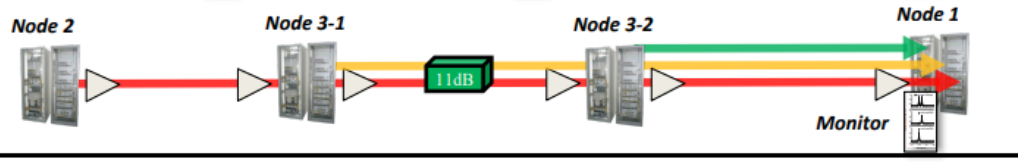
	Description	Magnitude	Location 1	Location 2
Scenario A ← Scenario 0	No failures	-	-	-
Scenario B ← Scenario 1	Excessive attenuation	11 dB	3-2 – 1	-
Scenario 2	Excessive attenuation	11 dB	3-1 – 3-2	-
Scenario 3	Excessive attenuation	11 dB	2 – 3-1	-
Scenario 4	Excessive attenuation	11 dB	3-1 – 3-2	3-2 – 1
Scenario 5	Excessive attenuation	11 dB	3-2 – 1	2 – 3-1
Scenario 6	Extra filtering	12.5, 25, 100 GHz	3-2 – 1	-
Scenario C ← Scenario 7	Extra filtering	12.5 GHz	3-2 – 1	-
Scenario 8	Extra filtering	50 GHz	3-2 – 1	-

Scenarios

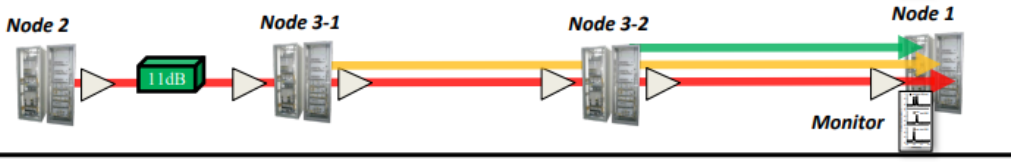
Scenario 1		
2-1	3-1-1	3-2-1
21600	21600	21600



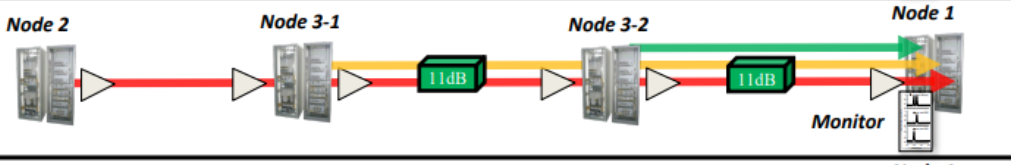
Scenario 2		
2-1	3-1-1	3-2-1
21600	21600	21600



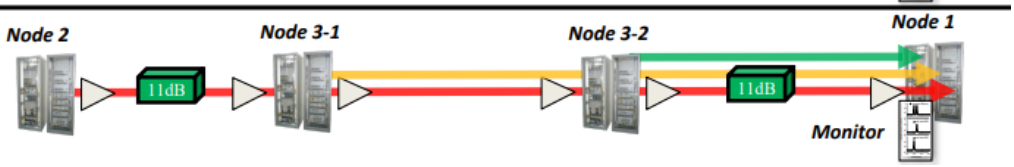
Scenario 3		
2-1	3-1-1	3-2-1
21600	21600	21600



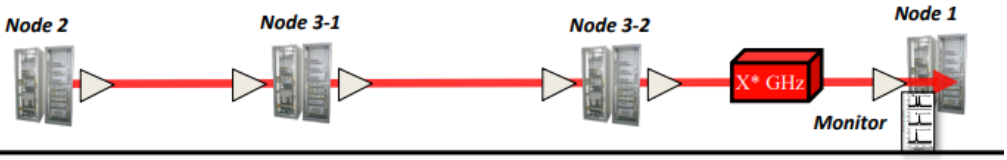
Scenario 4		
2-1	3-1-1	3-2-1
21600	21600	21600



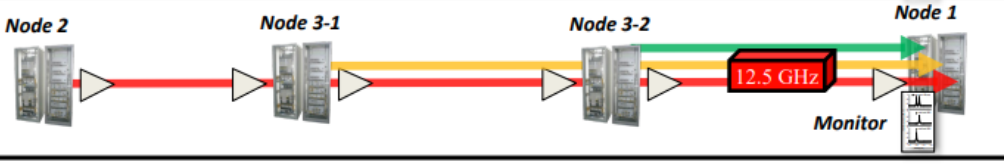
Scenario 5		
2-1	3-1-1	3-2-1
21600	21600	21600



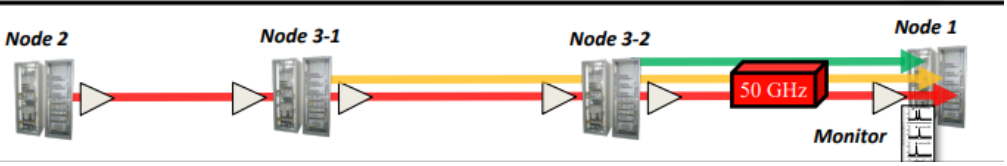
Scenario 6			
12.5	25	50	100
300	300	0	600



Scenario 7		
2-1	3-1-1	3-2-1
21600	21600	21600



Scenario 8		
2-1	3-1-1	3-2-1
21600	21600	21600



Approach

Features generation

In order to perform analysis of the data extracted from the scenarios above, 6 features have been generated from the last column of the dataset, the OSNR.

They are:

- Mean: mean value of the window;
- Max: maximum value of the window;
- Min: minimum value of the window;
- RMS: root mean square error of the window;
- Std: standard deviation of the window;
- Ptp: peak to peak.

These features have later been used to train the models.

Hyperparameters tuning

The models used to perform a machine learning-based approach to classification (XGB, MLP, KNN) all need some input hyperparameters. These values need to be computed and optimized. This operation is only performed once and the parameters obtained from the first optimization are used for the training of further analysis.

In this phase, we use the **K-fold Cross-Validation** with K equal to 5. K refers to the number of groups that a given data sample is to be split into.

General info

Training time test info

One of the performance metrics that has been considered is training time. To have an accurate value and to feed the candlestick representation, every training was executed three times.

Results

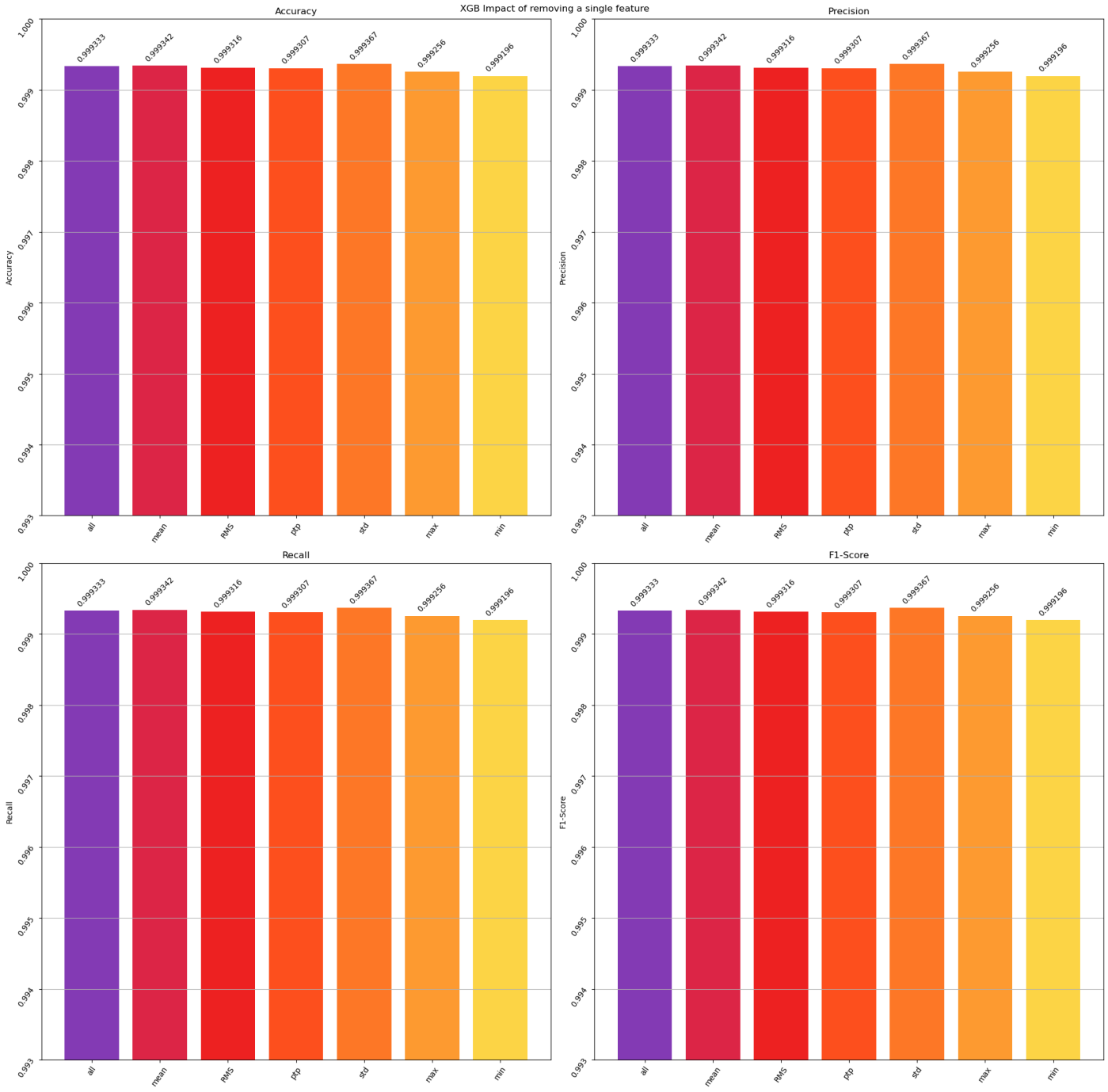
1.1 - What is the impact of including/excluding different features on the performance?

Naive Approach

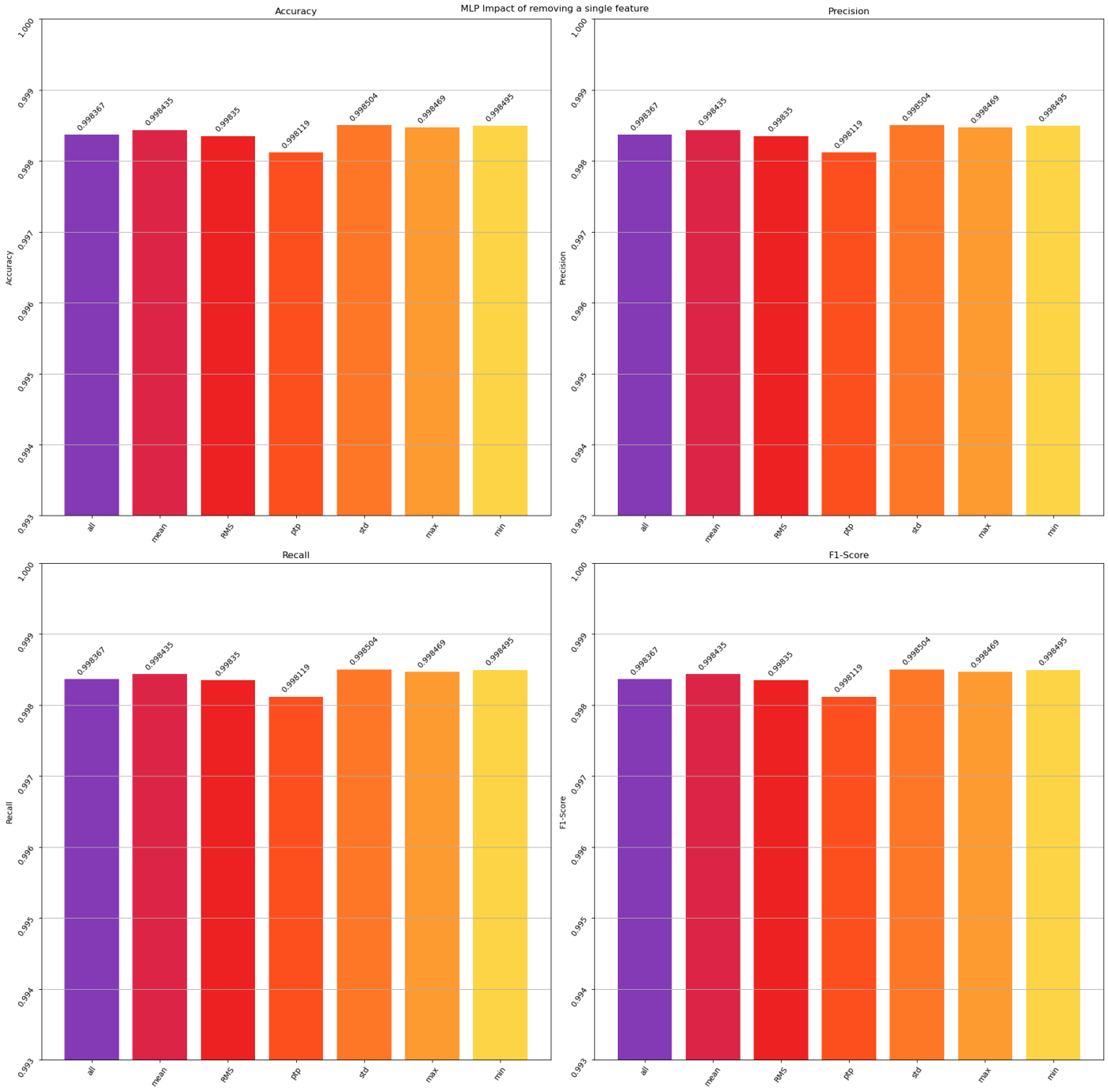
The first thing that we did to measure the impact of including/excluding different features was trying to remove a single feature and perform the training again.

In the below graphs, we listed a comparison of the Accuracy, Precision, Recall, and F-1 score with a single missing feature and with all the features.

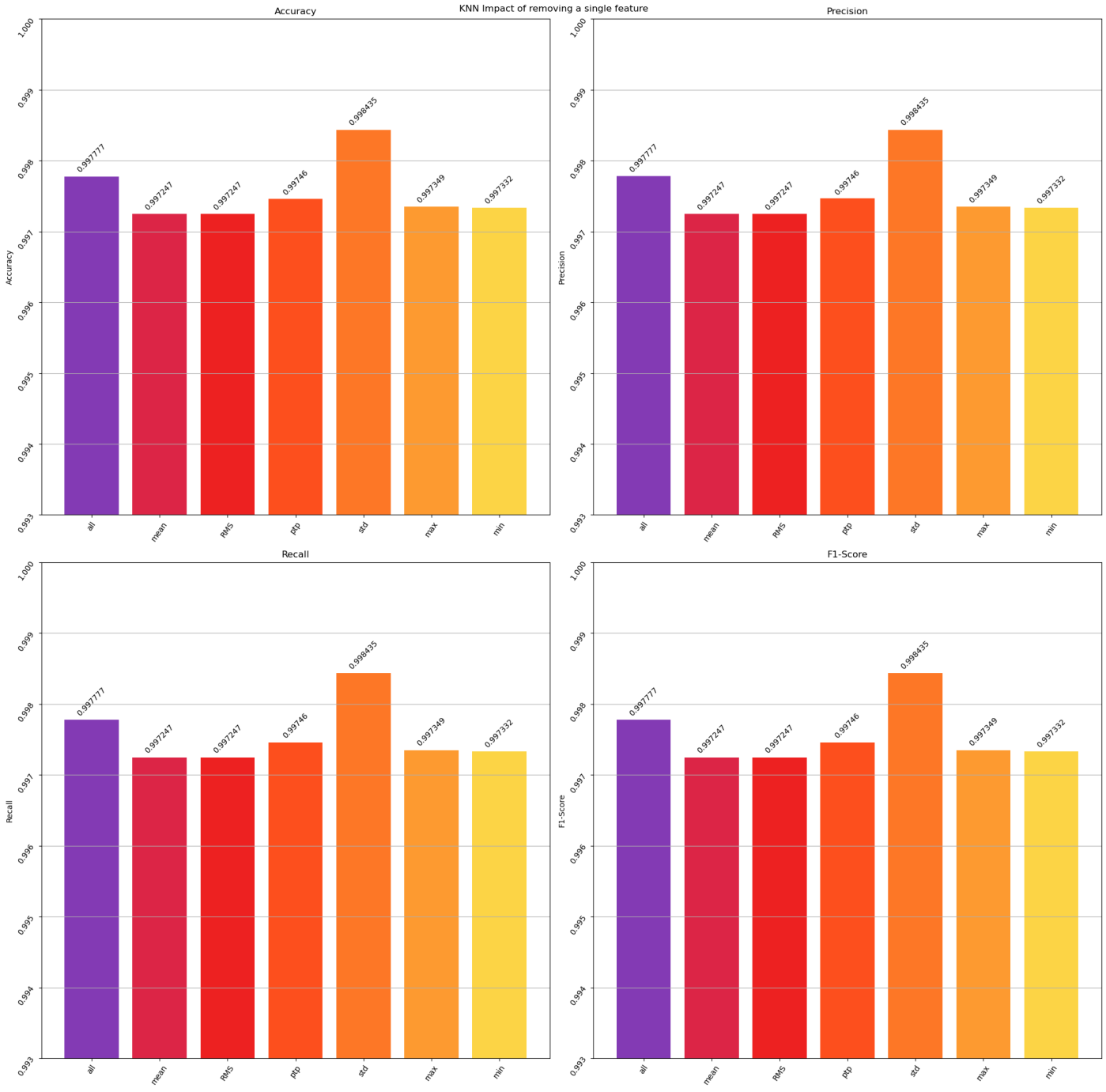
XGB Accuracy, Precision, Recall, and F-1 score



MLP Accuracy, Precision, Recall, and F-1 score



KNN Accuracy, Precision, Recall, and F-1 score



Advanced Approach

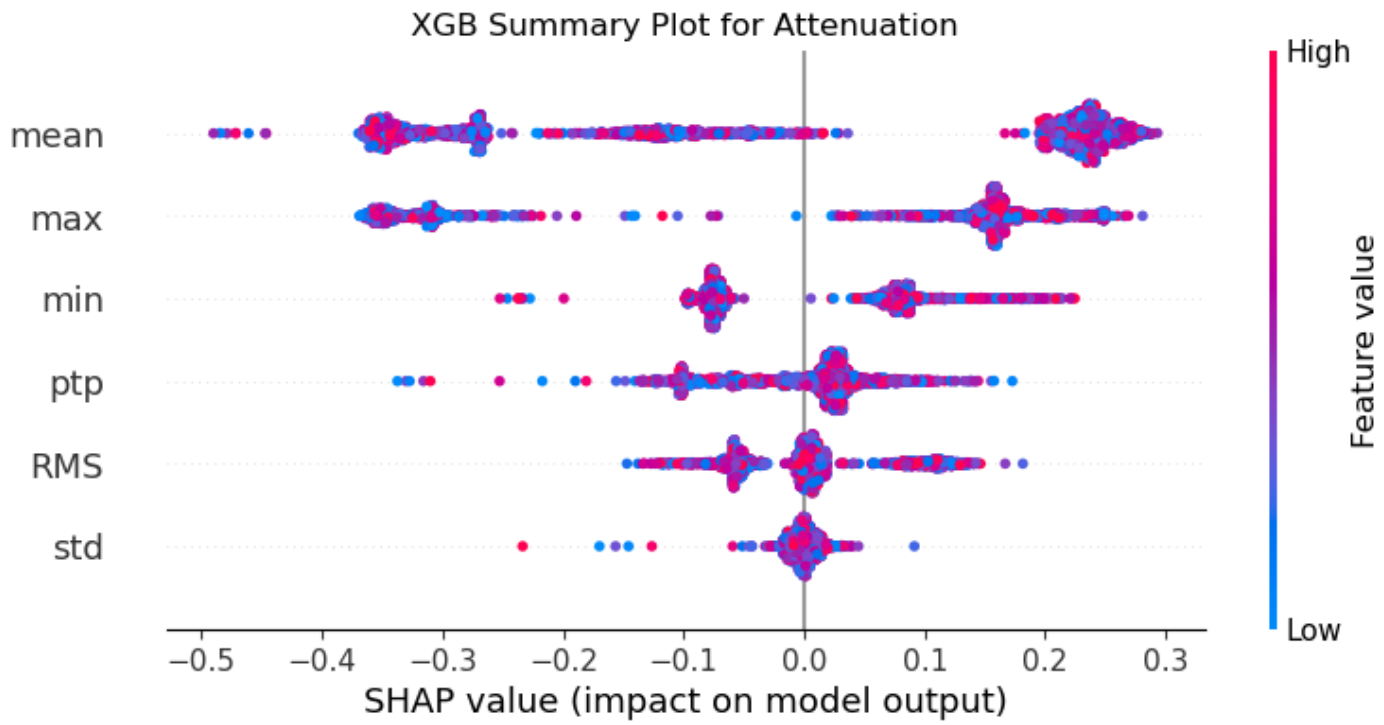
To check the impact of including or excluding different features on the model performances the **shap** and **lime** libraries have been exploited.

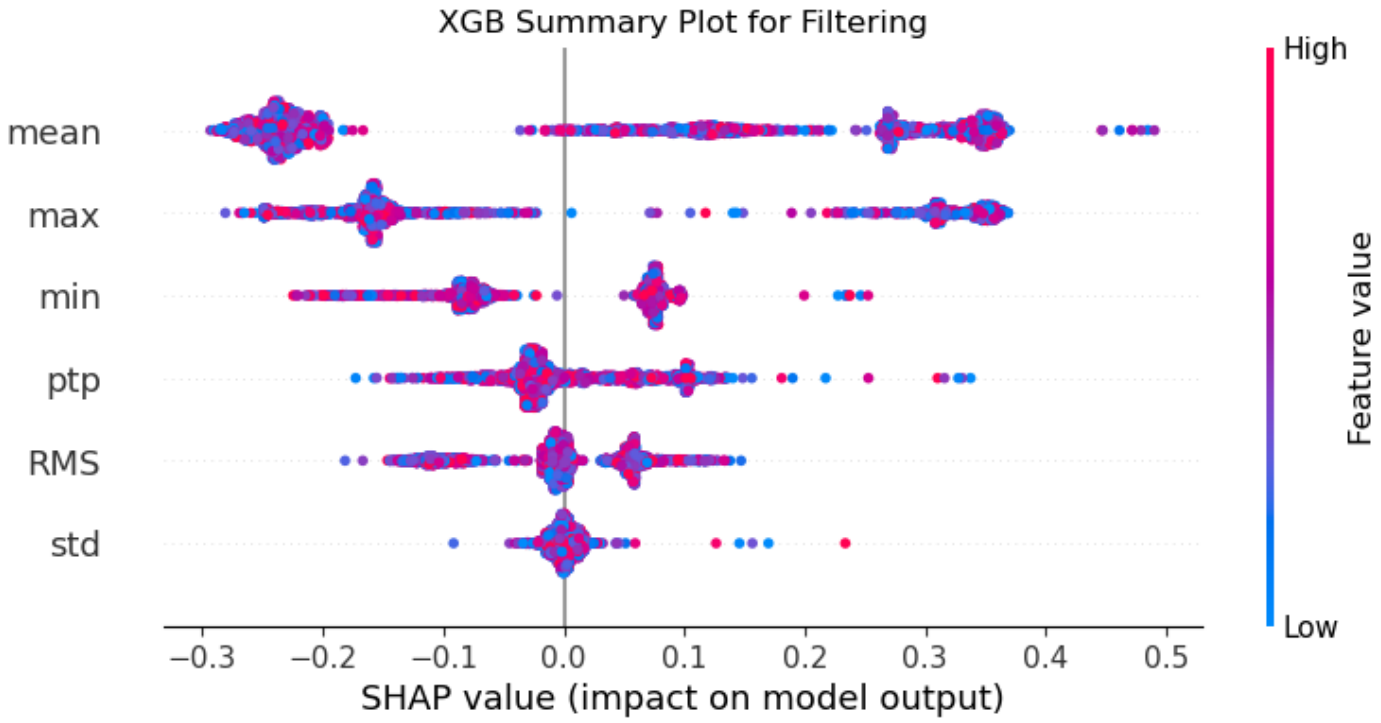
SHAP Plot

The plots below show how the Shapley values have impacted the decisions. They were plotted for both the XGB, KNN, and MLP models and for the two faulty cases.

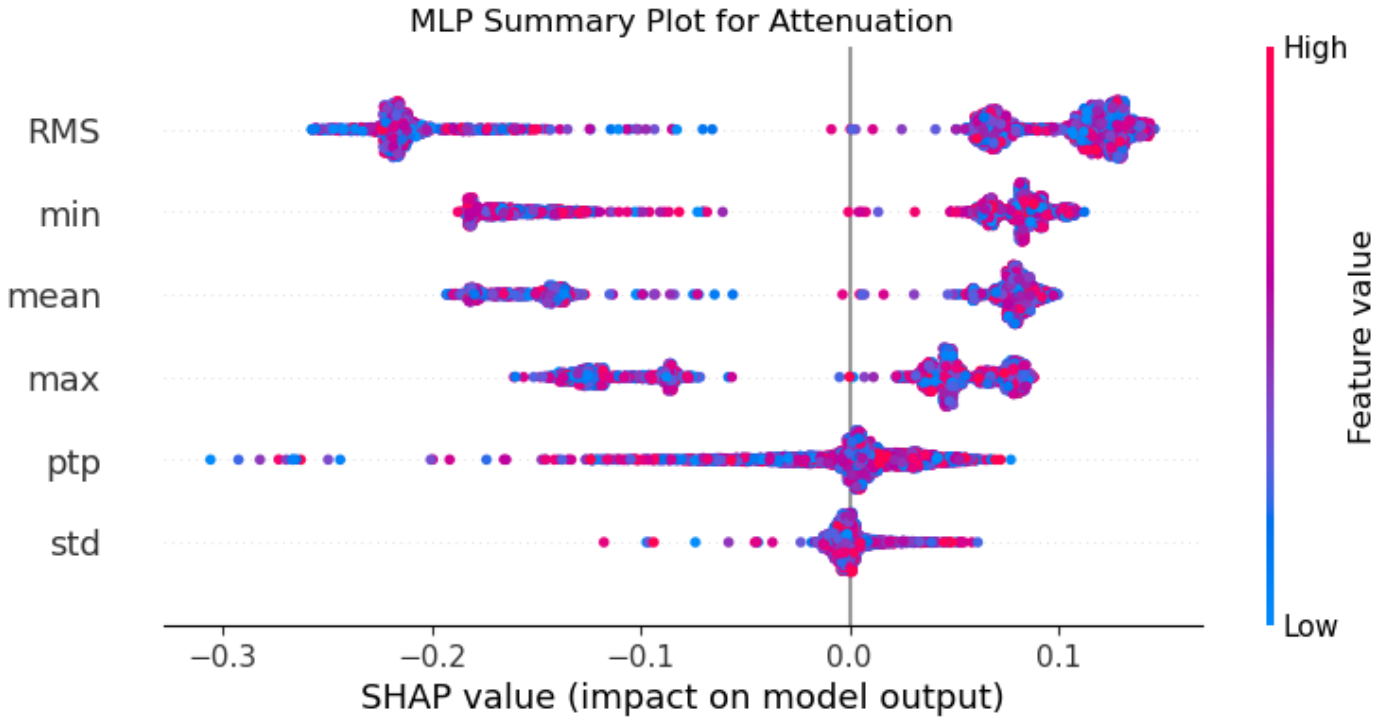
This kind of plot is meant to represent in descending order how much the features have impacted on the decisions. The x-axis represents these Shapley values, positive if positive influence, negative if negative. Each feature data point has also a "heat" value. It has a different colour according to its value in the dataset.

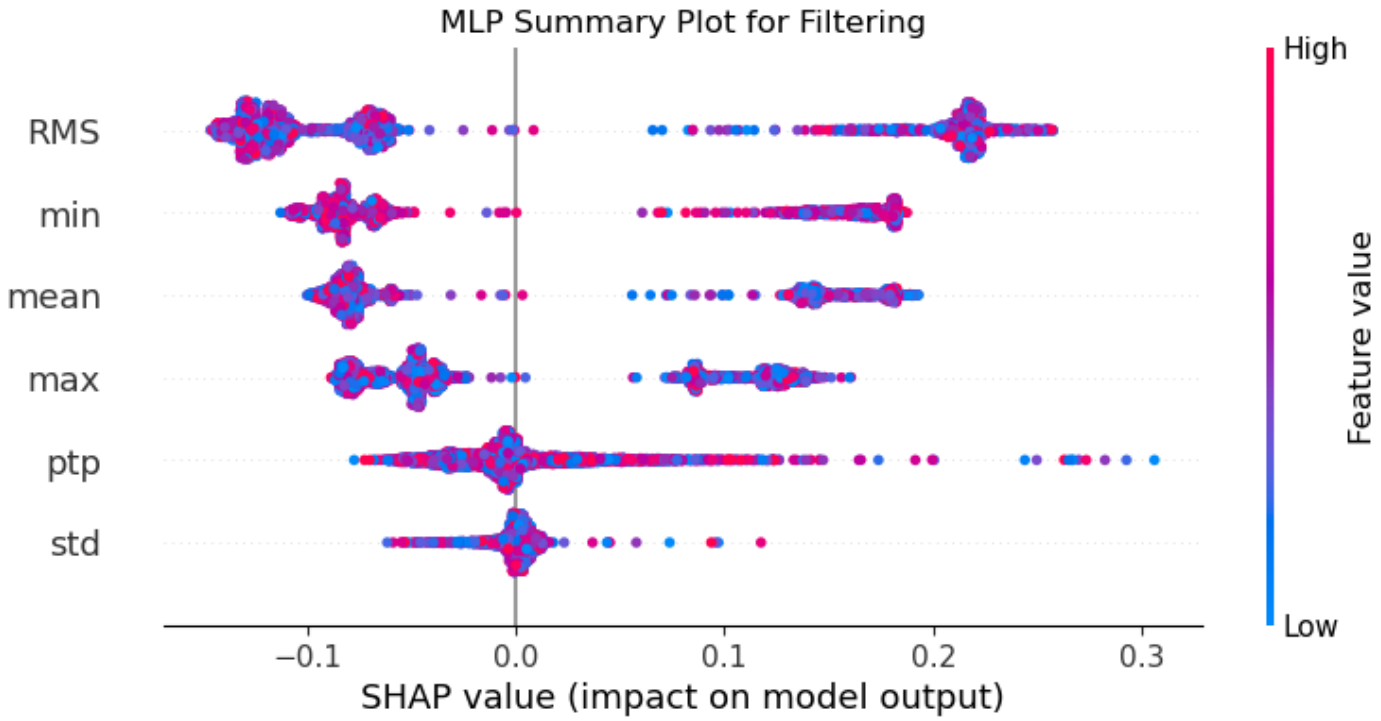
XGB SHAP plot



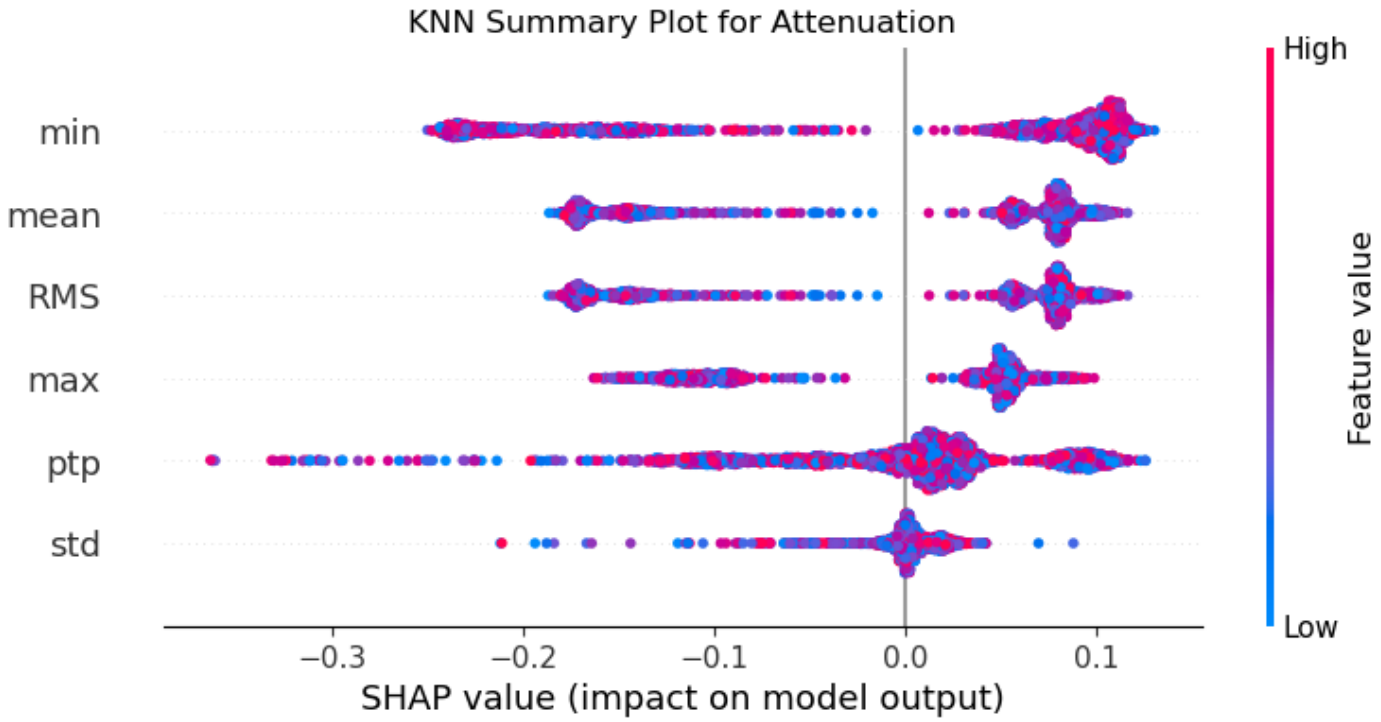

XGB Summary Plot for Filtering
mean
max
min
ptp
RMS
std
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
SHAP value (impact on model output)
High
Feature value
Low

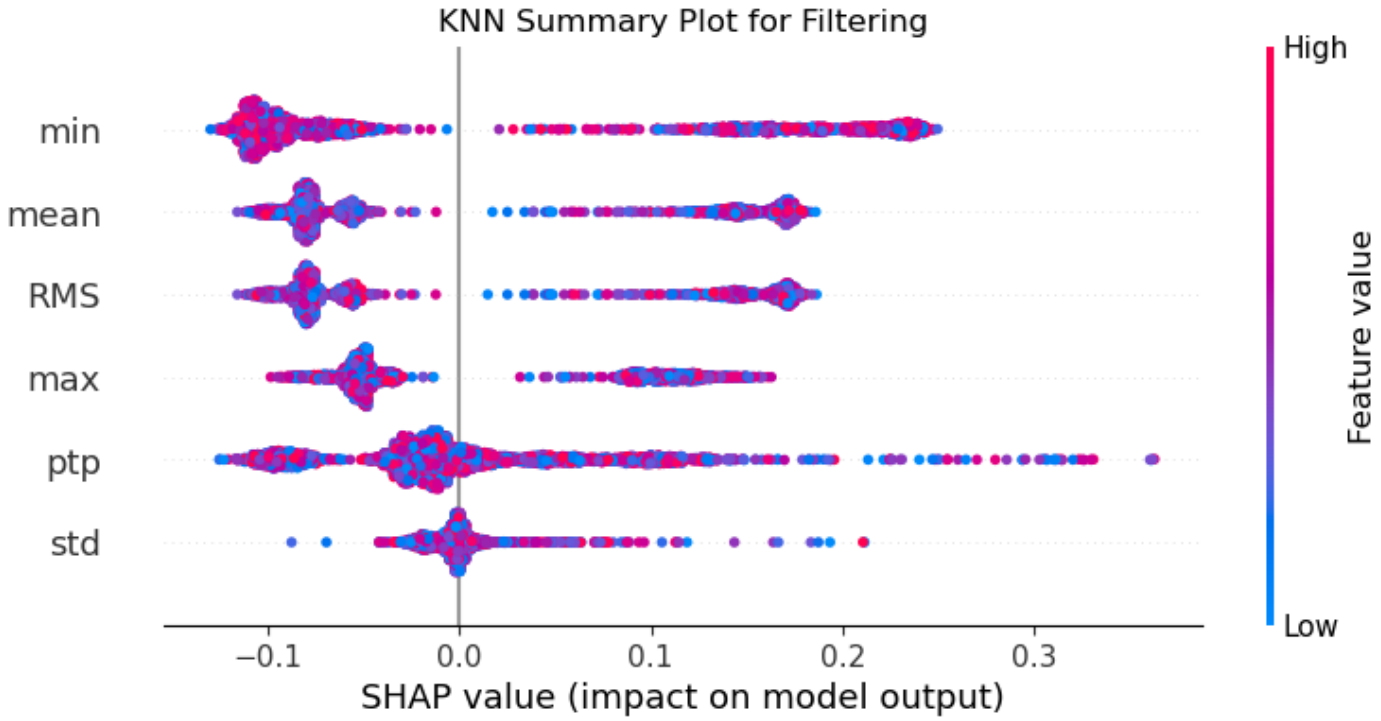
MLP SHAP plot





KNN SHAP plot

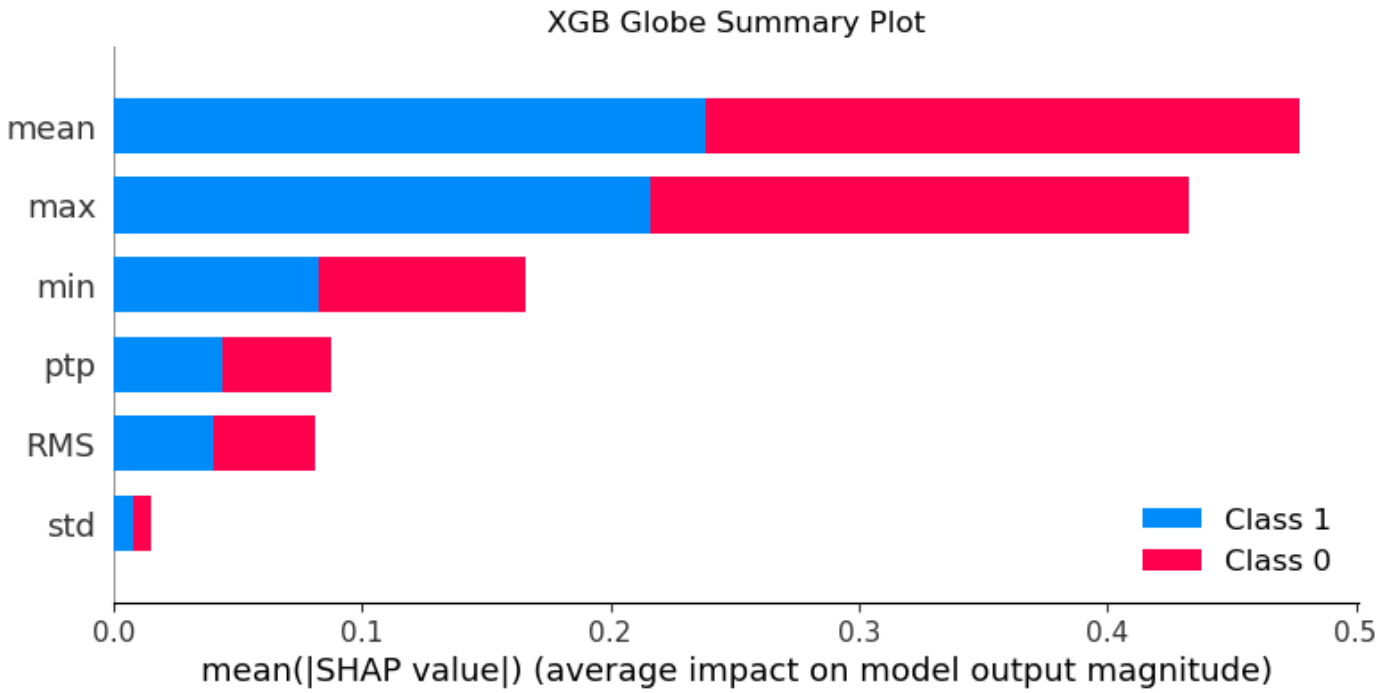




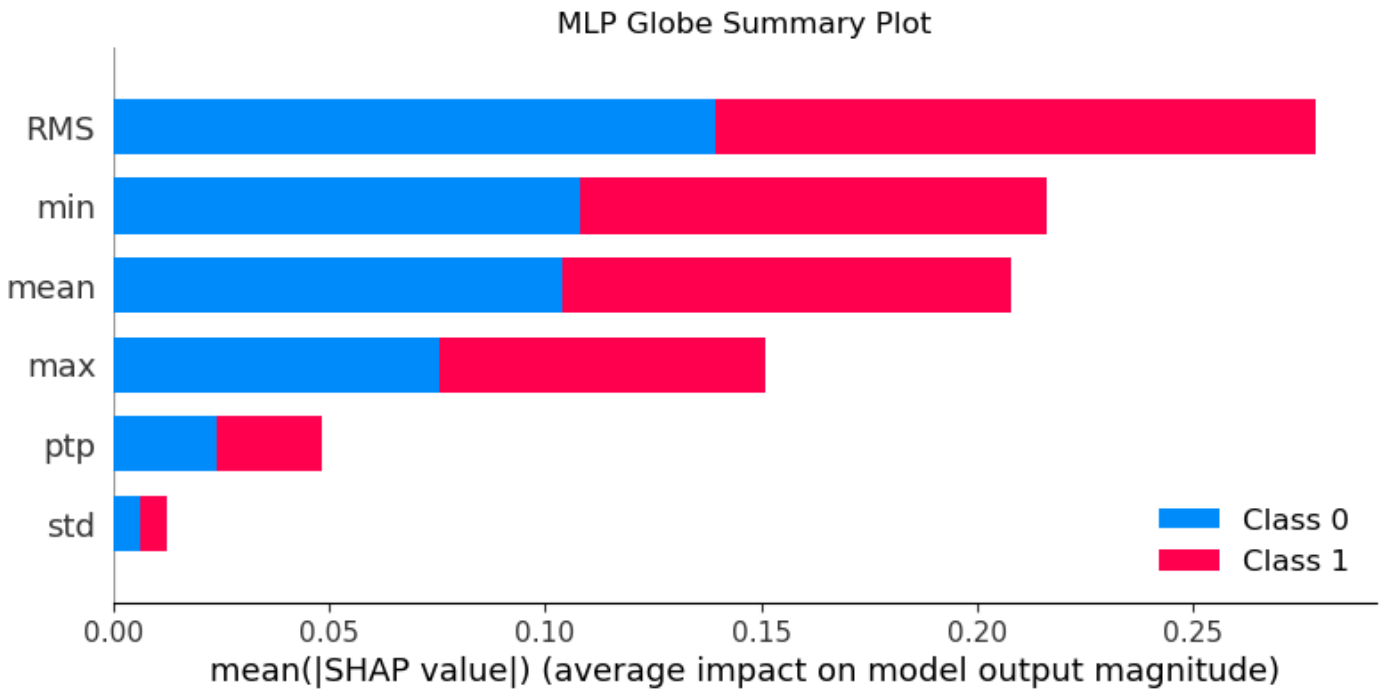
The neat separation between the positive and negative values, x-axis wise, of the first four features, which are also the most impactful, justifies the high accuracy the models have, even in the extreme cases of using such a low percentage of the dataset of the training set as 1%, introducing missing values for a reasonably high percentage of the dataset or removing a feature at all.

XGB SHAP Summary Plot

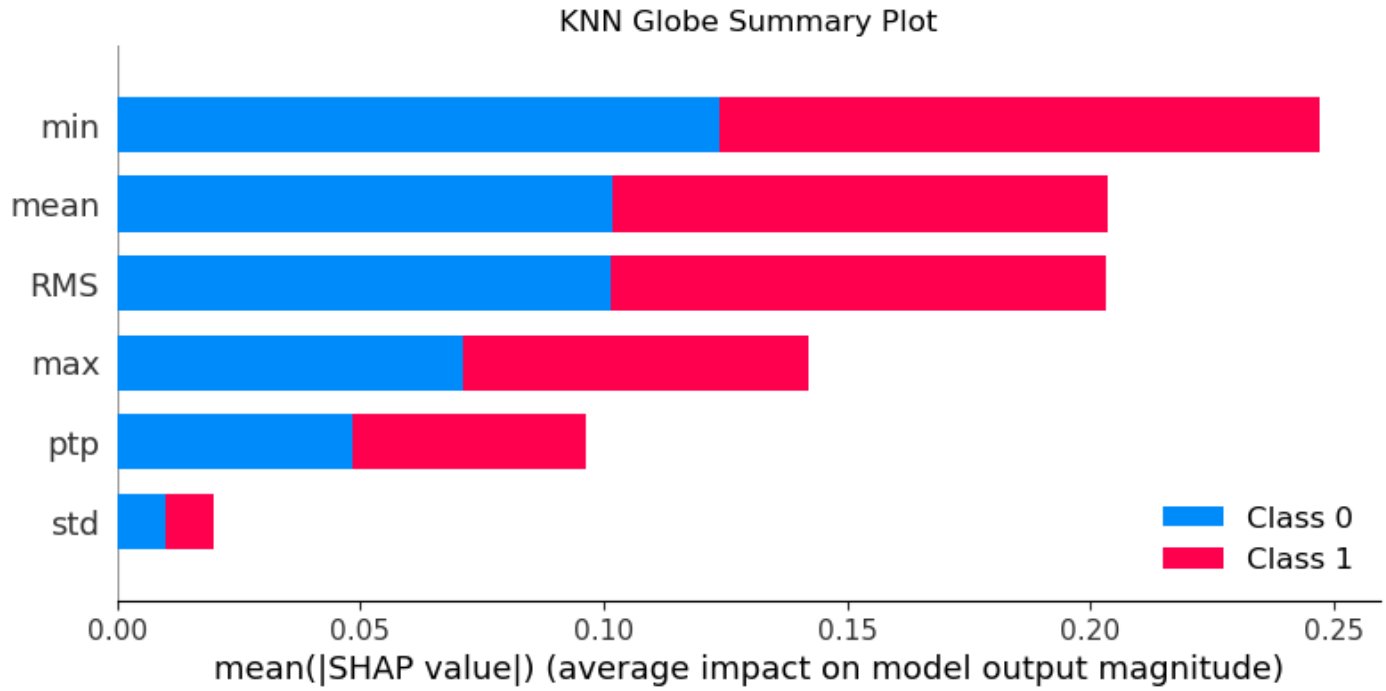
The plots below show how much the different features impact the decision.



MLP SHAP Summary Plot



KNN SHAP Summary Plot



LIME Plot

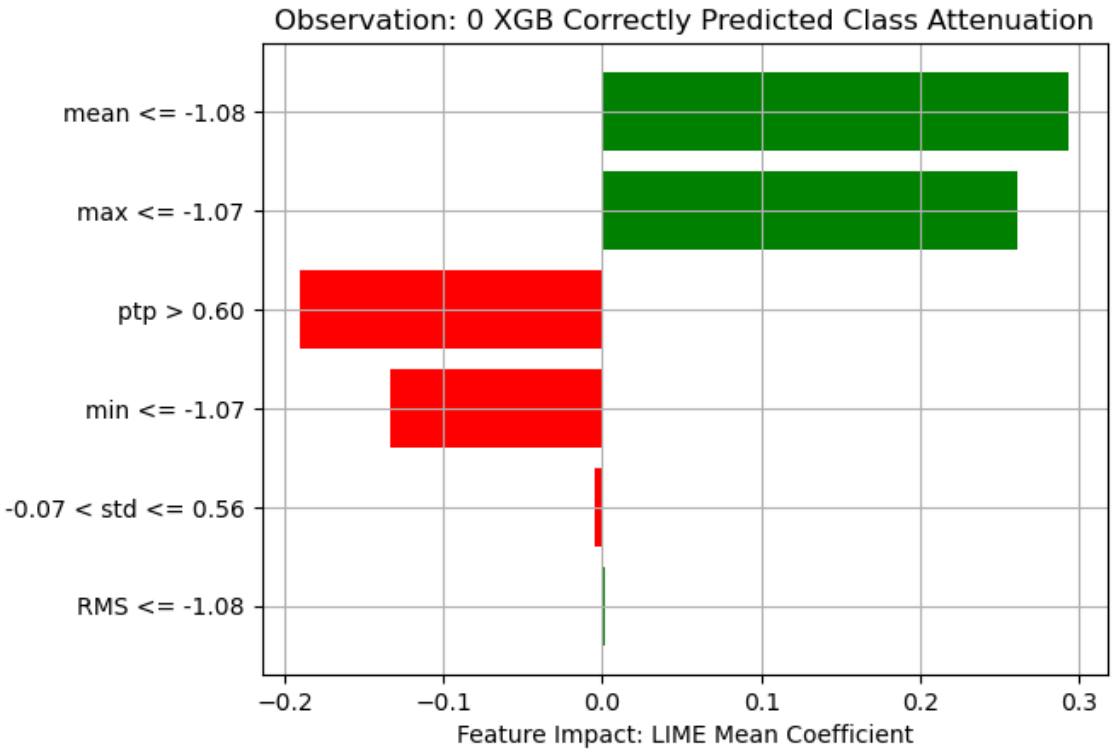
LIME is a python library that tries to solve for model interpretability by producing locally faithful explanations.

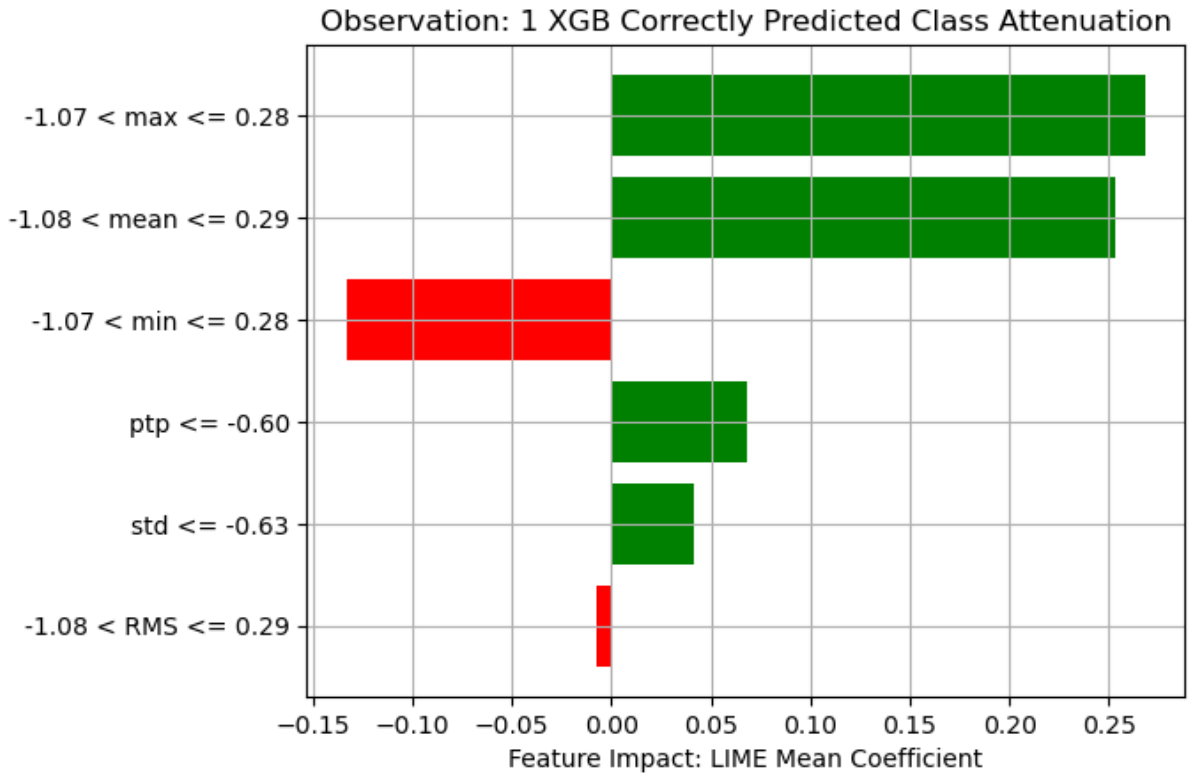
The graphs below show how much the different features had an impact on the decisions.

The green boxes represent the positive impact of features on the decisions, and the red boxes the negative impact. The range at the left of the graph is the domain of the features.

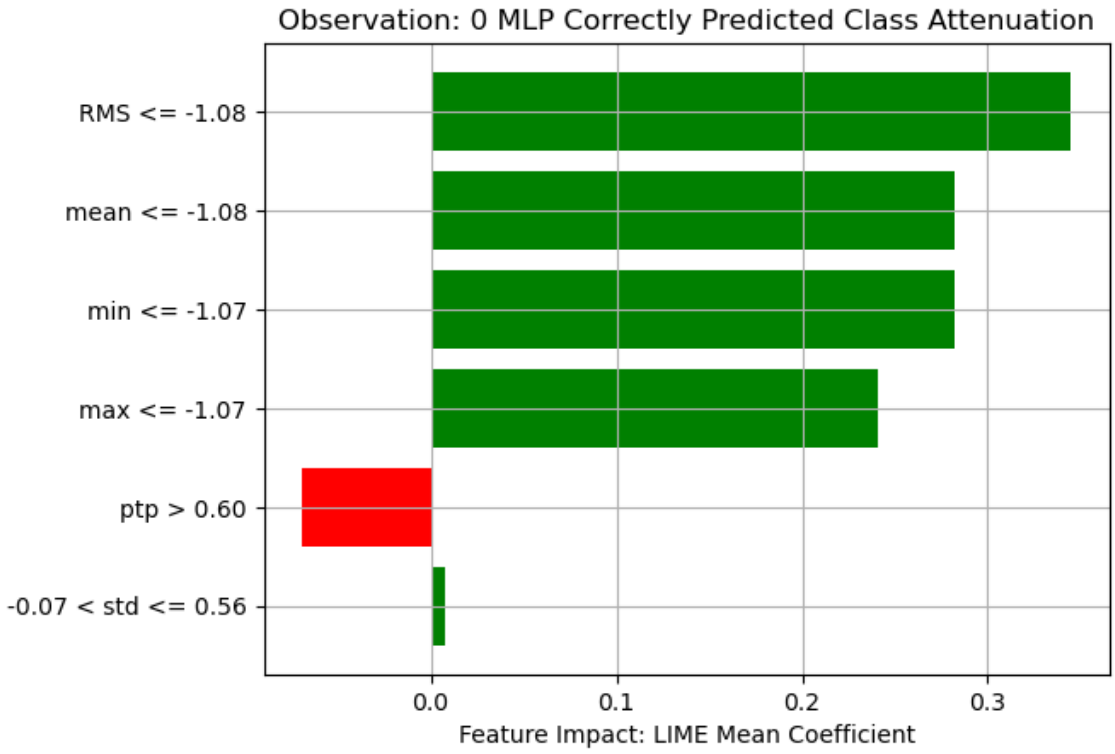
They are ordered based on the overall impact on the decision.

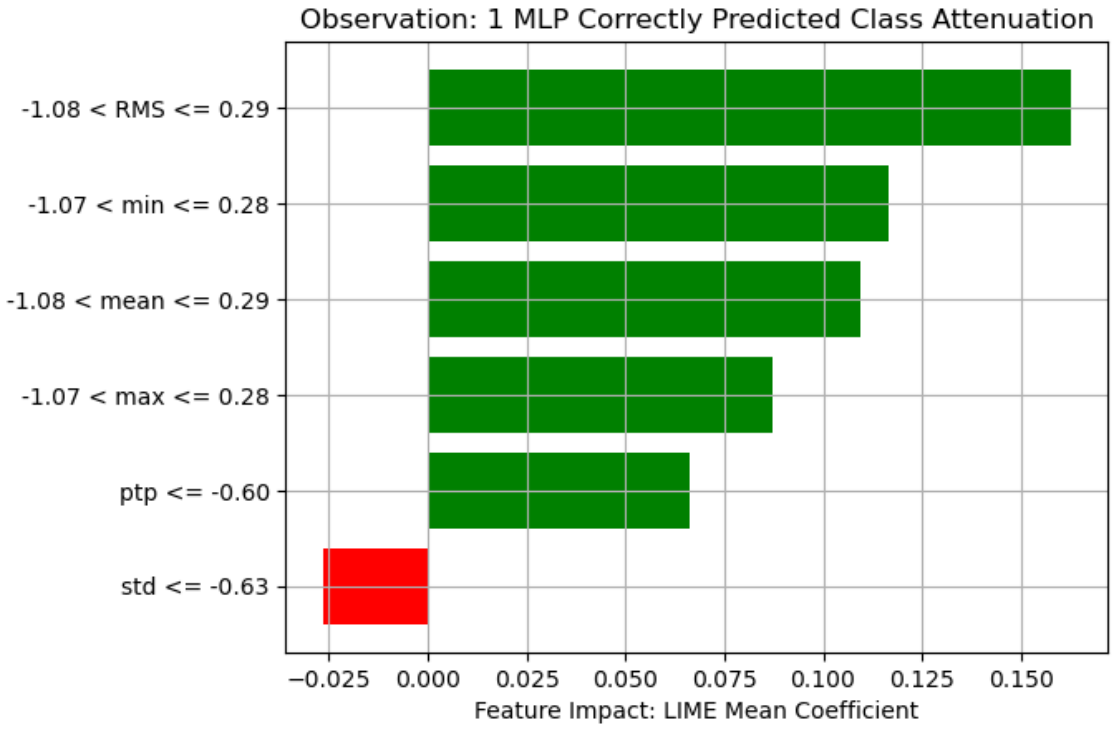
XGB LIME Plot



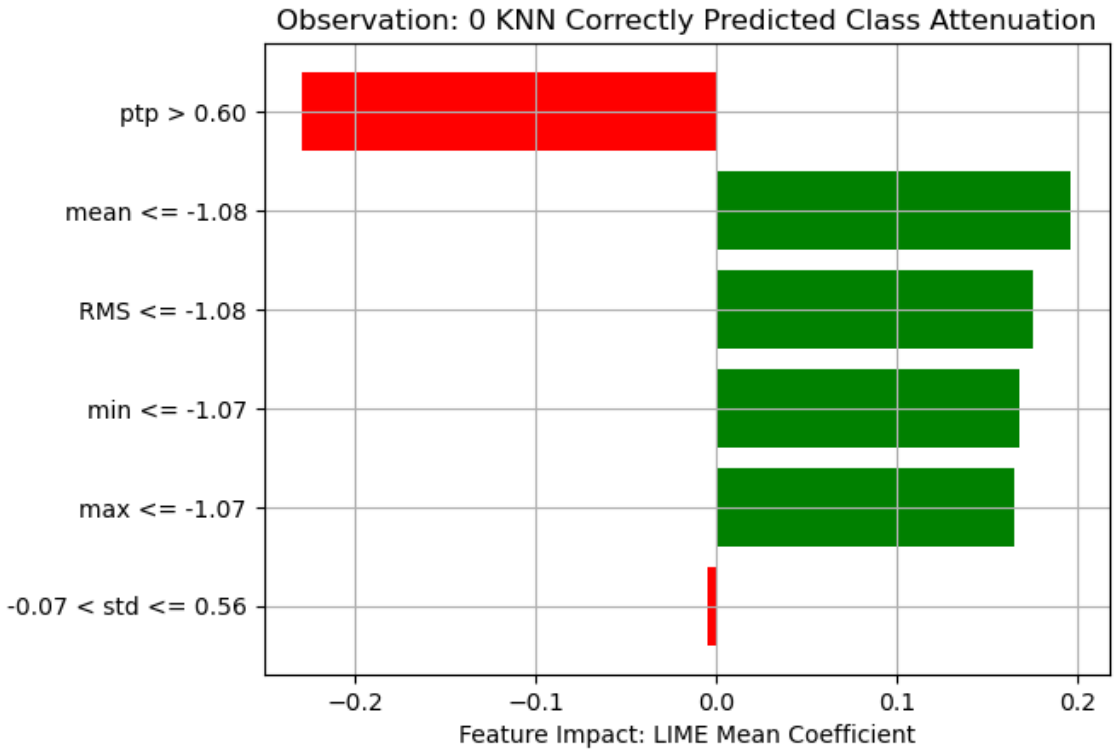


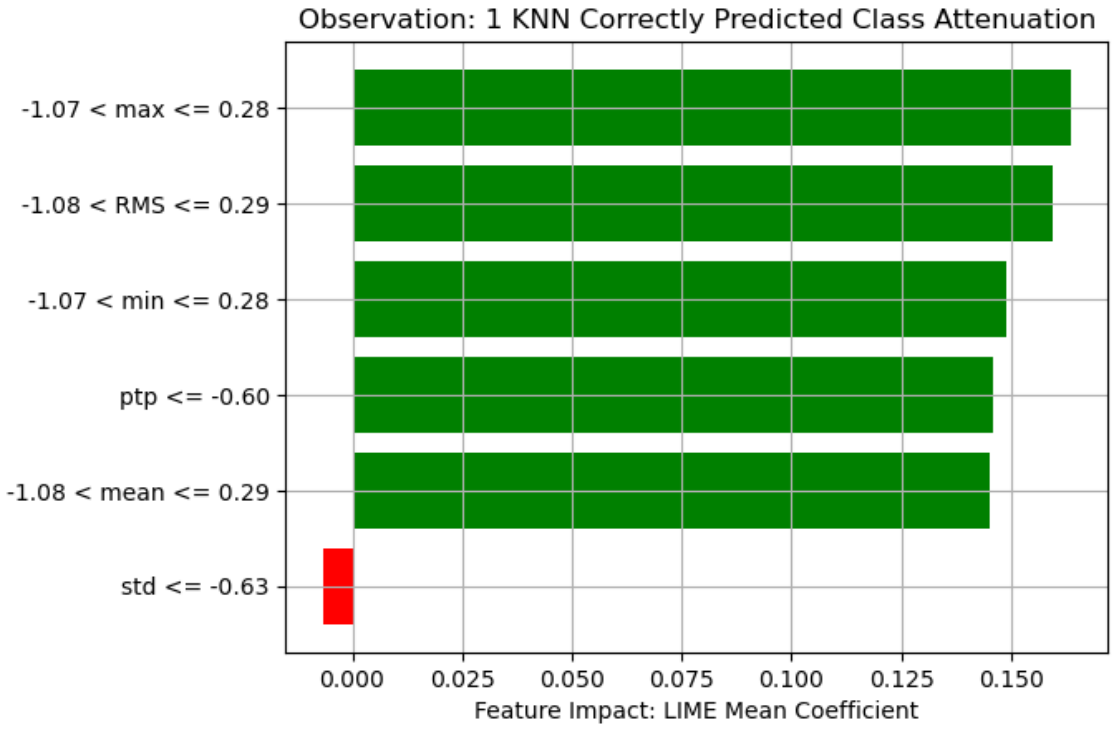
MLP LIME Plot





KNN LIME Plot




Observation: 1 KNN Correctly Predicted Class Attenuation
-1.07 < max <= 0.28
-1.08 < RMS <= 0.29
-1.07 < min <= 0.28
ptp <= -0.60
-1.08 < mean <= 0.29
std <= -0.63
0.000
0.025
0.050
0.075
0.100
0.125
0.150
Feature Impact: LIME Mean Coefficient

1.2 - What is the impact of feature normalization (MinMaxScaler) and standardization? (StandardScaler)?

Normalization

A value is **normalized** as follows:

$$y = (x - min) / (max - min)$$

Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1. Normalization ensures that all features are mapped to the same range of values. [1]

Standardization

A value is **standardized** as follows:

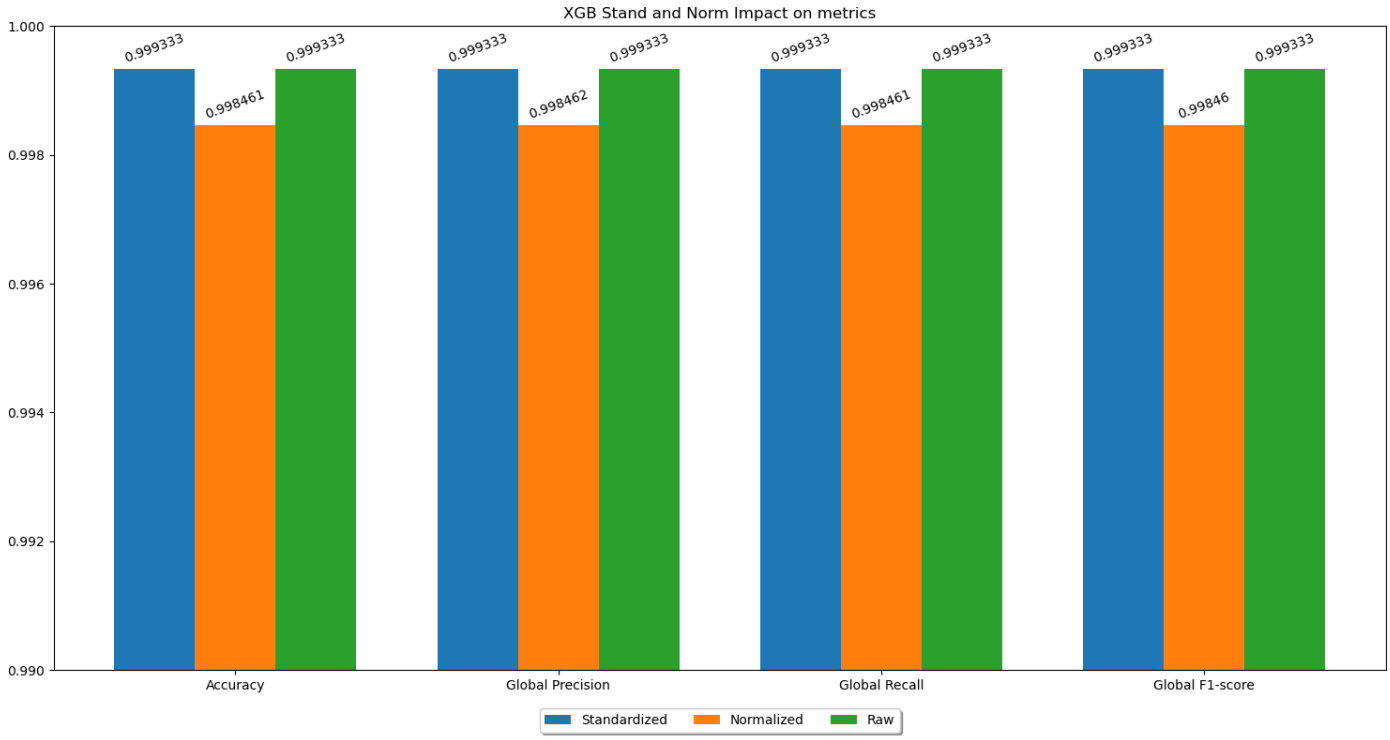
$$y = (x - mean) / standard\_deviation$$

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. Standardization doesn't ensure that the features are mapped to the same range. [1]

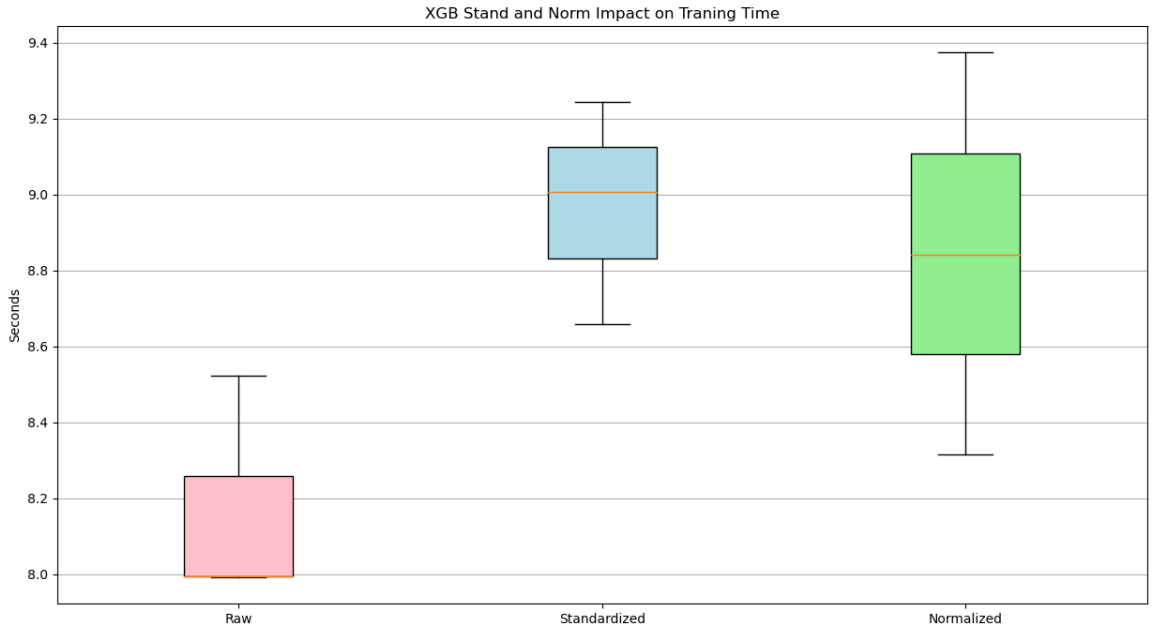
XGB

XGB is an algorithm based on Decision Trees, this type of algorithm splits the elements based on a single feature. Tree-based algorithms are **insensitive** to the scale of numerical input variables.

Metrics Comparison



Training Time Comparison

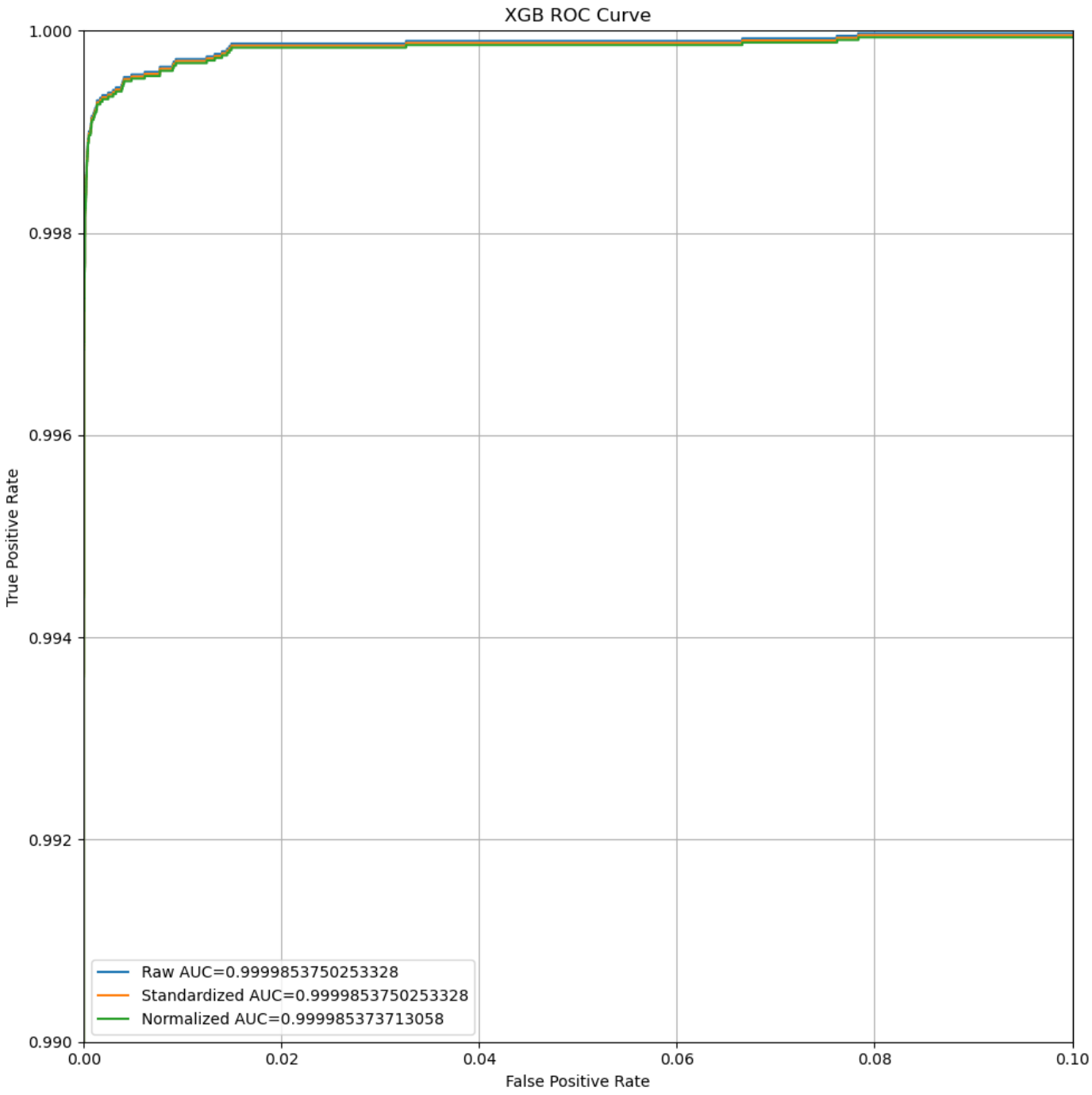


Candlesticks have been used to represent the data, which are particularly used in technical economic analysis because they can show in a single candlestick a variety of information such as in our case:

- The maximum value reached;
- The percentile between the 10 centiles and the 90 centiles;
- The median value;
- The minimum value achieved.

Standardization and Normalization are not always a good choice, in the above tests, we notice lower metrics and worse training time performance compared to the Raw version.

ROC Curve Comparison



In this test we used the **ROC Curve**, it's used to evaluate the performance of a classifier for various threshold values.

Classifiers that give curves closer to the top-left corner indicate better performance. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test is.

In this case, all tests give excellent results with an AUC (Area under curve) of about 1.

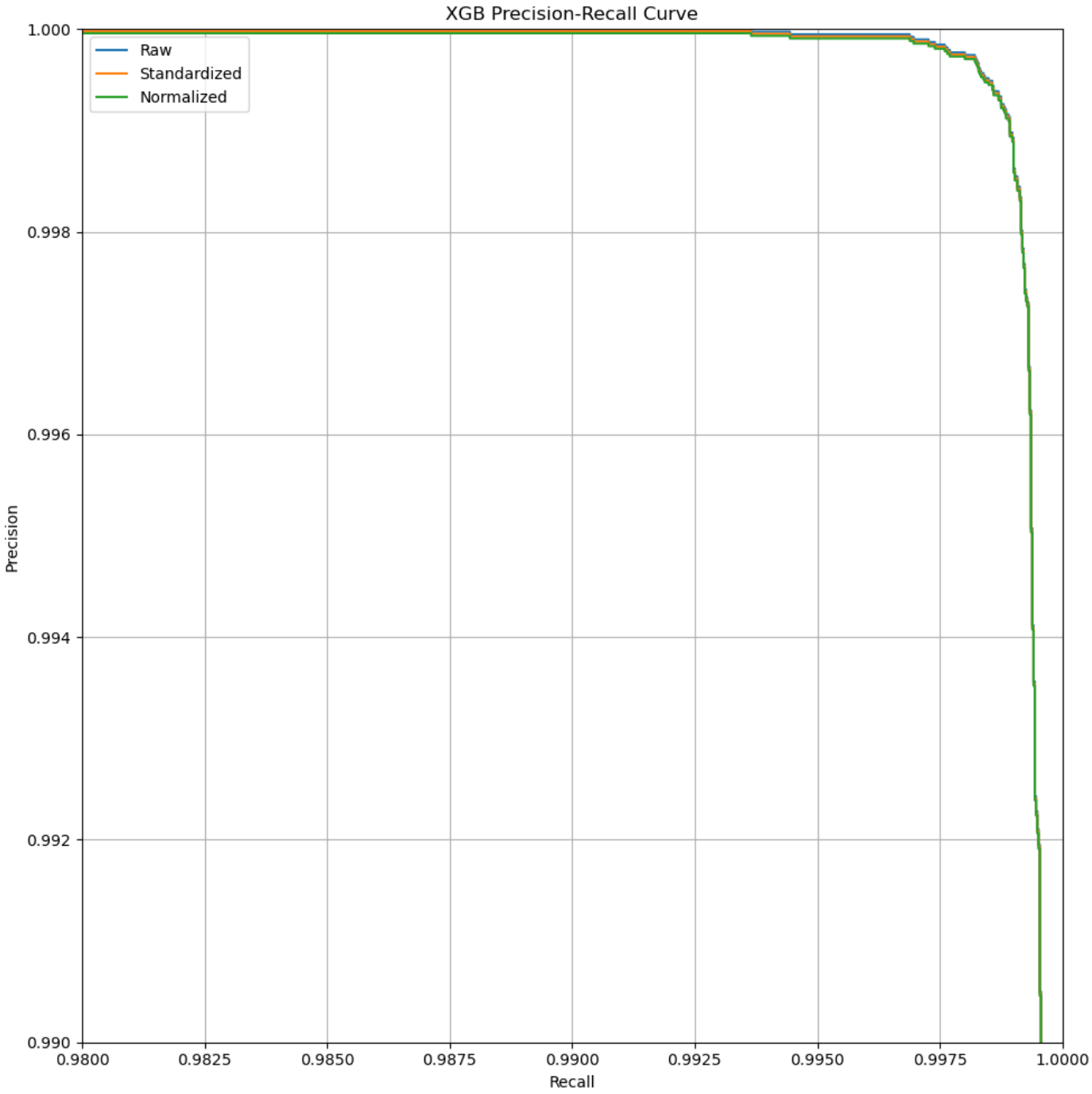
The AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. [2]

An excellent model has an AUC near 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability.

When AUC is 0.7, it means there is a 70% chance that the model will be able to distinguish between a positive class and a negative class.

A classifier with a high AUC can occasionally score worse in a specific region than another classifier with a lower AUC. But in practice, the AUC performs well as a general measure of predictive accuracy.

Precision-Recall Curve Comparison



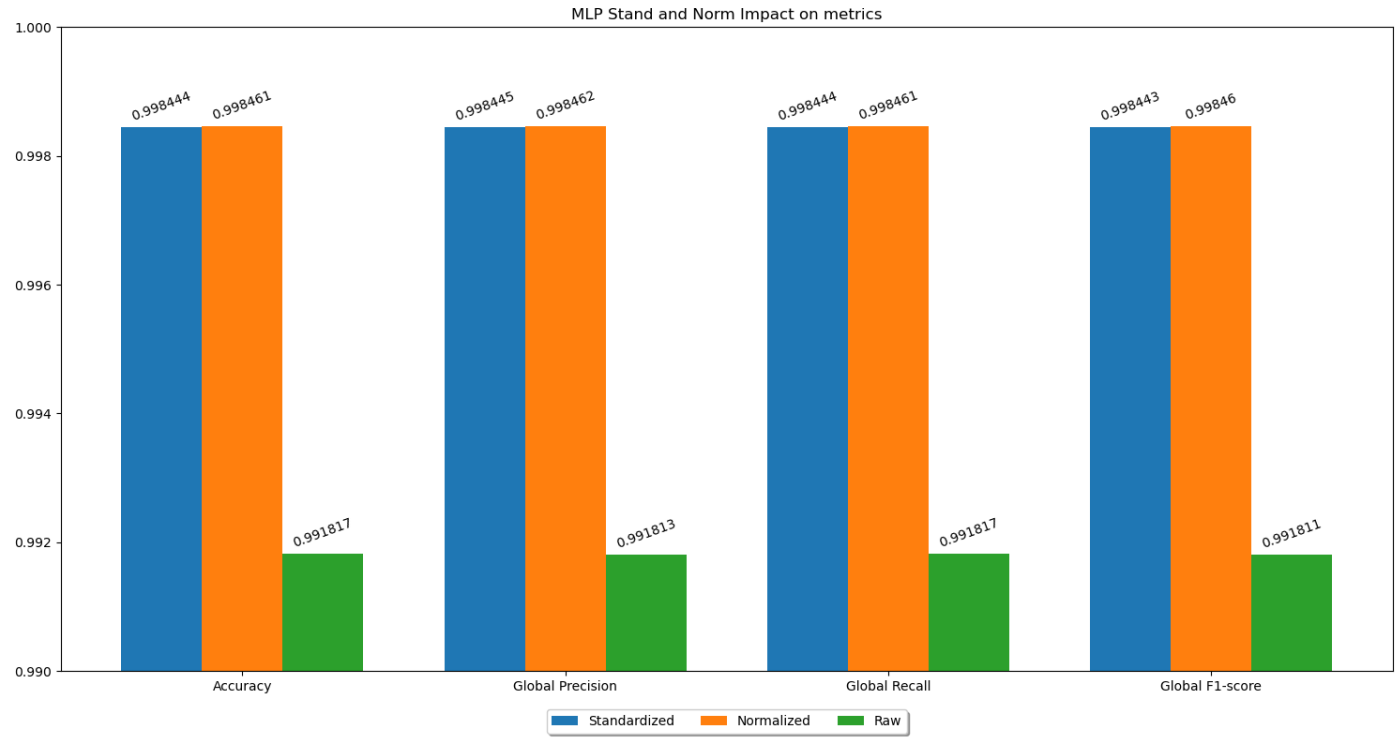
The comparison of the three different cases shown in the above tests does not highlight any change, the PR curve is the same, so they confirm what we said.

MLP

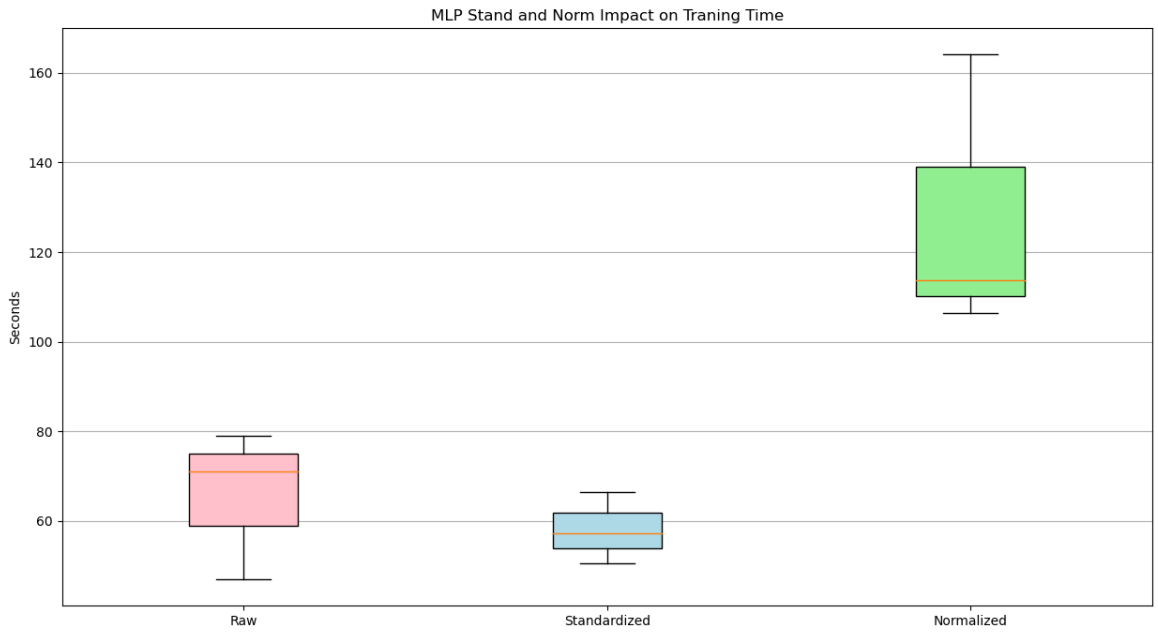
MLP (Multilayer Perceptron) belongs to the family of DNN (Deep neural networks) algorithms. In this type of algorithm standardizing the data generally speeds up learning and leads to faster convergence. A significant **training metrics improvement** can be noticed between the Raw data and the Normalized/Standardized data.

Standardizing the inputs can make training faster and reduce the chances of getting stuck in local optima.

Metrics Comparison



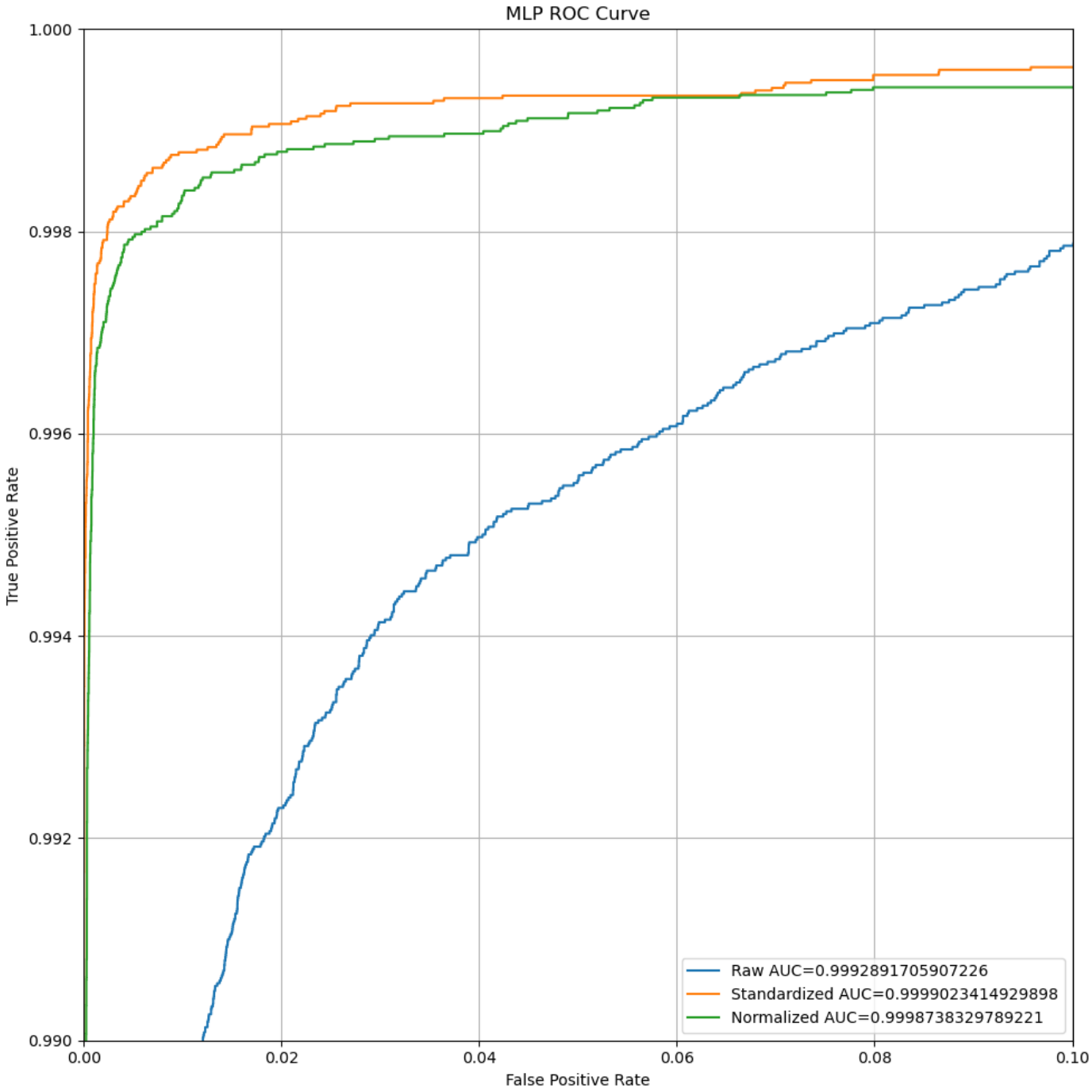
Training Time Comparison



Standardization decreased the Training Time by about 20 seconds compared to Raw data, it had a very good impact on the training time.

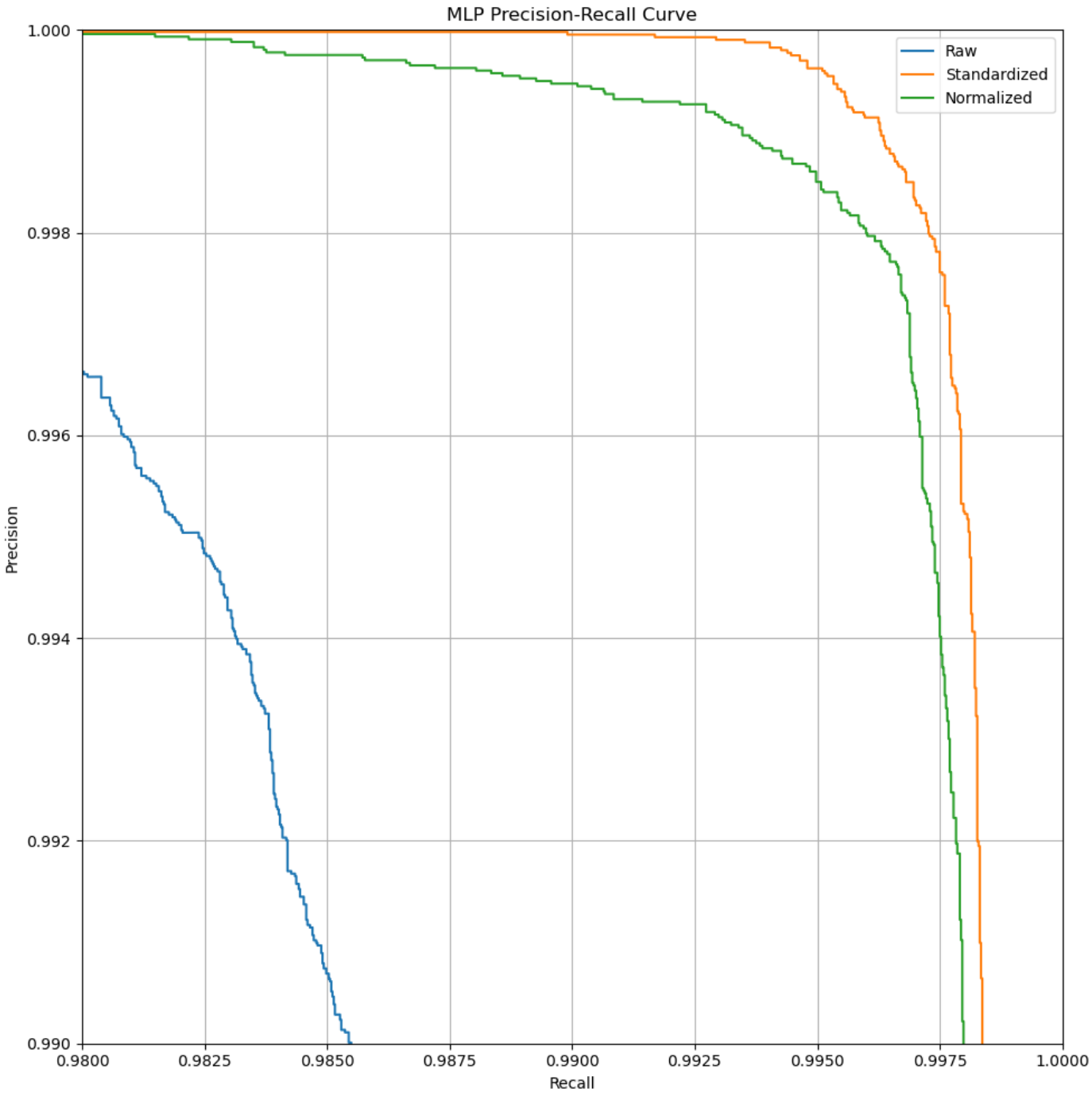
The same thing can not be said for Normalization. It increased a lot Traning Time, from about 75 seconds to about 115 seconds.

ROC Curve Comparison



As expected In the above ROC curve there is a difference between the Raw Curve and the Standardized/Normalized one, with the last two closest to 1.

Precision-Recall Curve Comparison

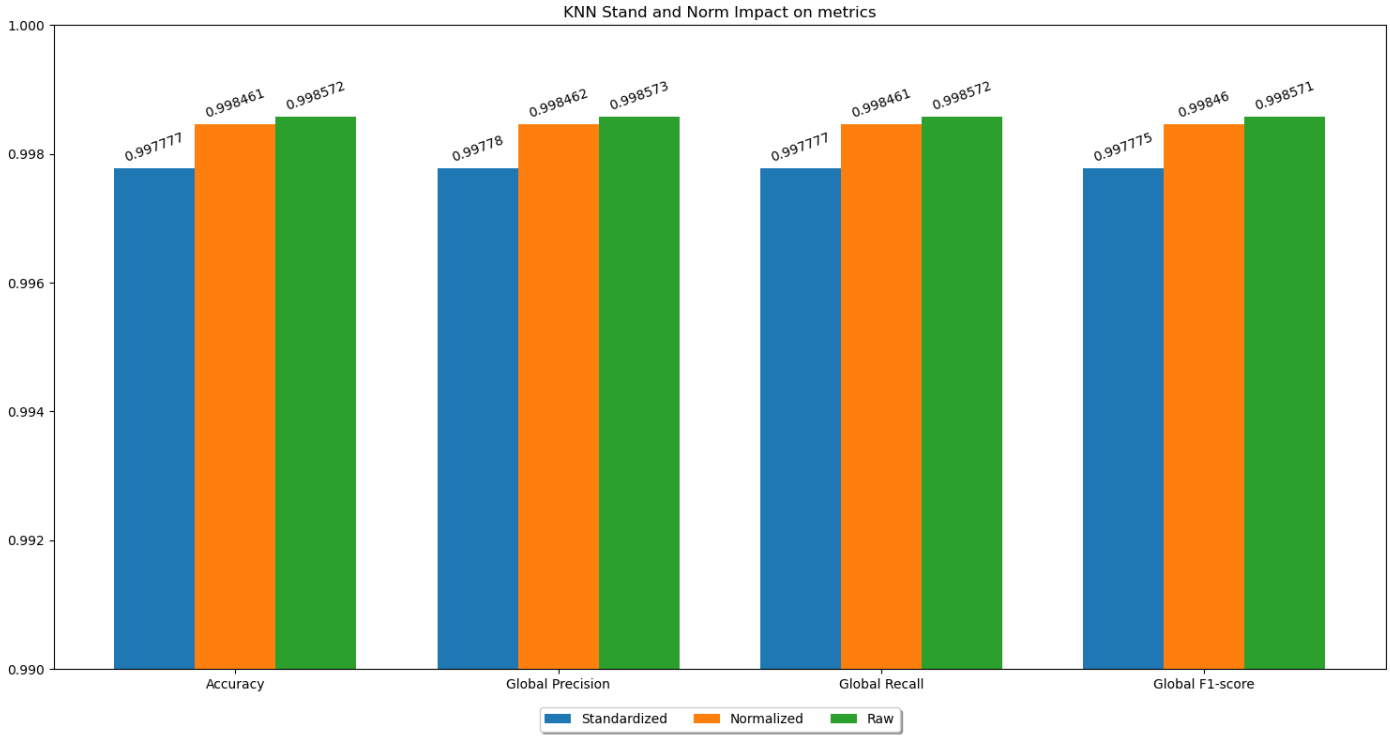


The Precision-Recall curve, like the above graphs showed how Standardized/Normalized data perform better with MLP compared to the Raw one, showing high precision and high recall.

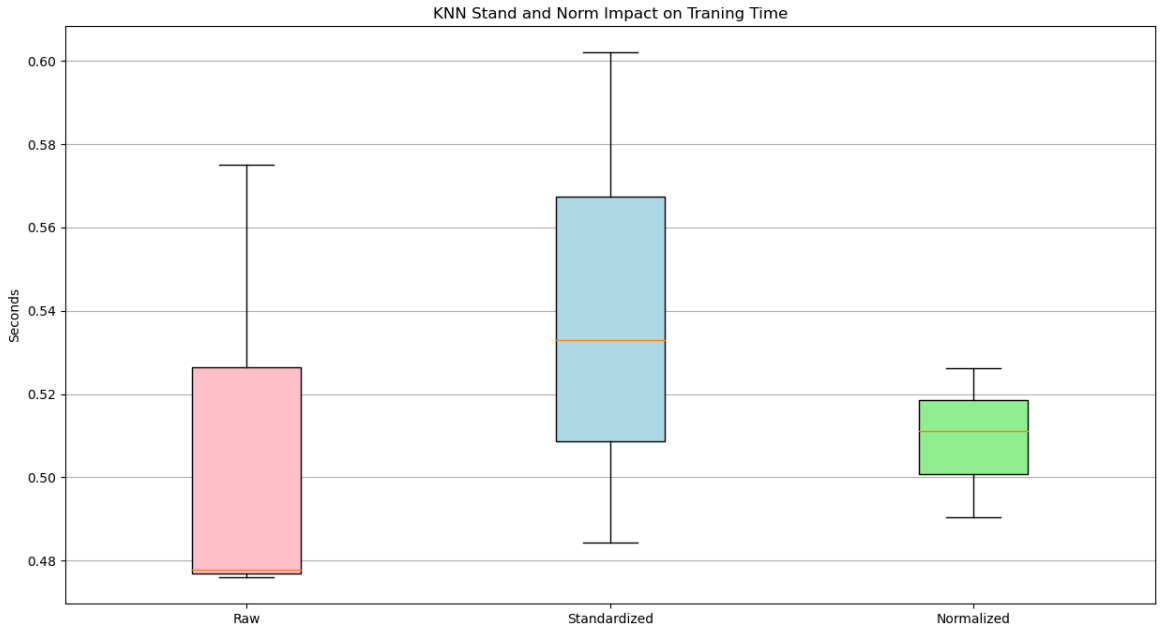
KNN

KNN stands for k-nearest neighbors algorithm, it uses the Euclidean distance as its means of comparing examples. For **all of the features to be of equal importance** when calculating the distance, the features must have the same range of values. This is only achievable through normalization, so independently from the below comparison, we know that normalization is the right thing to do.

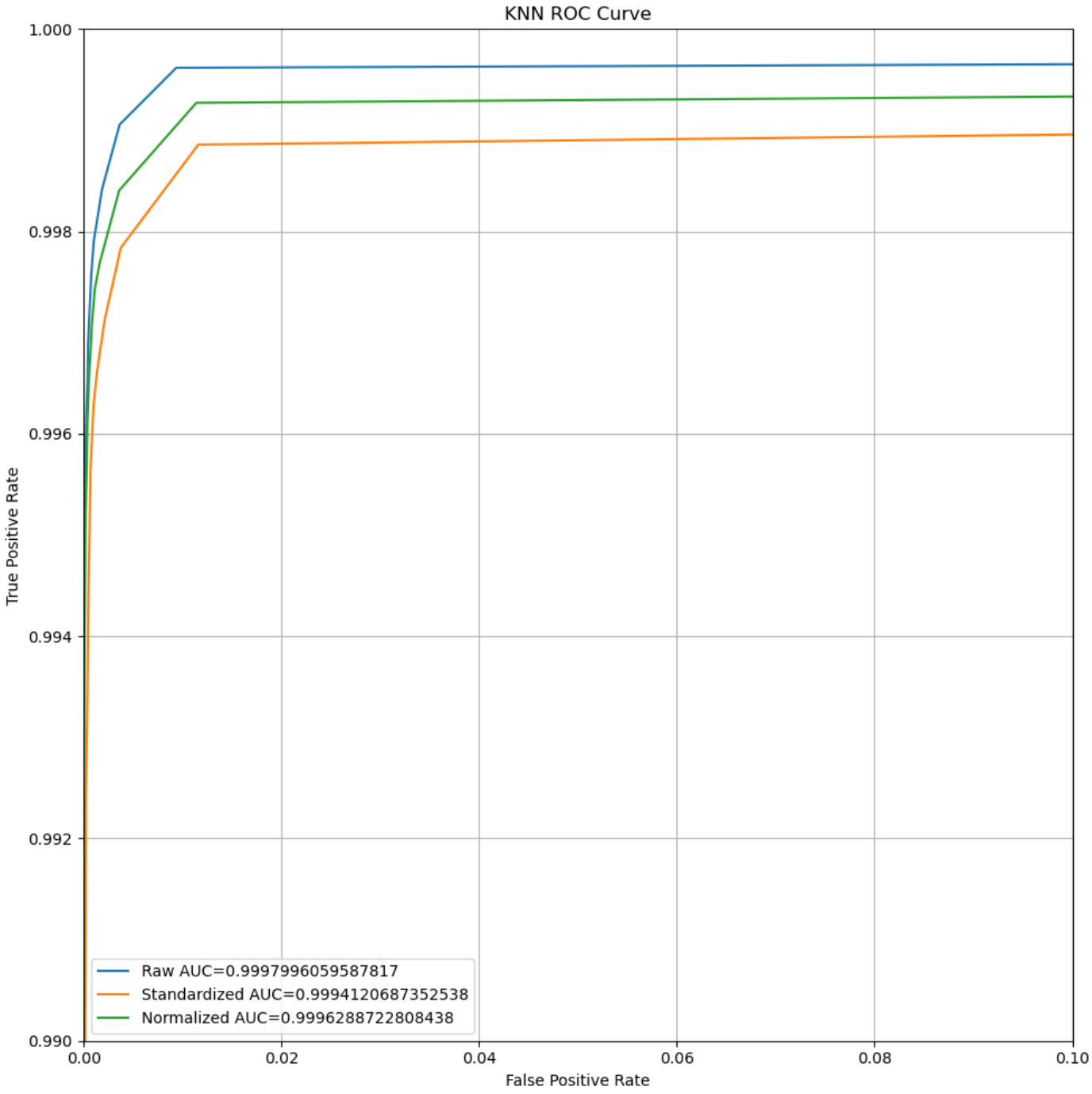
Metrics Comparison



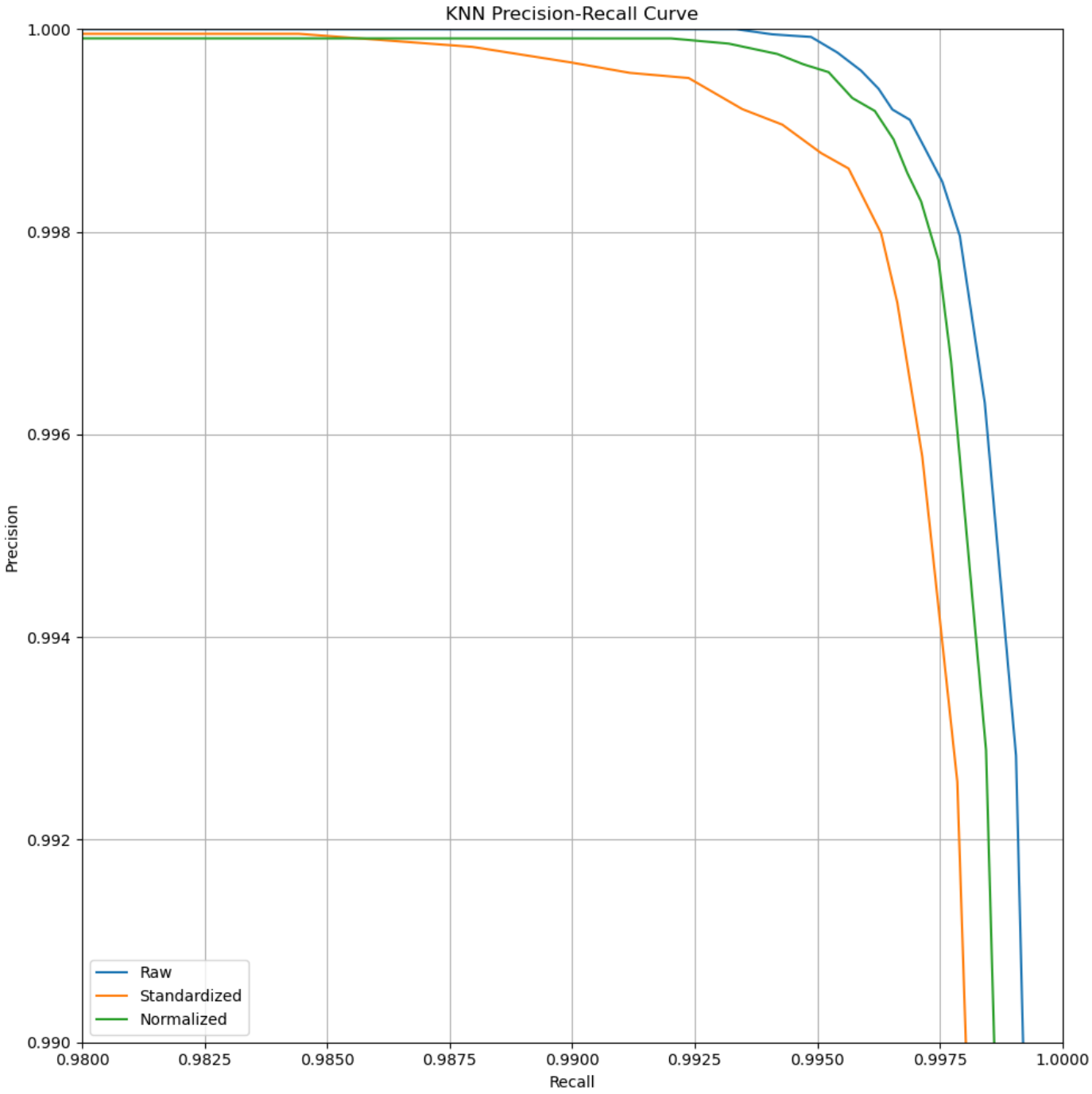
Training Time Comparison



ROC Curve Comparison



Precision-Recall Curve Comparison



1.3 - What if features are noisy or imprecisely known (e.g., as in the QoT-estimation lab)?

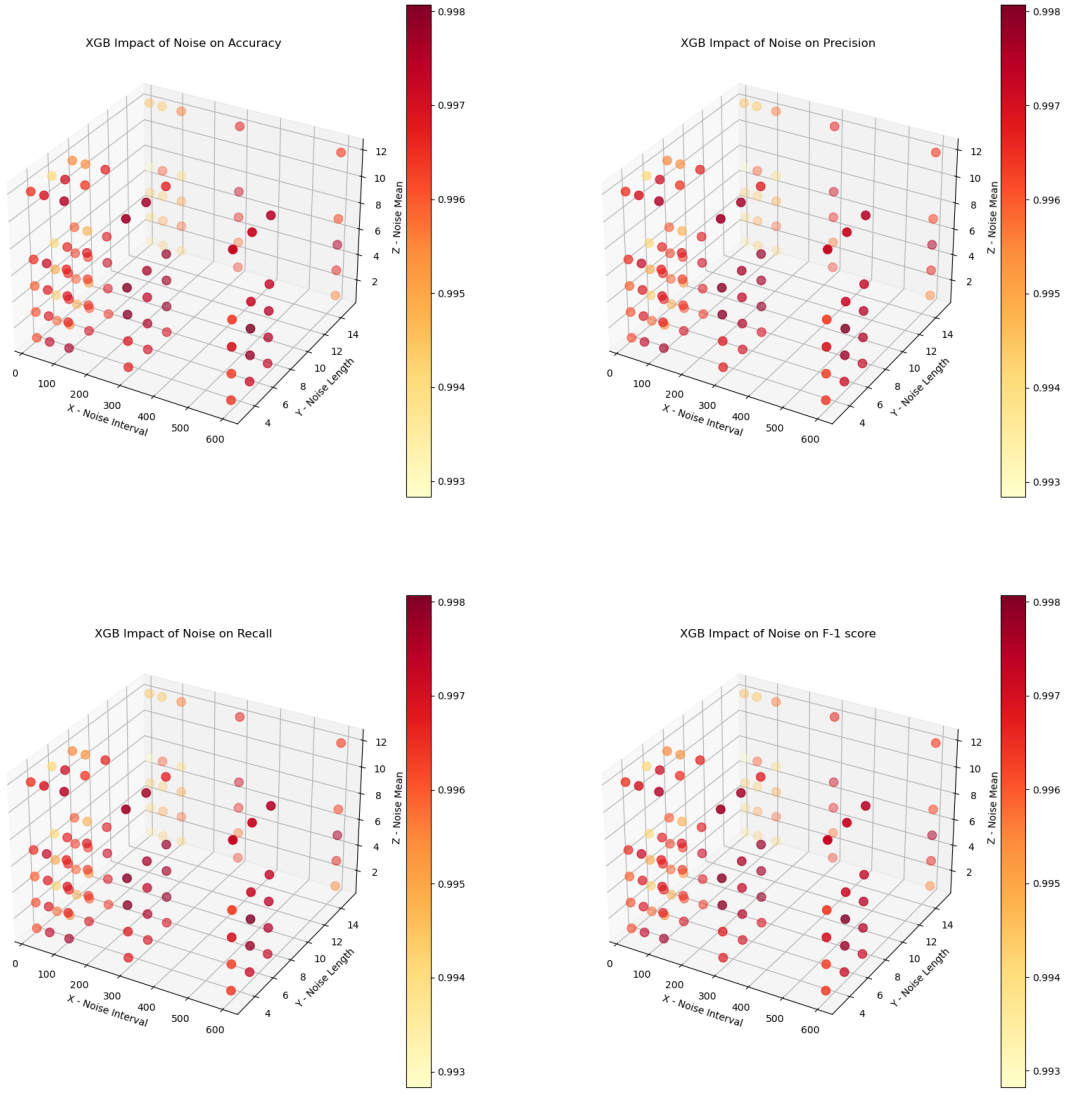
The case in which the signal is affected by some noise is also taken into consideration.

Since the features of the dataset are precisely known an additional noise has been introduced. The noise is gaussian with the mean selected on an interval from 1 to 12 and variance 1. The noise was applied to the OSNR estimate periodically, on a span that ranges from 20 to 600 seconds and with a variable length that lasts from 4 to 14 seconds.

The model has been trained over the altered dataset and tested against the raw one.

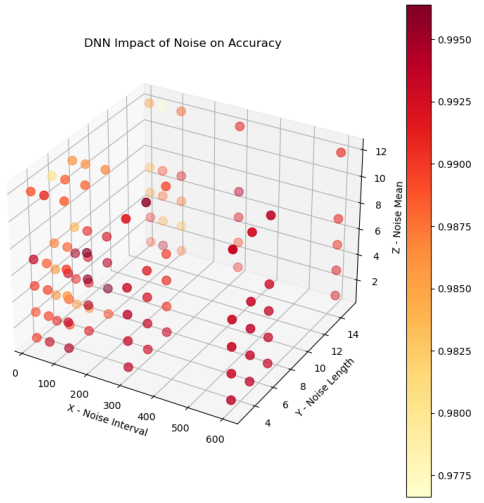
The impact of noisy features on the metrics of the model has then been plotted on a 4D heatmap (the 4th dimension is given by the color), representing how the **interval** between one application of the noise and another, noise **length,** and **mean** value affect the performances.

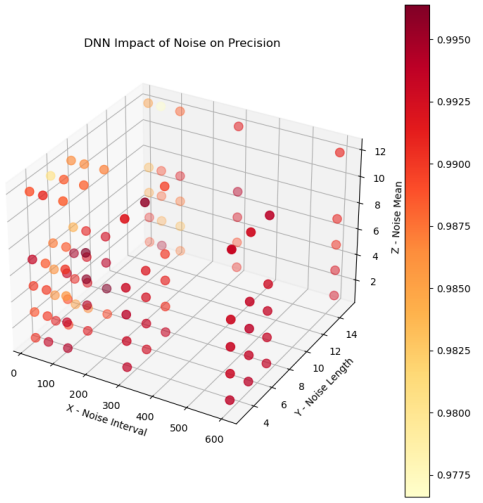
XGB Accuracy, Precision, Recall, and F-1 score

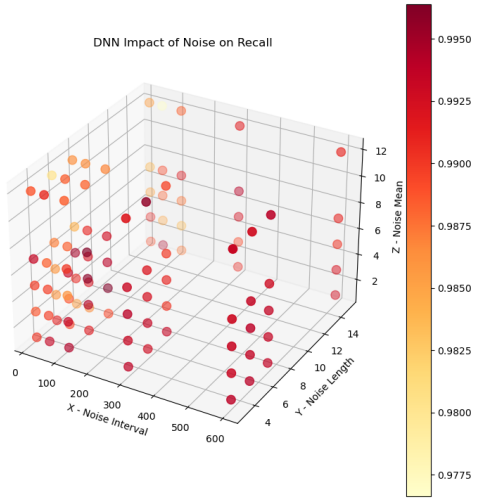


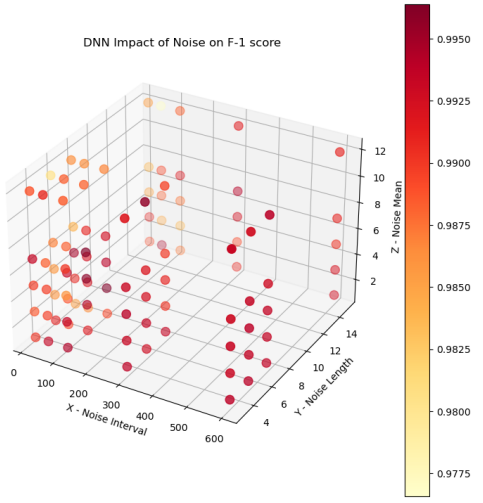
The best cases are the ones in which the noise is applied with a shorter length and at long intervals. Vice versa the lowest performances are obtained for longer noisy periods and small intervals. Similar results can be highlighted for all the other algorithms.

MLP Accuracy, Precision, Recall, and F-1 score

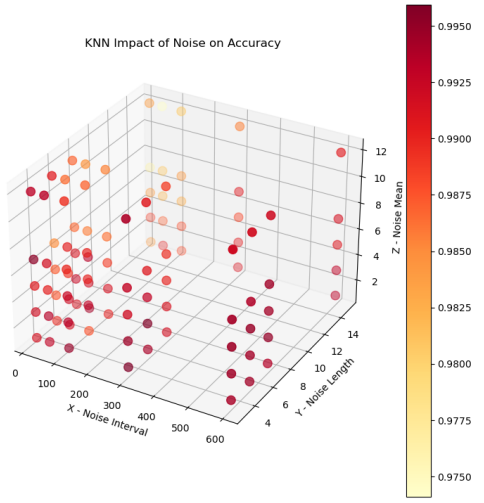


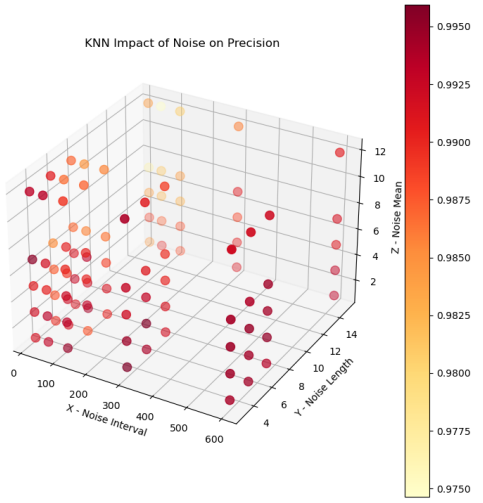


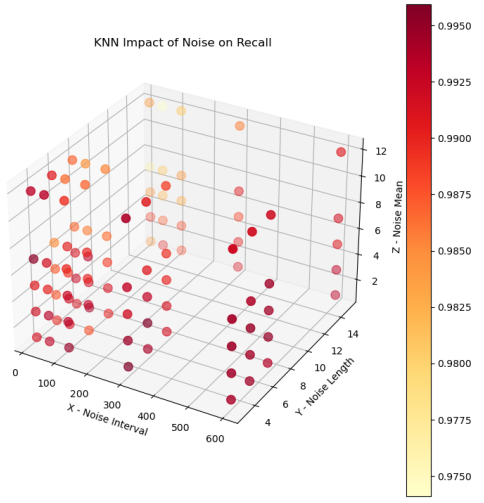


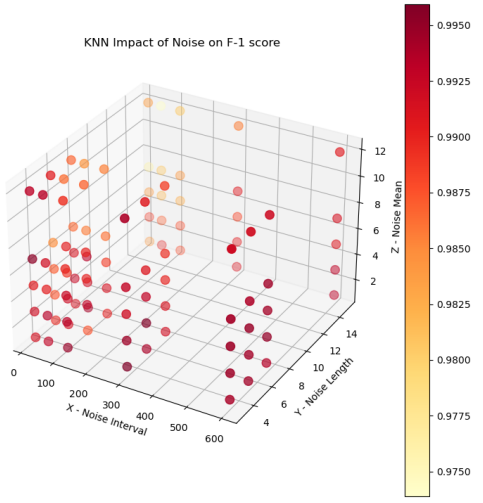


KNN Accuracy, Precision, Recall, and F-1 score









1.4 - What is the impact of training set size on algorithms' performance?

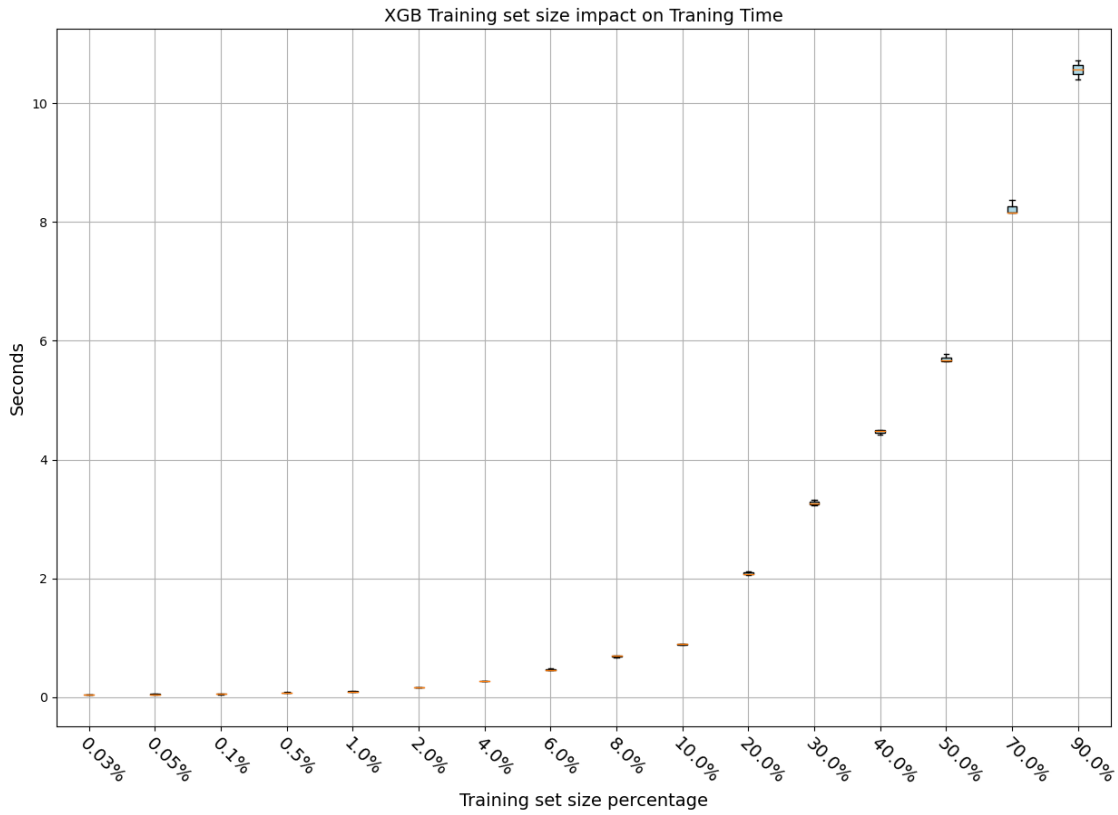
To see the impact of the training set size on the performance of the algorithms we have split the dataset according to sizes ranging from 0.01 to 0.99 for the training set (and so the test set size from 0.99 to 0.01).

We later performed the training of the model on these sub-datasets and checked and compared the performances.

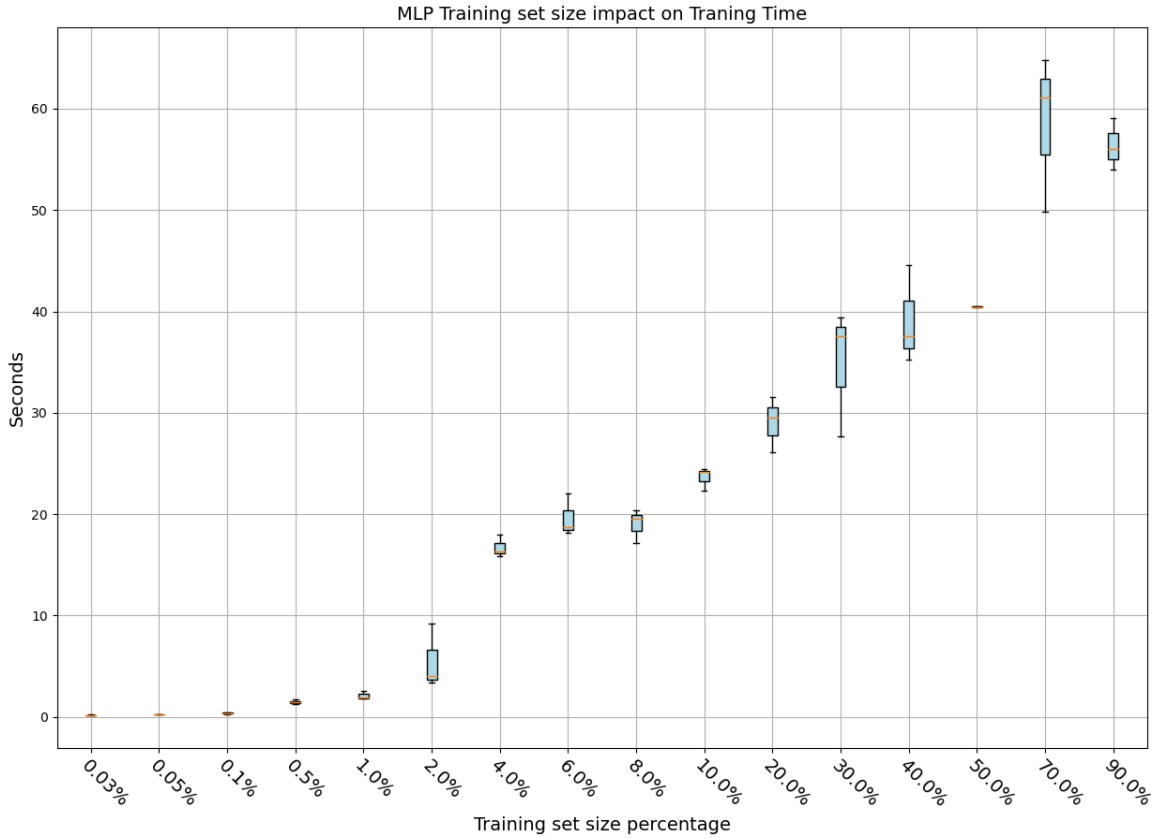
The performances related to the training time of the algorithm are, as expected, decreasing as the training set size decrease, as shown in the plots below for the different models.

This result is obvious and quite self-explaining: the more the dataset grows the more time the algorithm needs to process the data and train the model.

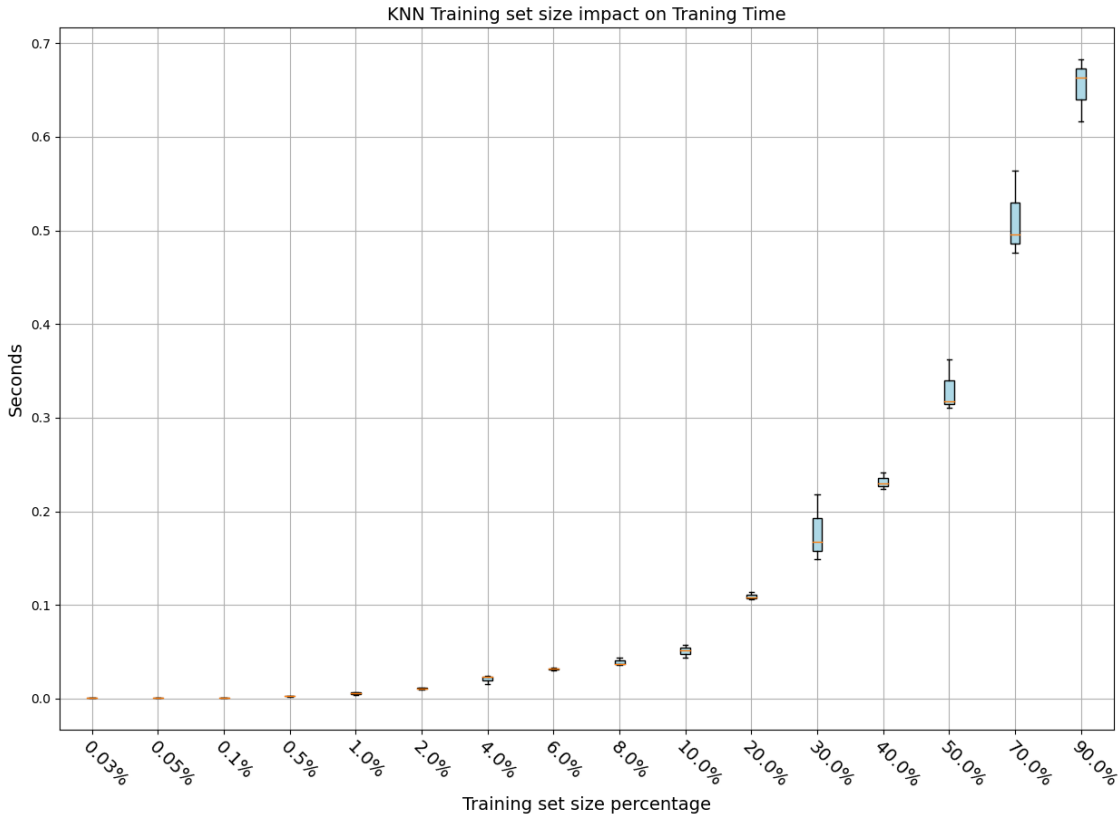
XGB Training Time



MLP Training Time

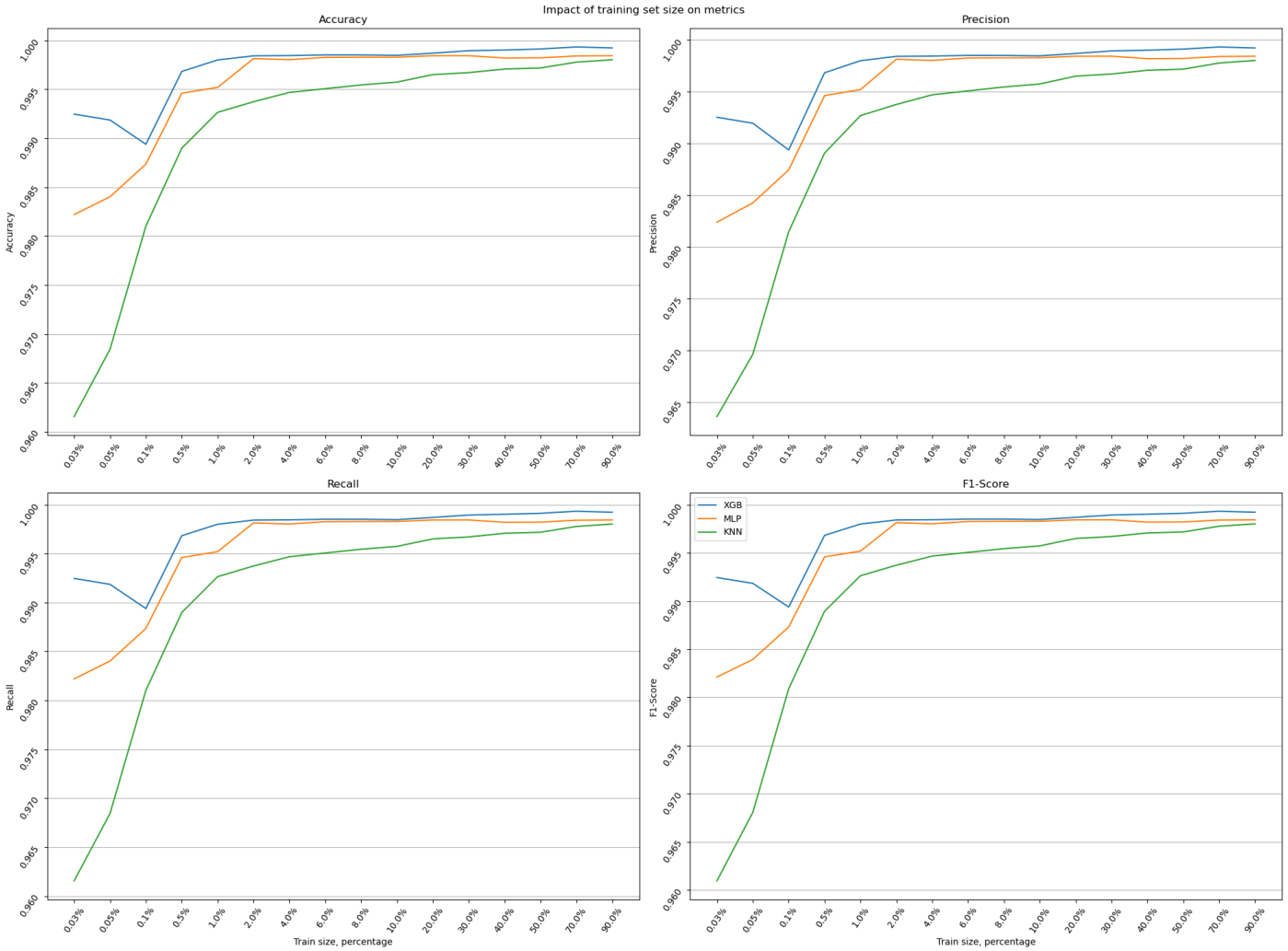


KNN Training Time



Impact of training set size on the metrics

The other performance metrics, Accuracy, Precision, Recall, and F-1 score were as expected growing as the size of the training set grew for the MLP and KNN model, peaking around 0.7.



1.5 - What happens if some features are missing for some data points? How do we handle this? What is the impact of the strategy used to handle missing data?

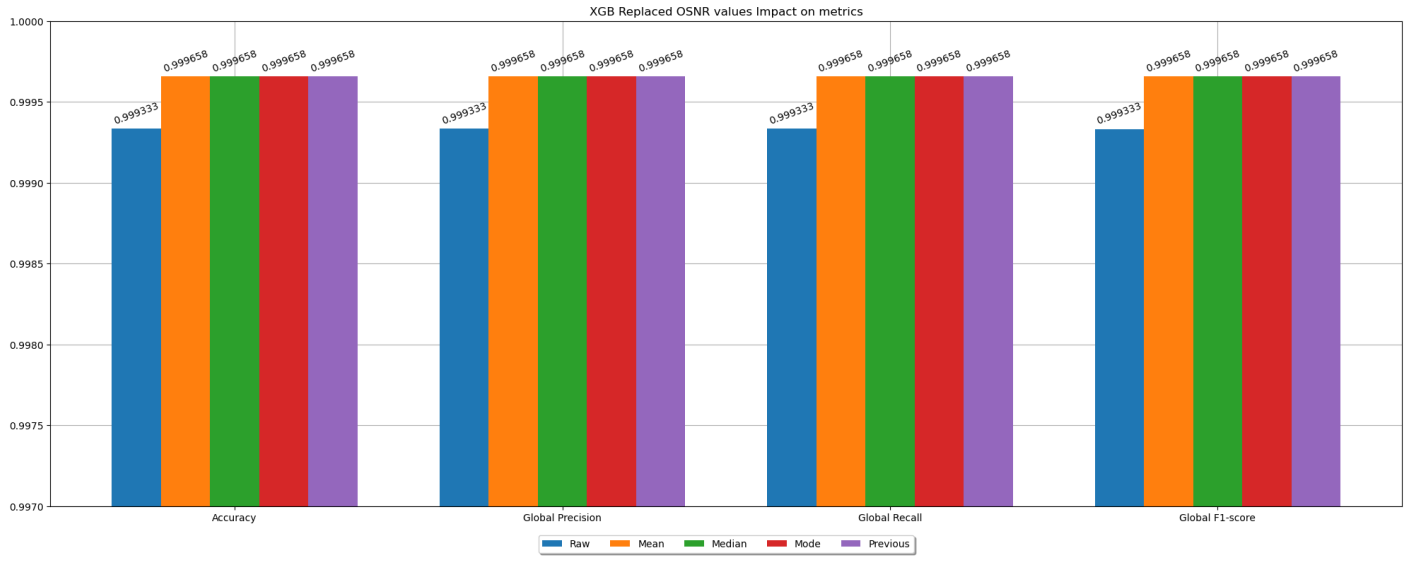
Another interesting point is the impact of missing data points on the performances. Missing data points can severely afflict performances and cause bias in the estimation of the metrics.

Since the dataset was not missing any, a new one has been generated, randomly erasing OSNR data points with percentages as high as 10%, for the sake of imagining a critical scenario.

To check the impact of different missing values replacement strategies different solutions have been tried. More in detail the missing data points were replaced by the **mean**, **median**, **mode,** or the **previous value**.

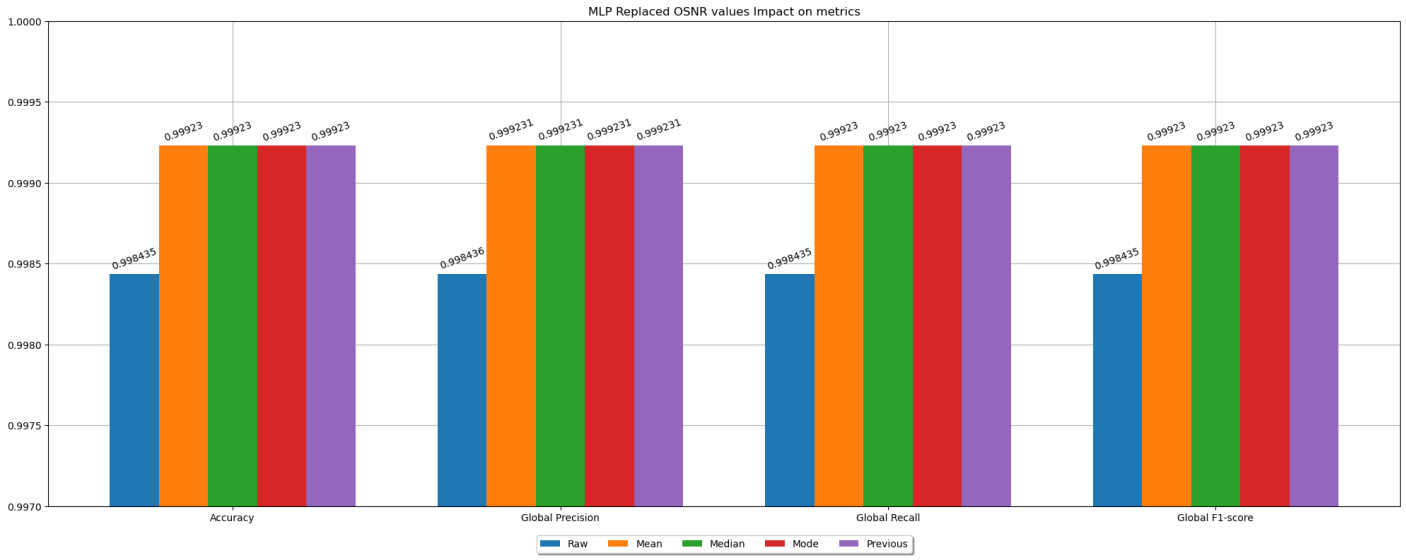
The comparison between the results of the models tested against the altered dataset and the raw one has been computed and plotted below.

XGB Accuracy, Precision, Recall, and F-1 score

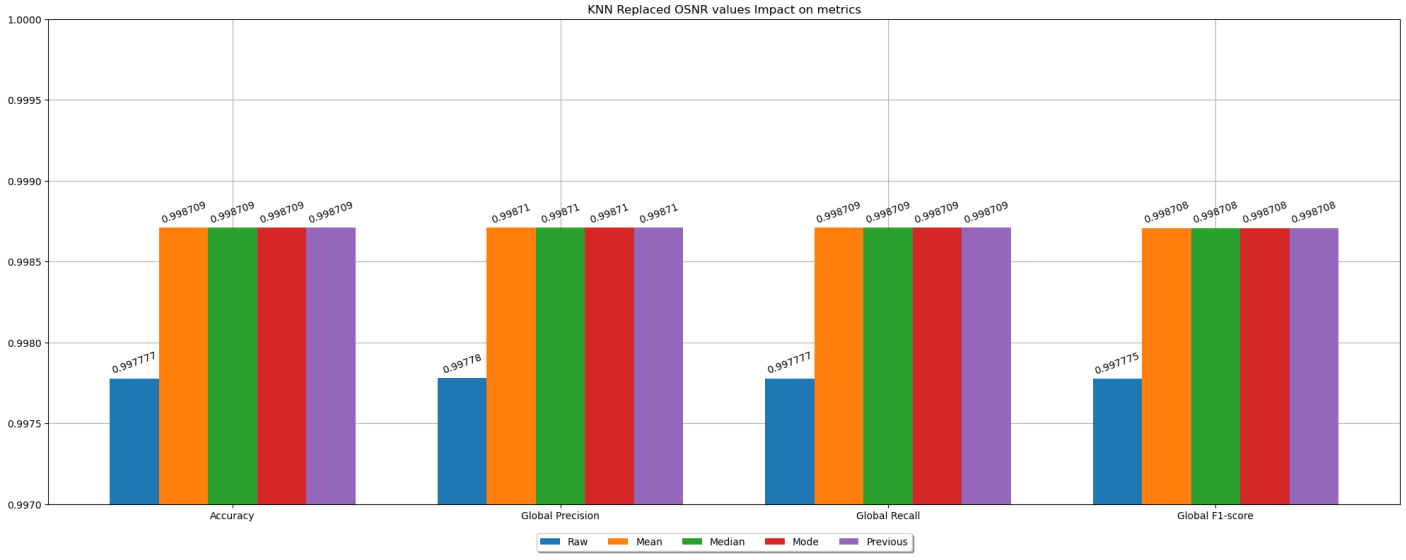


On average the deleted and replaced valued dataset had a positive impact on the training since every metric performance proved better than the raw dataset case.

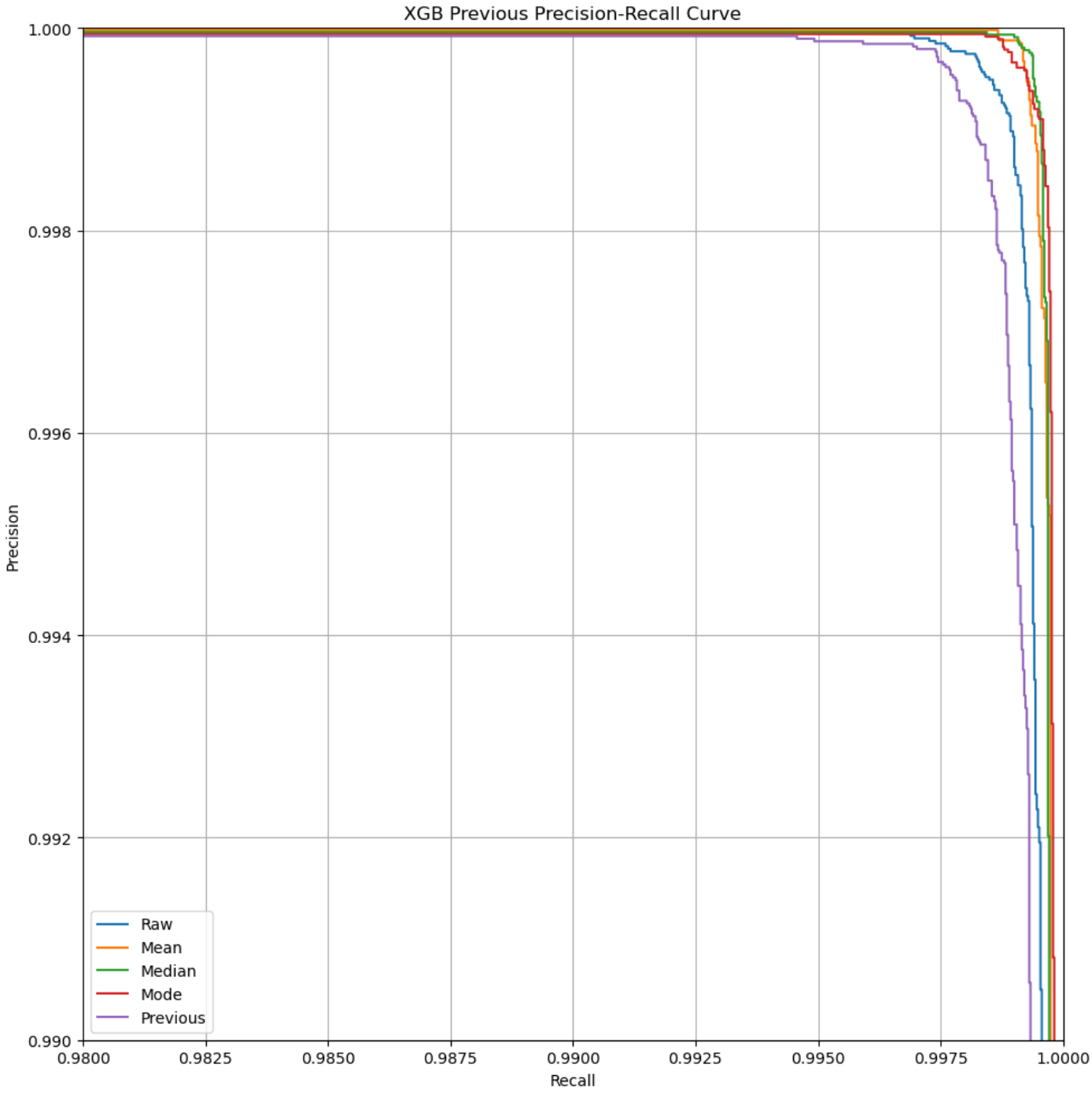
MLP Accuracy, Precision, Recall, and F-1 score

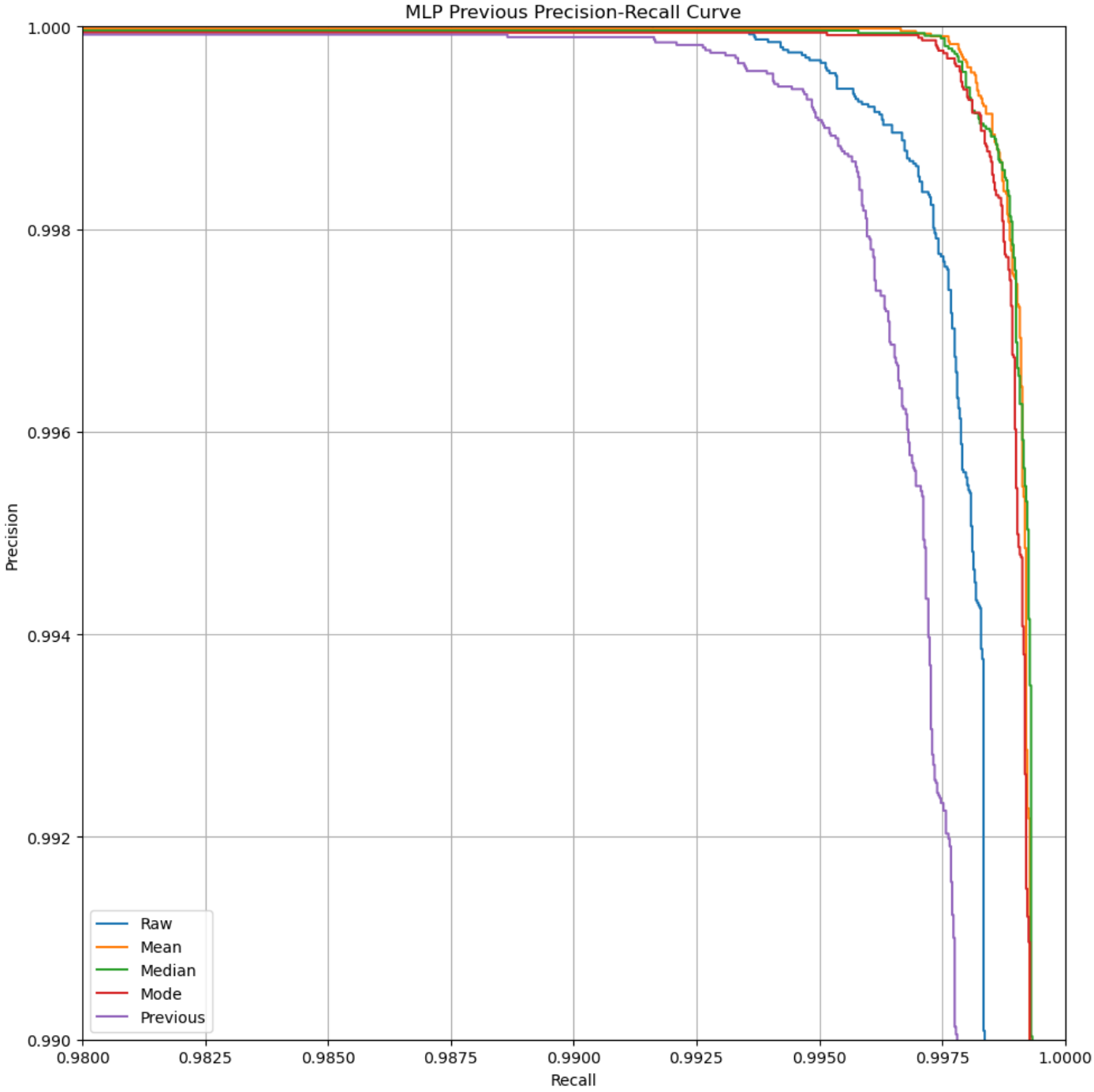


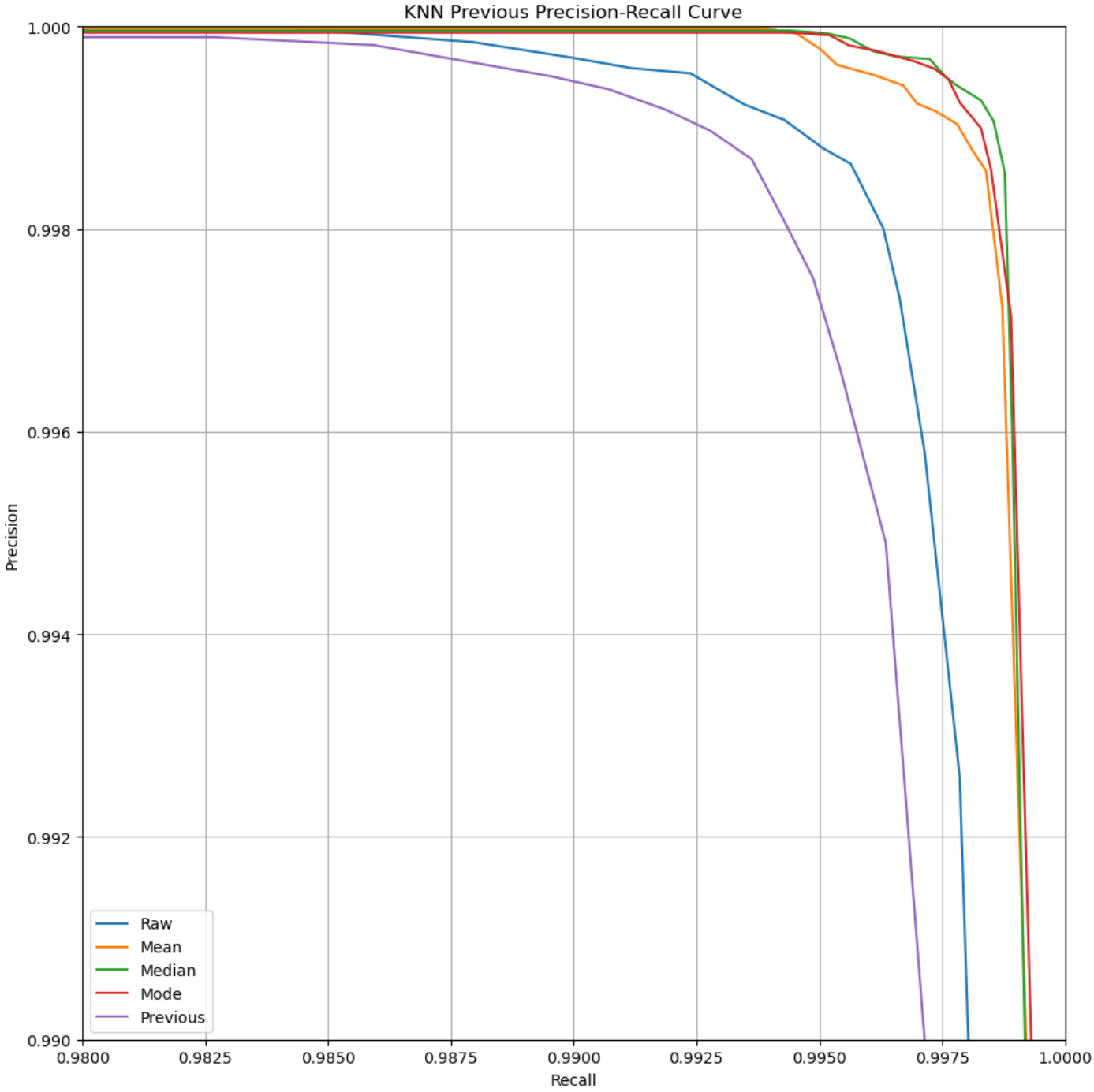
KNN Accuracy, Precision, Recall, and F-1 score



Precision-Recall Curves

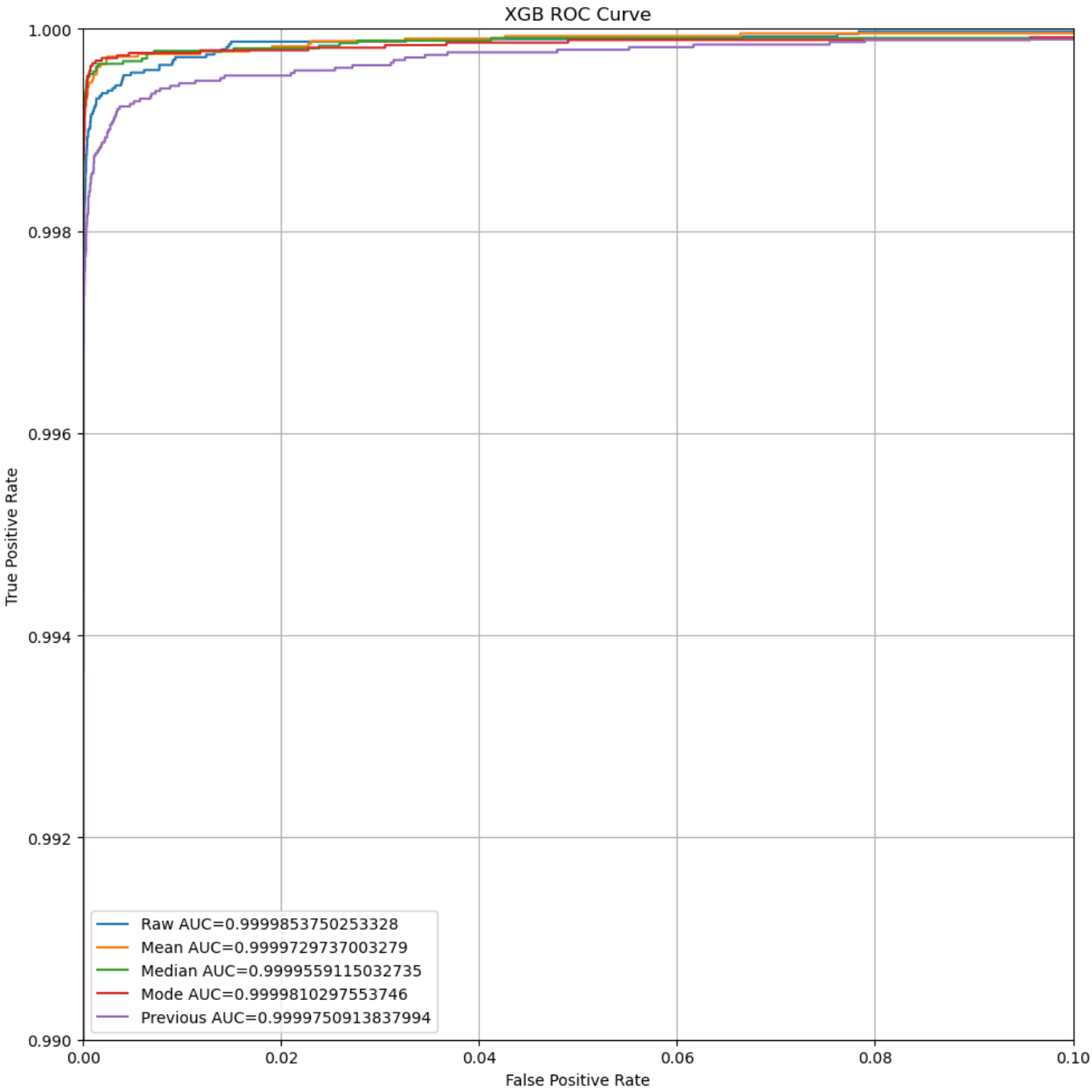


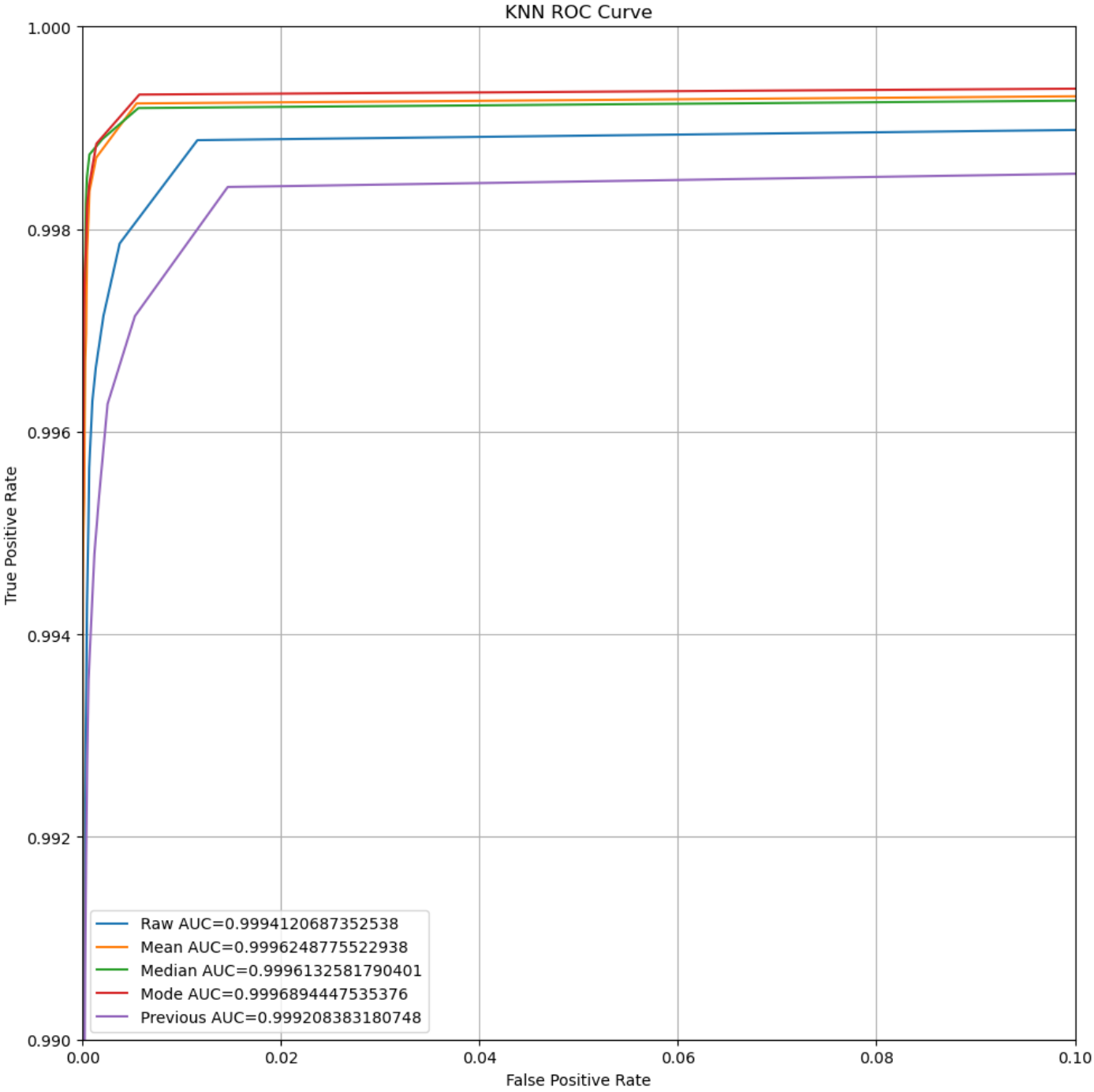

MLP Previous Precision-Recall Curve
Precision
Recall
1.000
0.998
0.996
0.994
0.992
0.990
0.9800
0.9825
0.9850
0.9875
0.9900
0.9925
0.9950
0.9975
1.0000
Raw
Mean
Median
Mode
Previous

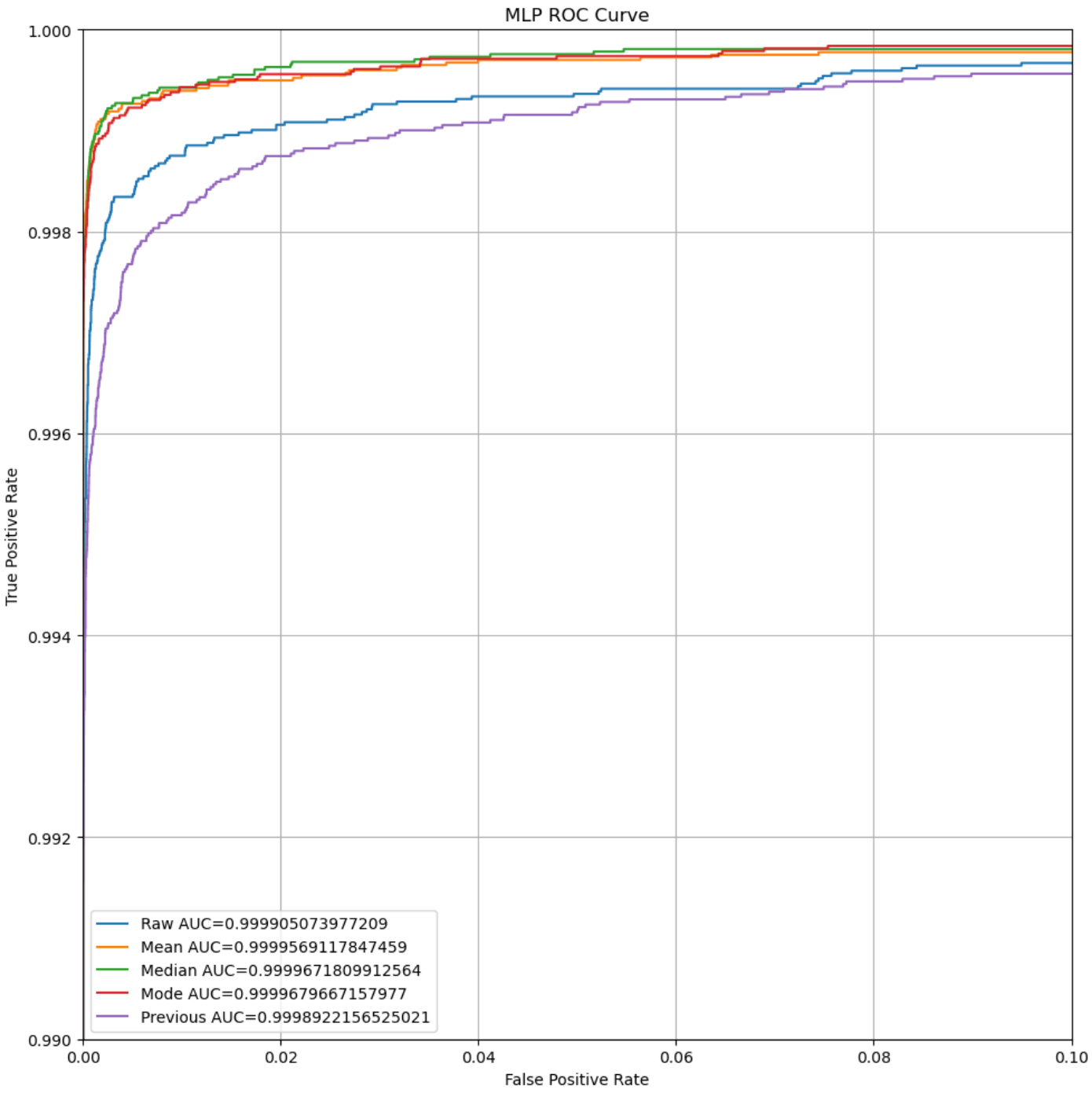


This series of graphs show the same trend as the previous one. In general worse results are achieved with the raw dataset. The replacement with the previous value had the worst impact on performances.

ROC Curves


XGB ROC Curve
True Positive Rate
False Positive Rate
1.000
0.998
0.996
0.994
0.992
0.990
0.00
0.02
0.04
0.06
0.08
0.10
Raw AUC=0.9999853750253328
Mean AUC=0.9999729737003279
Median AUC=0.9999559115032735
Mode AUC=0.9999810297553746
Previous AUC=0.9999750913837994


KNN ROC Curve
True Positive Rate
False Positive Rate
1.000
0.998
0.996
0.994
0.992
0.990
0.00
0.02
0.04
0.06
0.08
0.10
Raw AUC=0.9994120687352538
Mean AUC=0.9996248775522938
Median AUC=0.9996132581790401
Mode AUC=0.9996894447535376
Previous AUC=0.999208383180748



The ROC curves show the same trend as the previous series of graphs. The worst results are the ones with the previous value replacement approach.

A slight increment, below 0,0001, in the performance of the models can be noticed. This may happen because the dataset with missing and replaced values are more uniform than the raw dataset.

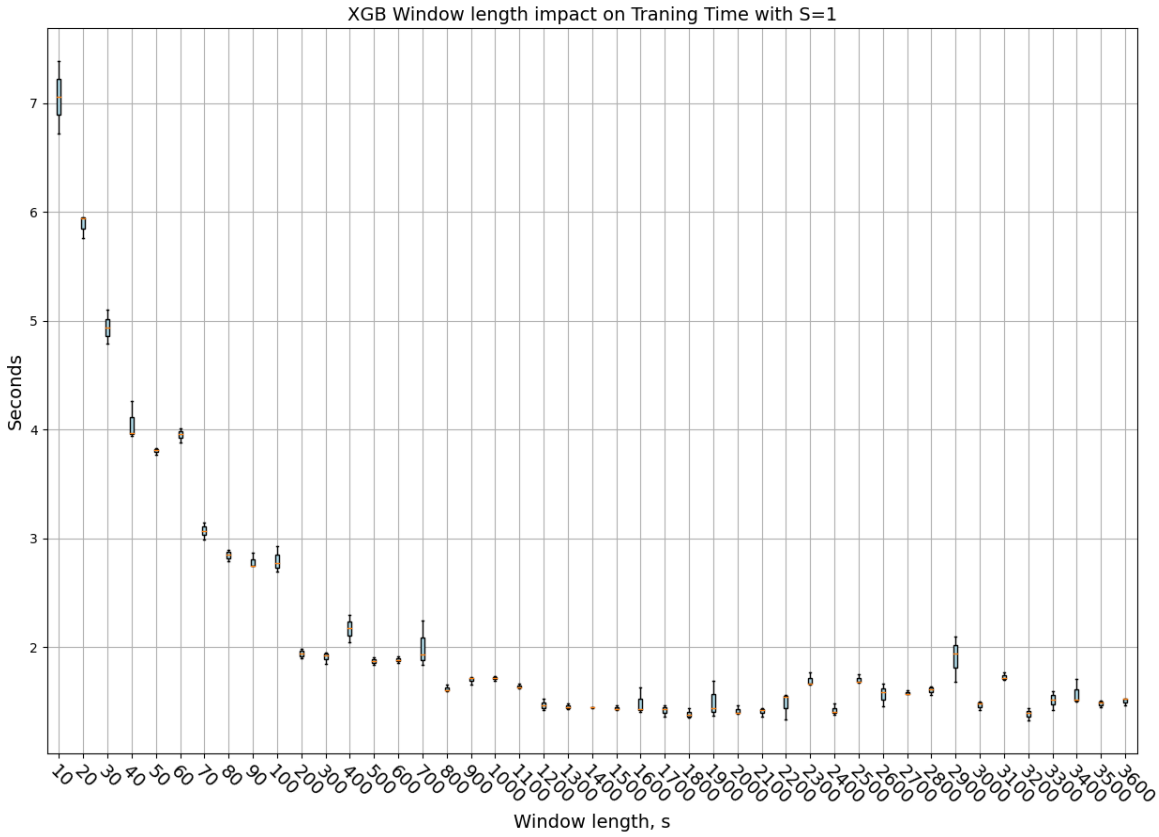
1.6 - What is the impact of window duration and window spacing on the performance?

The **window duration** is the number of samples the features are extracted over, assuming the signal is sampled once every second. In contrast, **window spacing** is the distance in time between two windows.

For the following analysis, the results are being carried out by fixing the spacing time and later the window size.

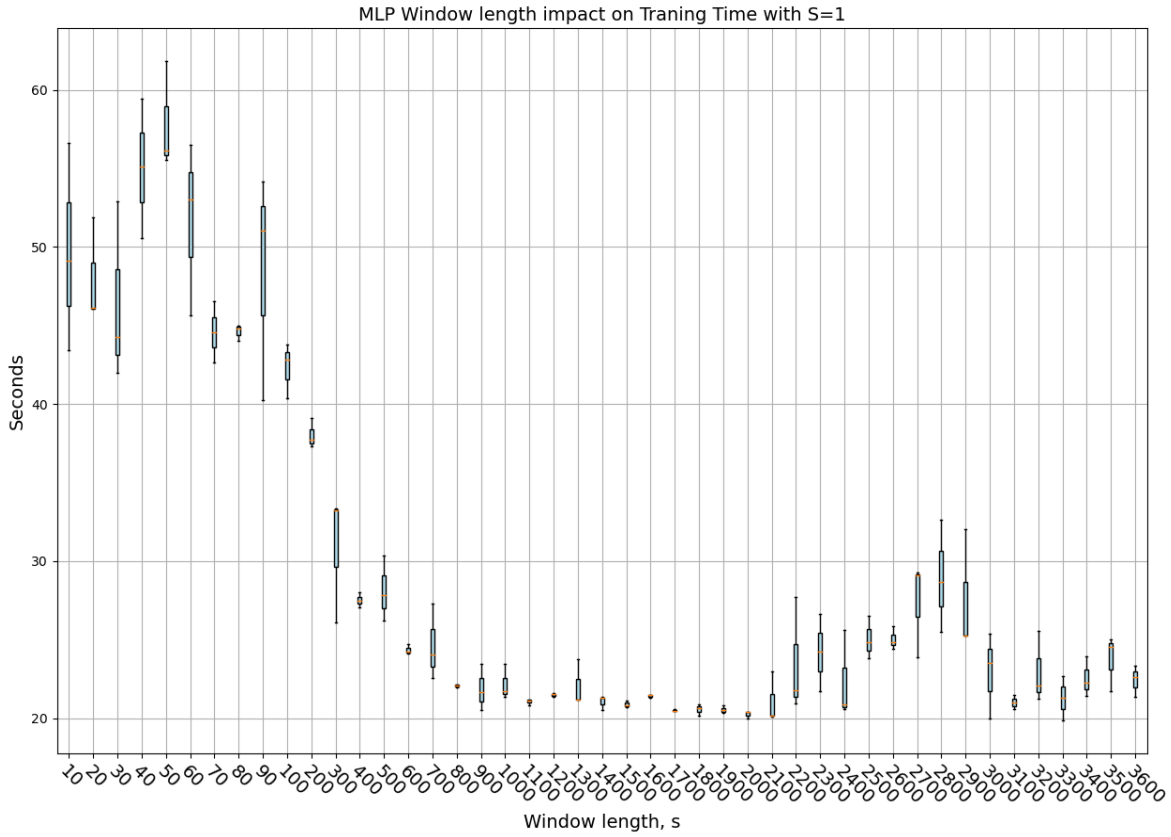
In the below tests, the spacing is fixed to 1 second and the duration is ranged from 10 to 3600 seconds.

XGB Training Time with constant spacing length

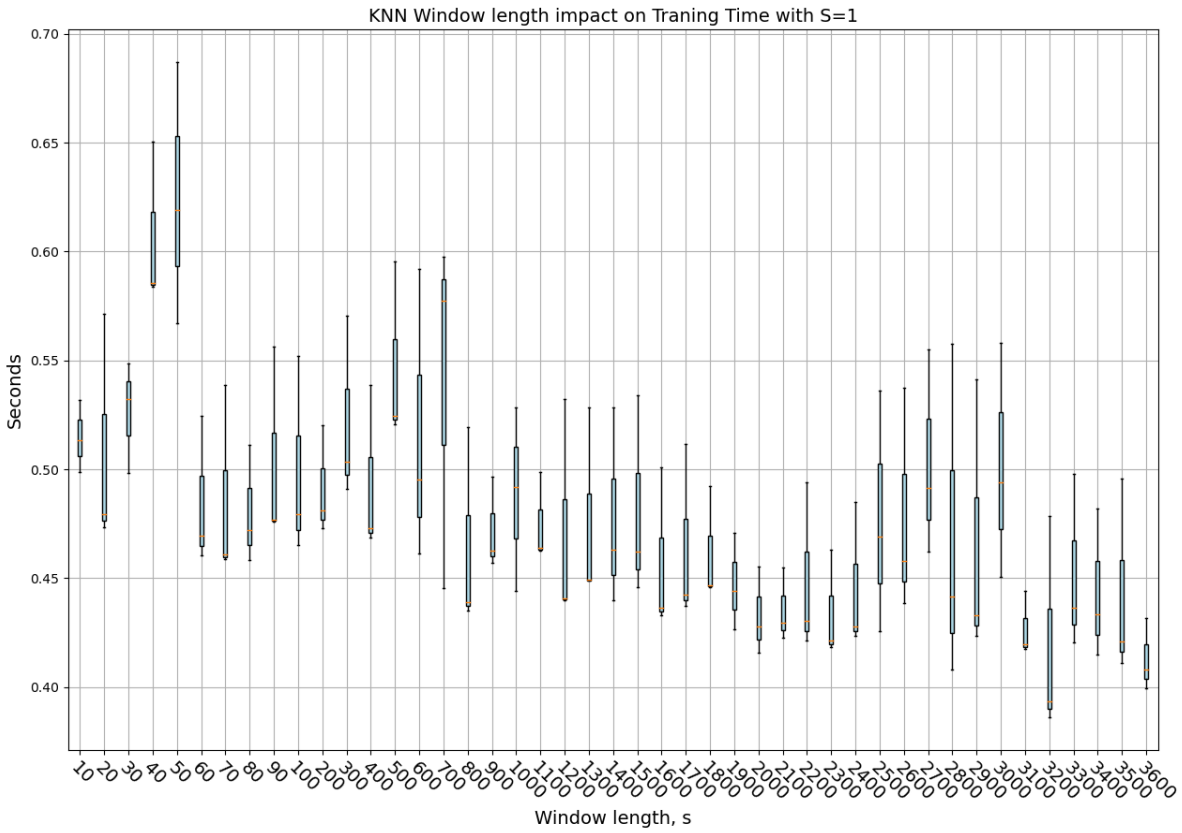


The first intuitive result comes from the training time. The longer the window is the less time it takes to train the model because a longer window duration means fewer windows.

MLP Training Time with constant spacing length



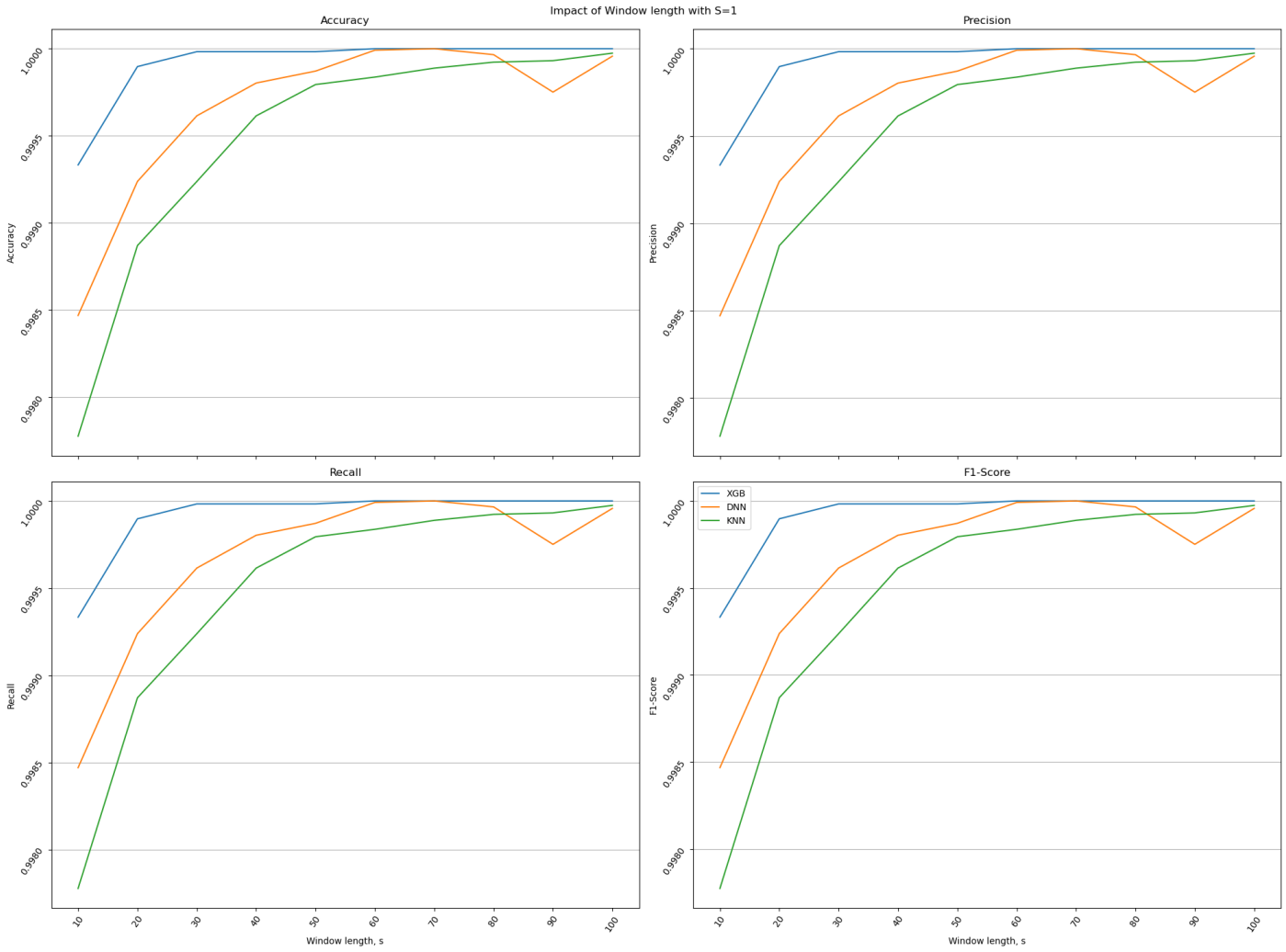
KNN Training Time with constant spacing length



KNN differs from the previous algorithms, it shows a pretty **constant training time** for each Window length. This is because the KNN complexity for Training is O(1) which means it is constant. [3]

Accuracy, Precision, Recall, and F-1 score with constant spacing length

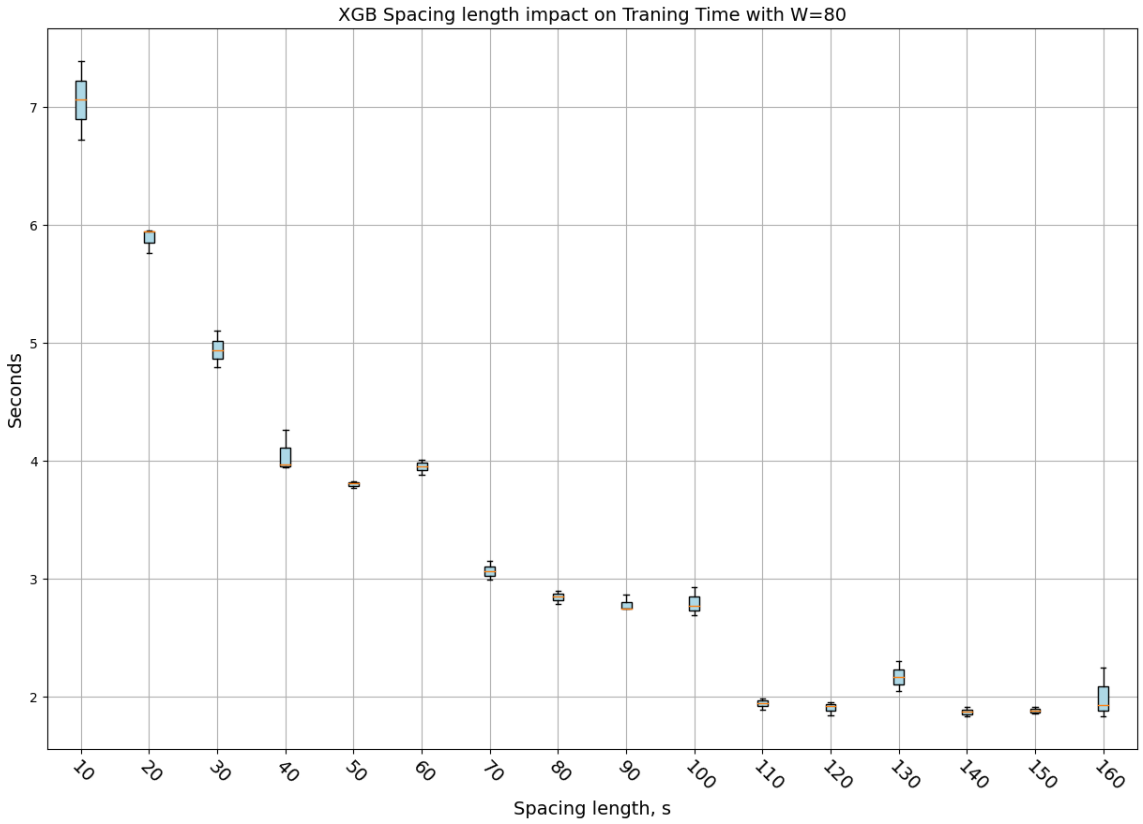
Changing the window duration also has an impact on the performance of the algorithm, as shown in the series of plots below. The larger the window size, the higher the metrics values.



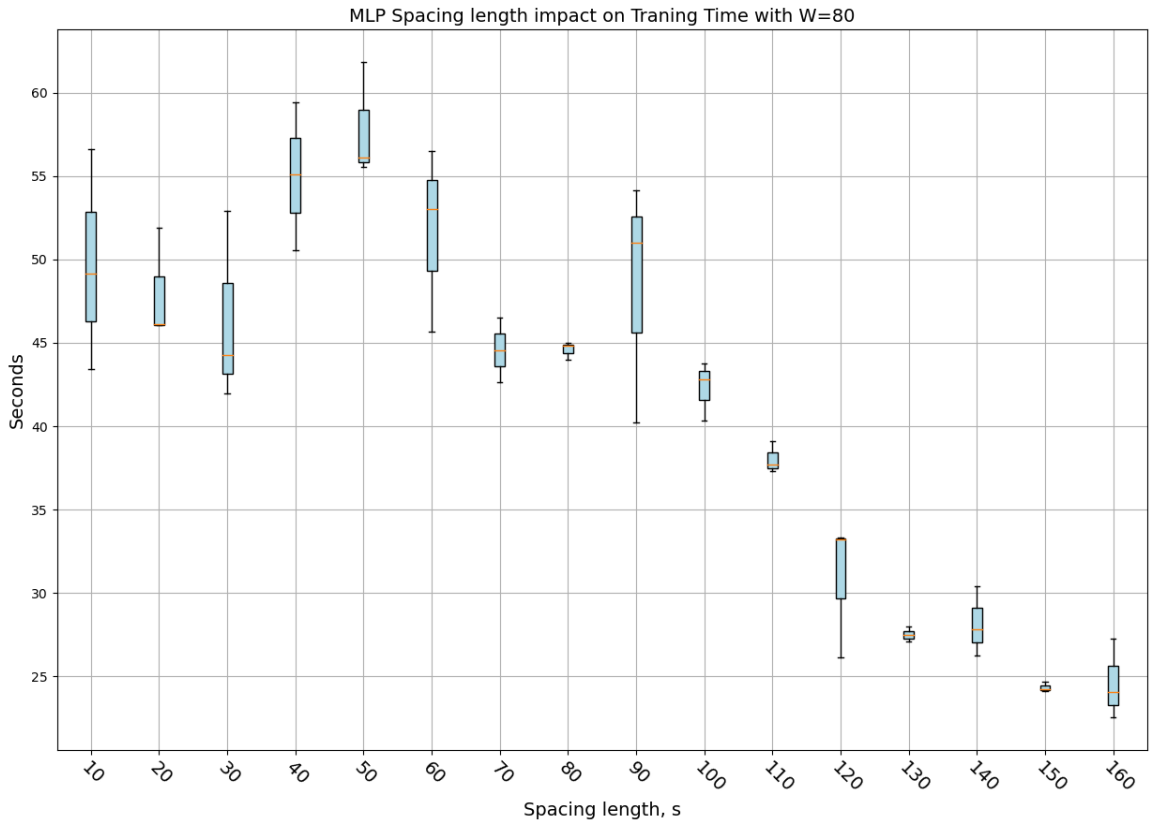
XGB Training Time with constant window length

As for the window duration, the **impact of spacing** is analyzed by fixing the window duration to 80.

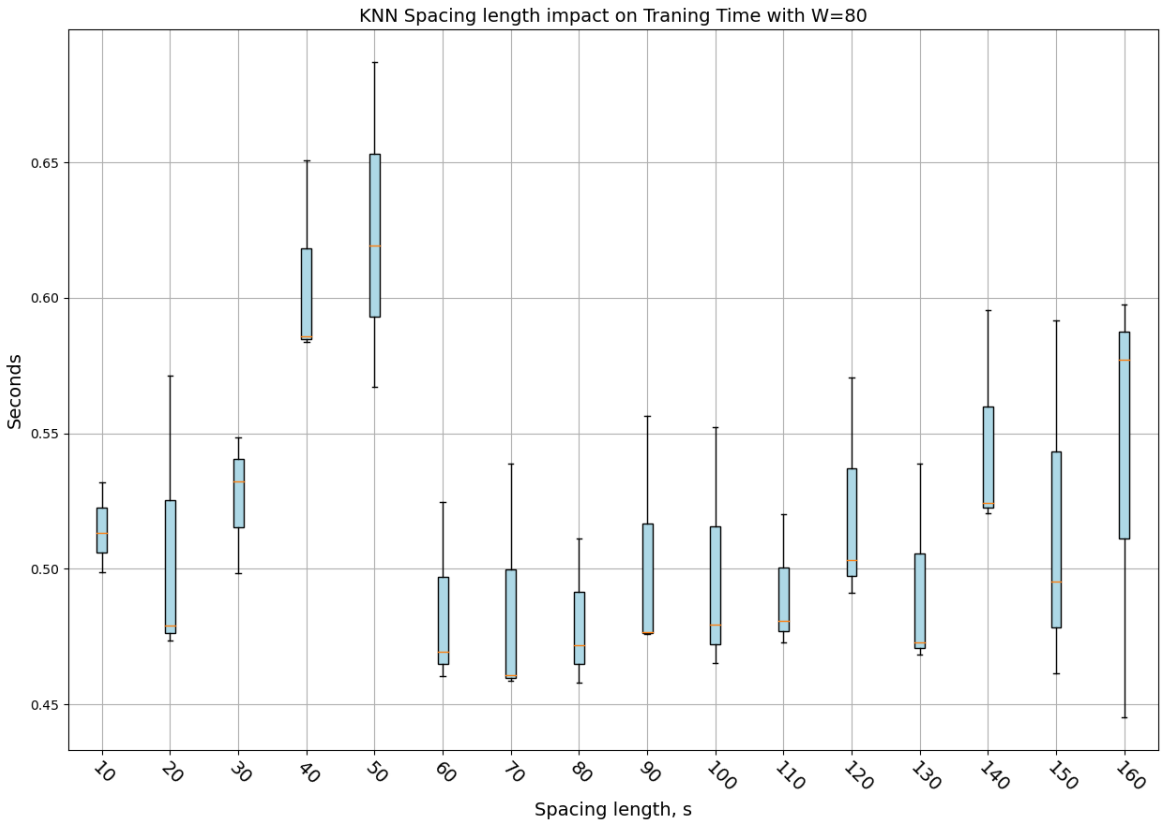
The training time has also in this case a decreasing trend as the spacing gets higher. This happens because more spacing means fewer windows analyzed.



MLP Training Time with constant window length

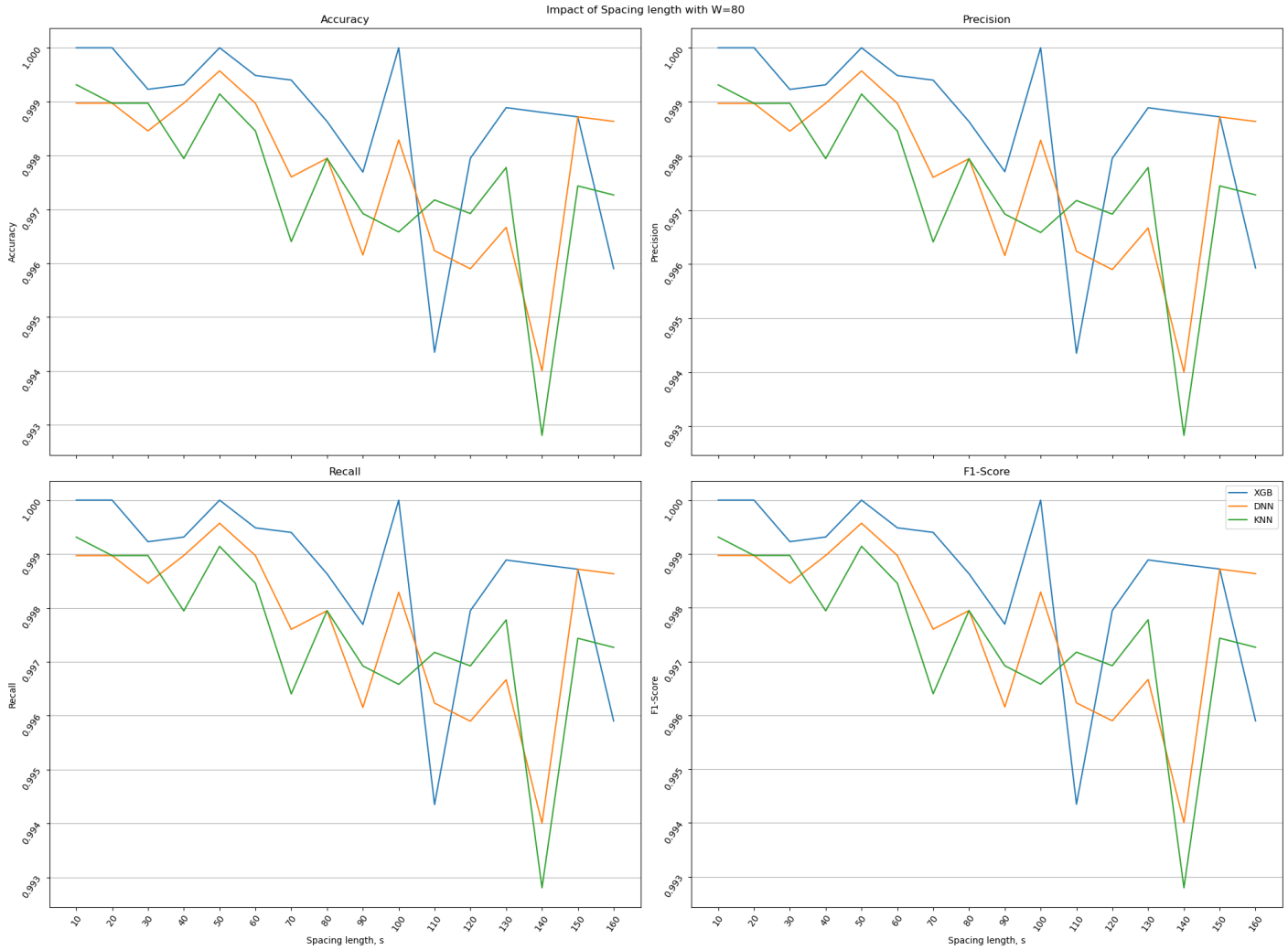


KNN Training Time with constant window length



As for the case with constant spacing length KNN differs from the previous algorithms, it shows a pretty **constant training time** of each Spacing length.

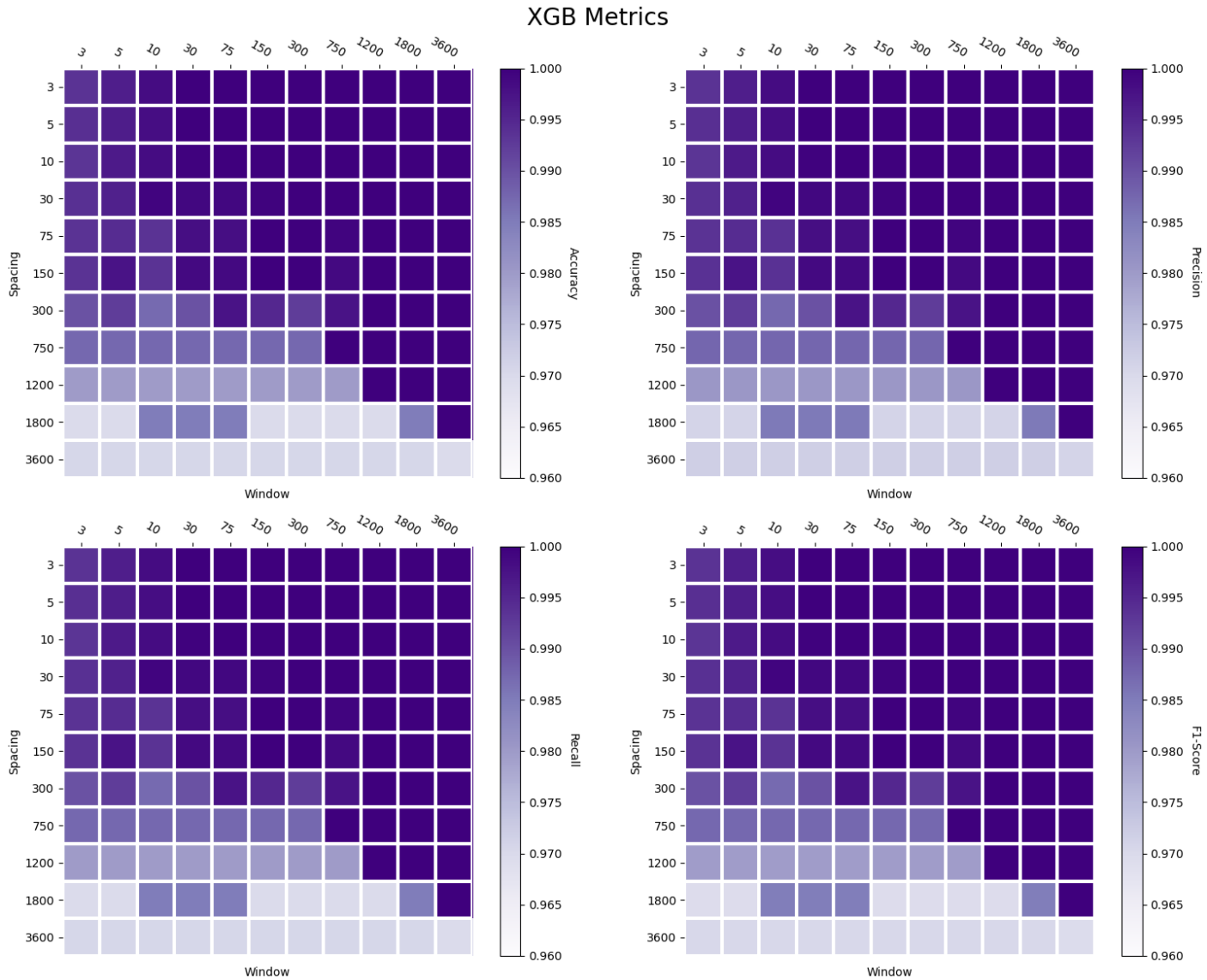
Accuracy, Precision, Recall, and F-1 score with constant window length



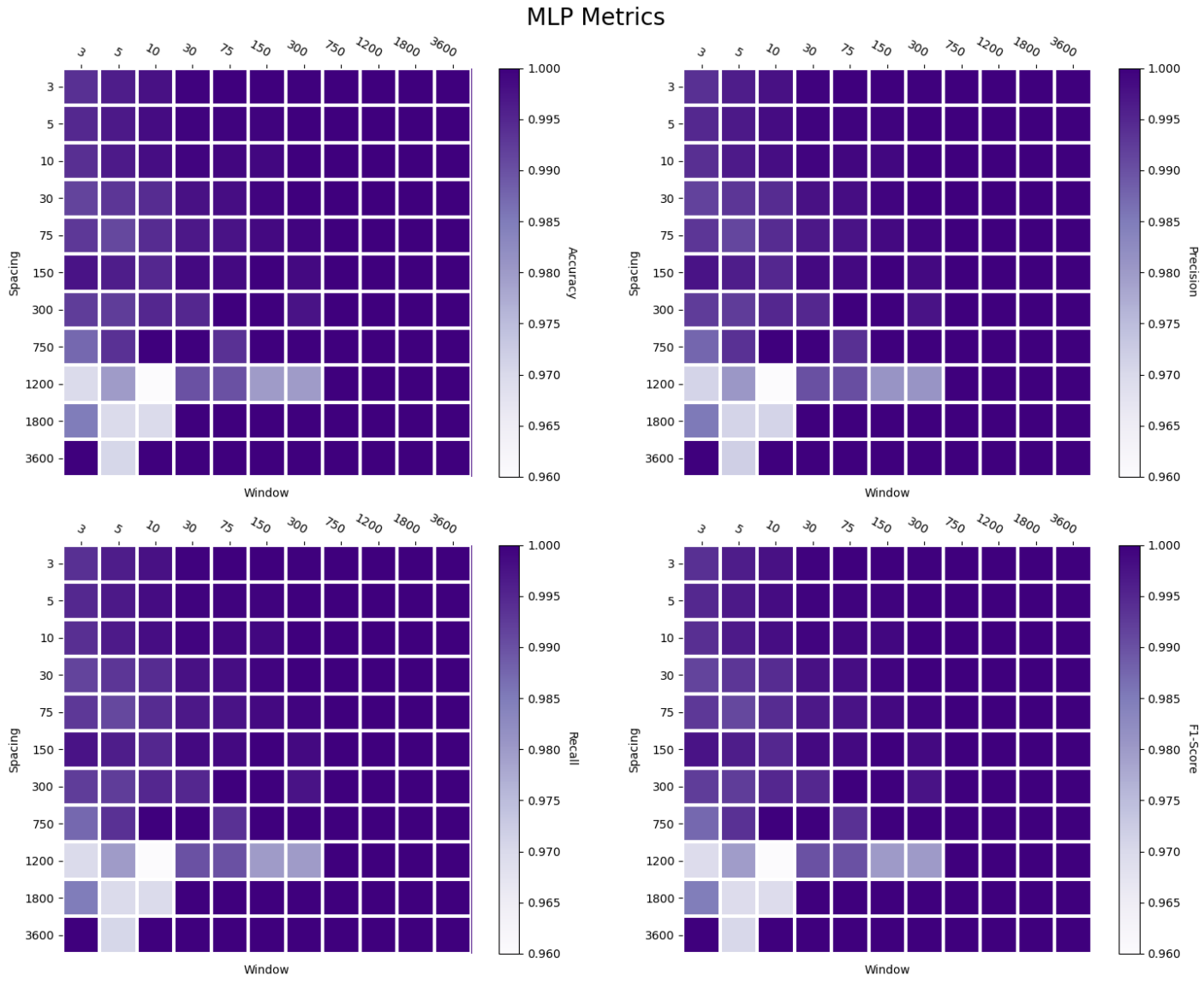
XGB Accuracy, Precision, Recall, and F-1 score - Heatmap

An interesting outcome of this point is the best tuning of window spacing/duration parameters. The optimal combination of the two is indeed a compelling topic since it allows us to pick the less resource-wasting pair of the two parameters so that high performances can still be achieved.

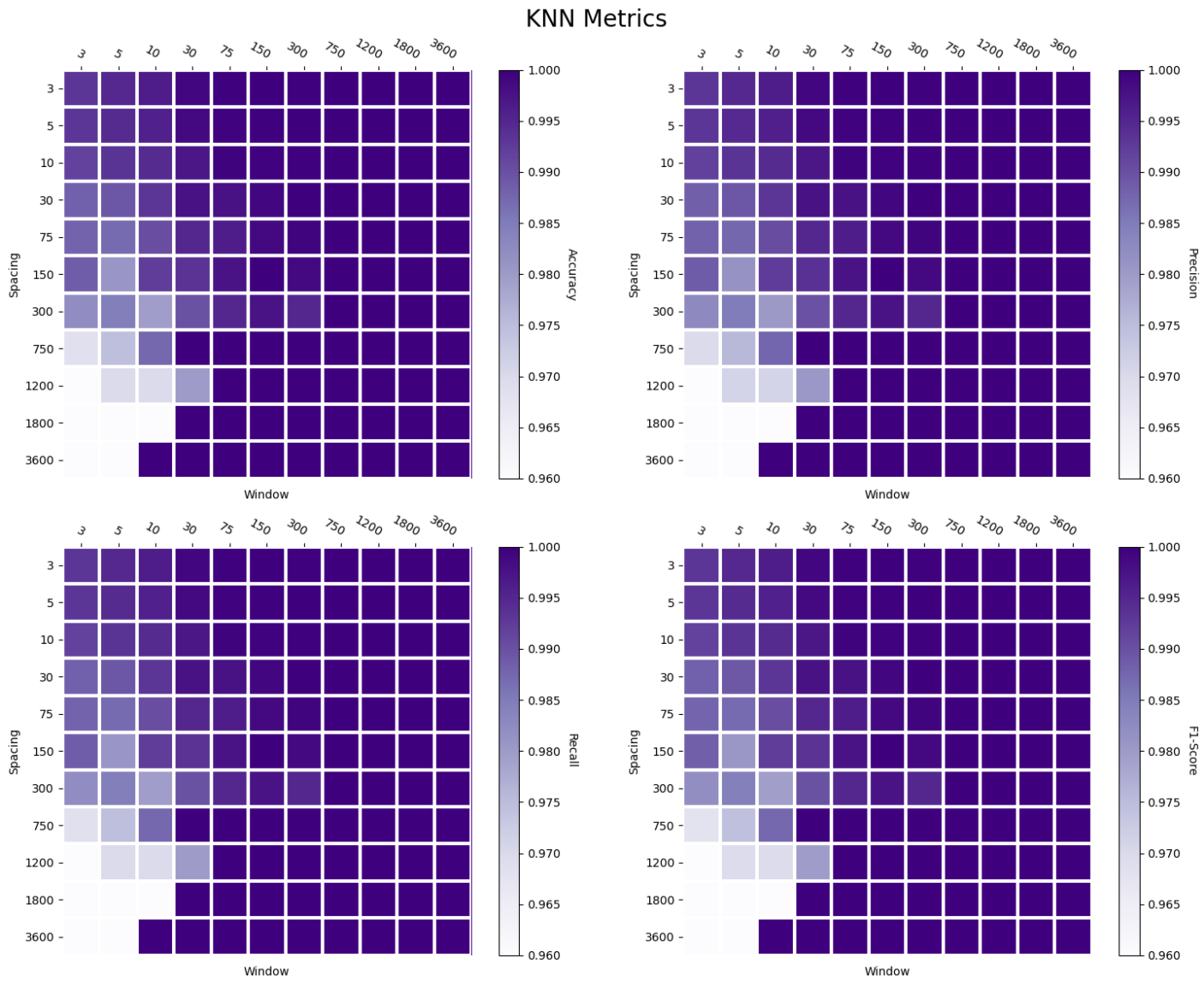
The correlation between metrics performances and the values of these two parameters has been plotted using a heatmap, as shown in the graphs below.



MLP Accuracy, Precision, Recall, and F-1 score - Heatmap



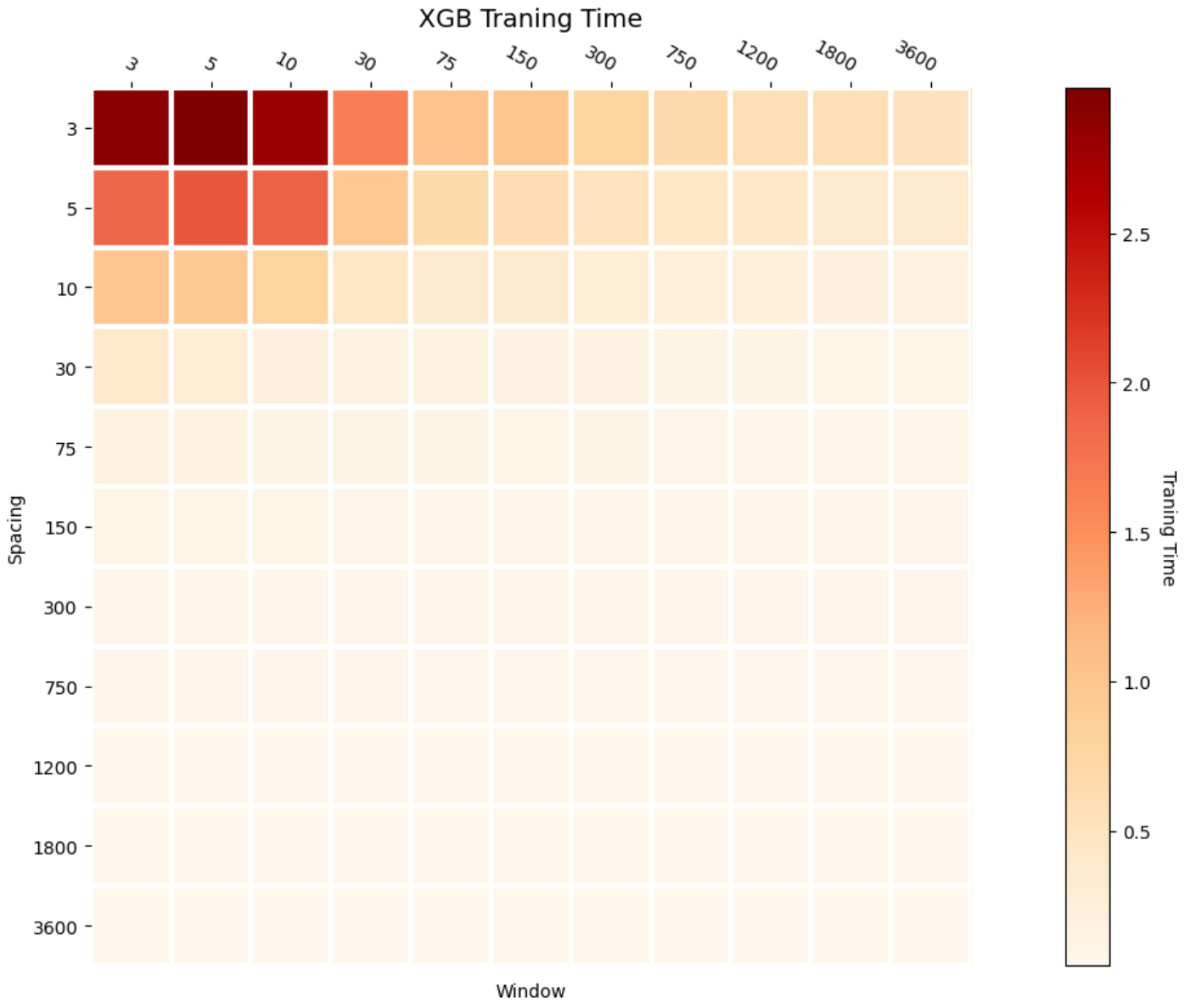
KNN Accuracy, Precision, Recall, and F-1 score - Heatmap



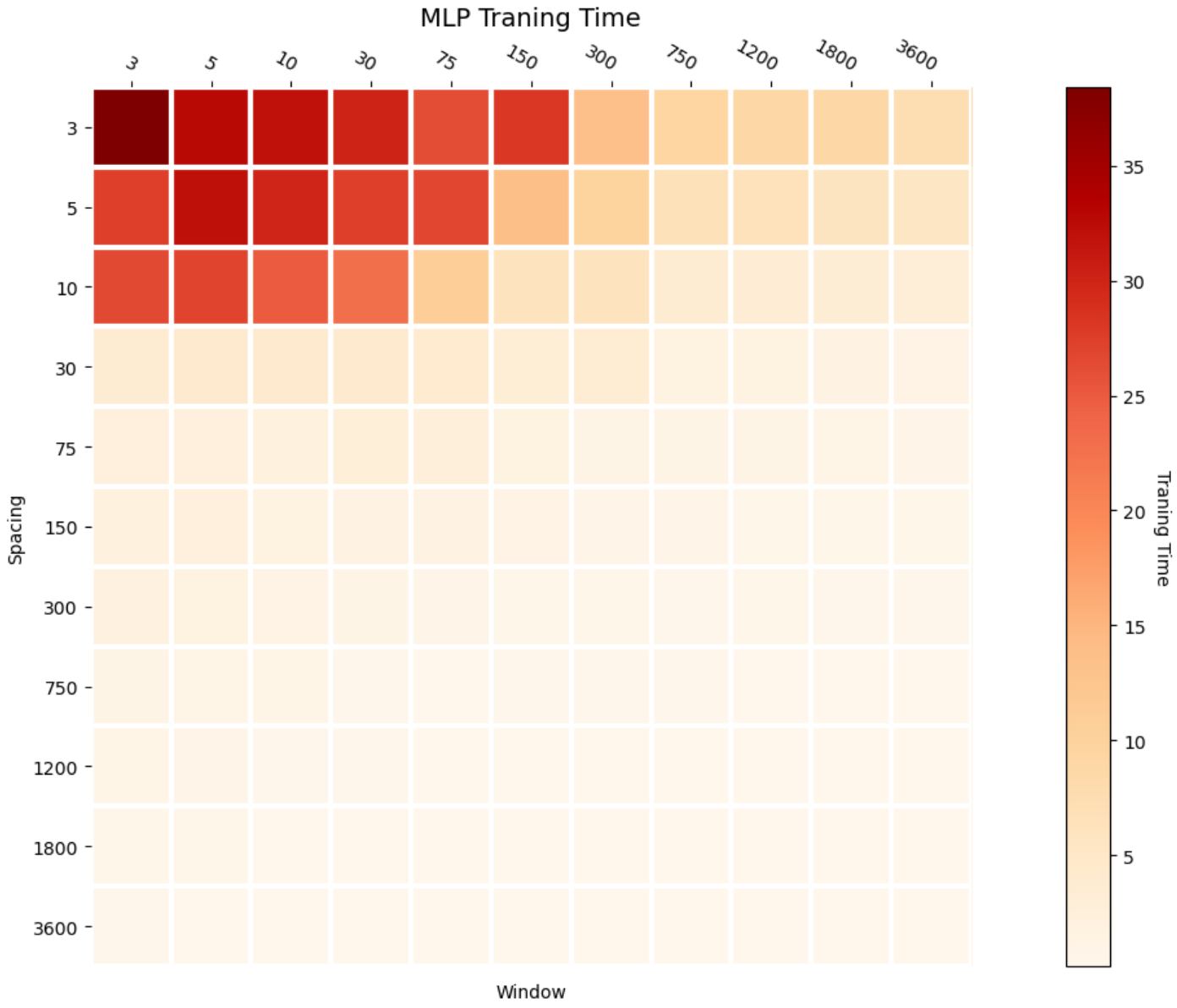
Crossing the performance and the training time plots it can be stated that an optimal combination of parameters is large window duration and spacing (both 3600) for MLP and KNN models and window duration worth 3600 and spacing equal to 1800 for the XGB model.

Thanks to these parameters tuning the ideal combination that ensures both memory and time optimization and high performances can be carried out.

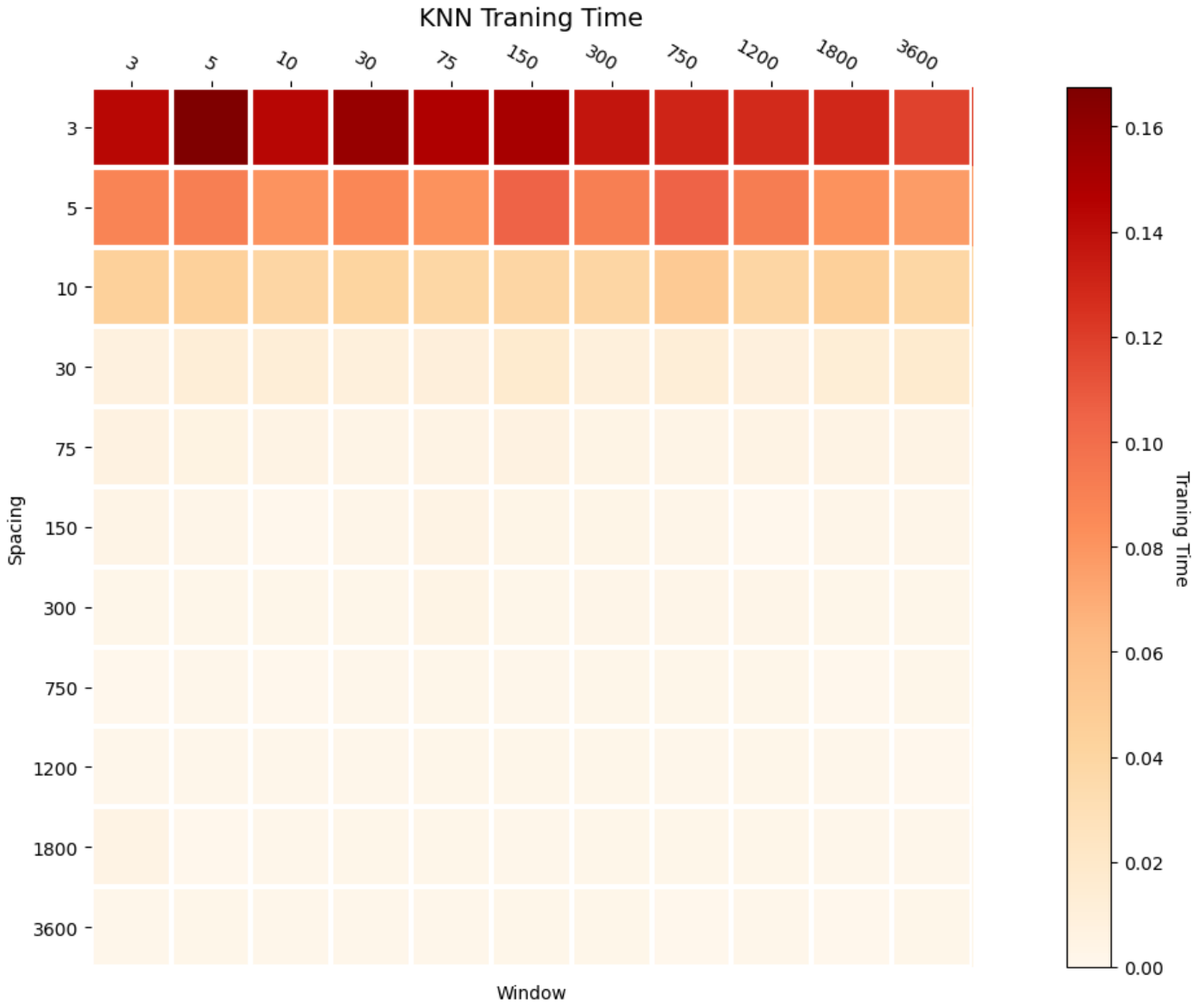
XGB Training Time - Heatmap



MLP Training Time - Heatmap



KNN Training Time - Heatmap



From the above graphs it is clear how we obtain short training time for large Spacing and Window sizes.

1.7 - What is the impact of a shorter sampling period TOSNR? For a given window duration (in seconds) can we have similar performance with fewer OSNR samples? Should we compensate with longer windows?

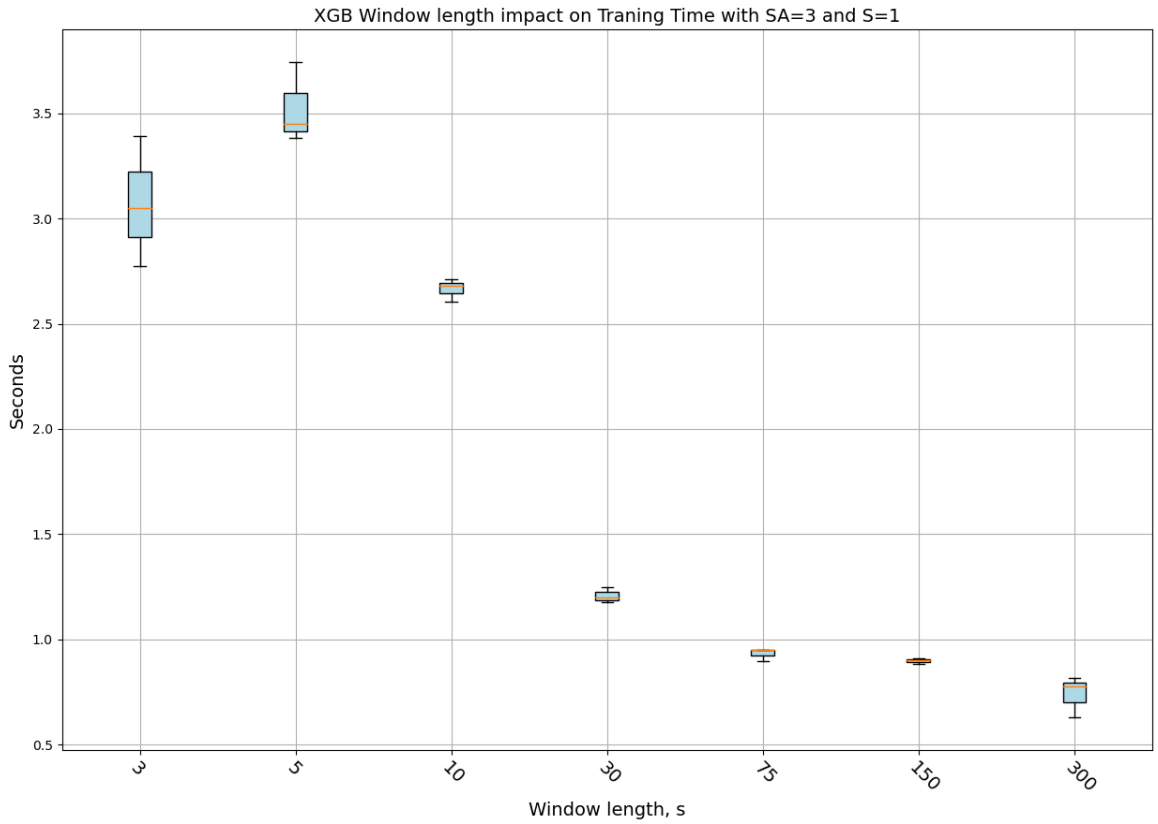
The case of having fewer OSNR samples has also been considered.

This hypothetical scenario was built up by extracting 1 sample from every x sample from the raw dataset. In this way, the size of the newly built dataset is 1/x the original dataset size.

As for the previous points, the impact on training time and metrics performance have been carried out.

Training Time Comparison

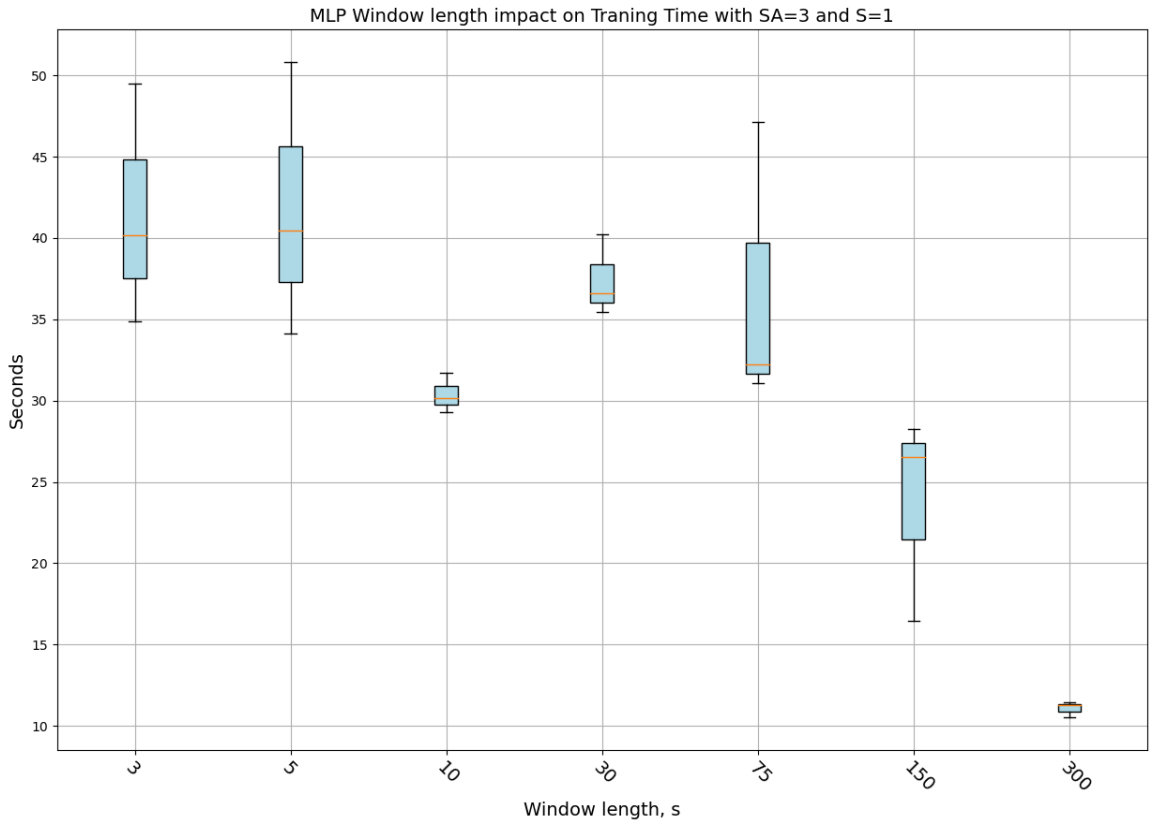
XGB Training Time Comparison



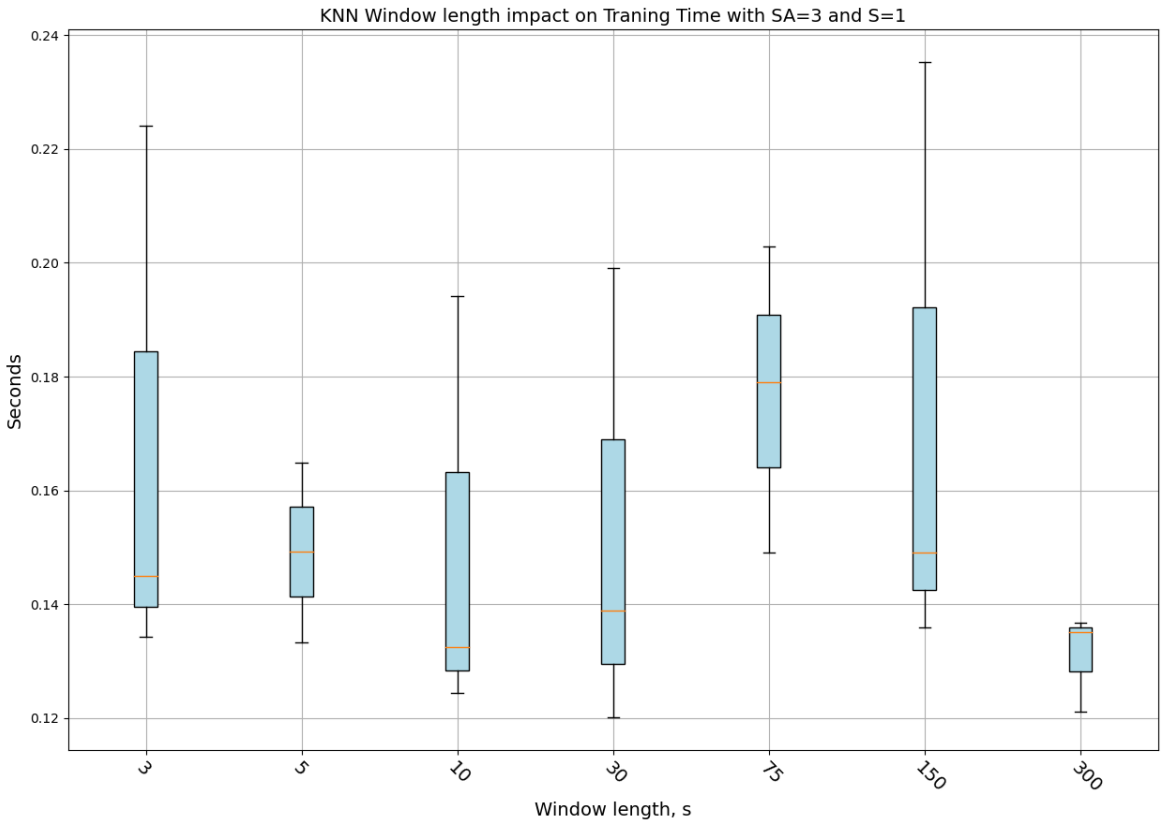
The graph above shows how in general the training time decreases as the window length grows. This result was already obtained before and is coherent with the previous point.

The training time is decreasing as the window spacing grows

MLP Training Time Comparison

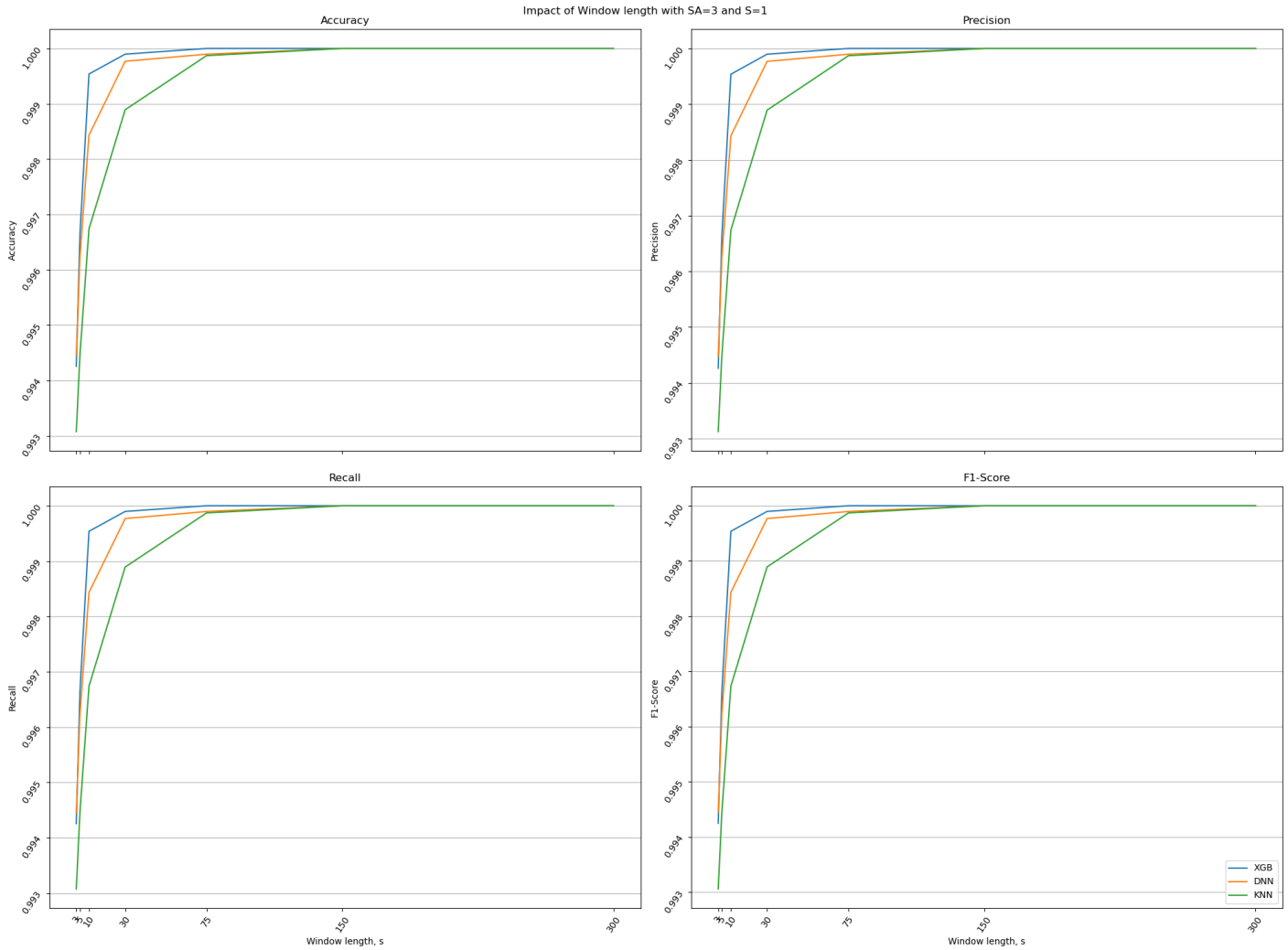


KNN Training Time Comparison



Impact of different window lengths on metrics with fixed sampling interval and spacing

In this case all the metrics: Accuracy, Precision, Recall, and F-1 score are growing as the window length grows. The plots below show this trend.

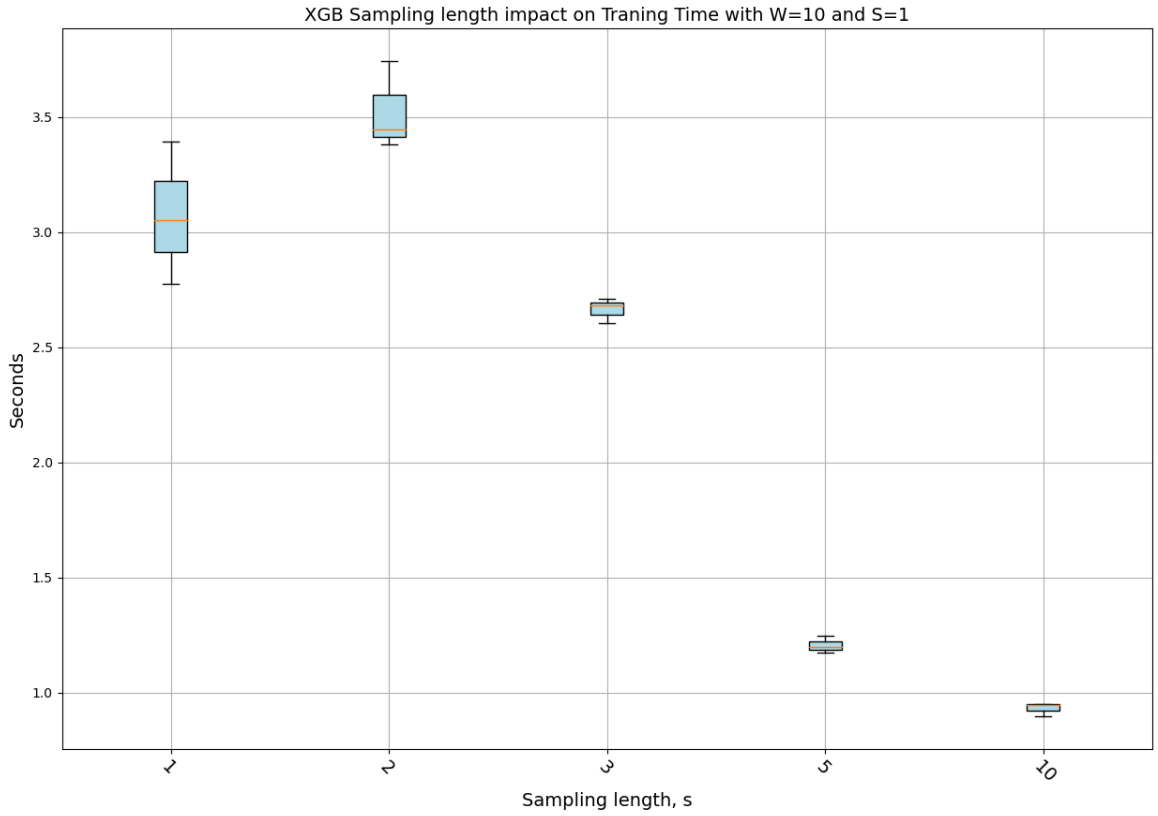


Training time with fixed window length and spacing

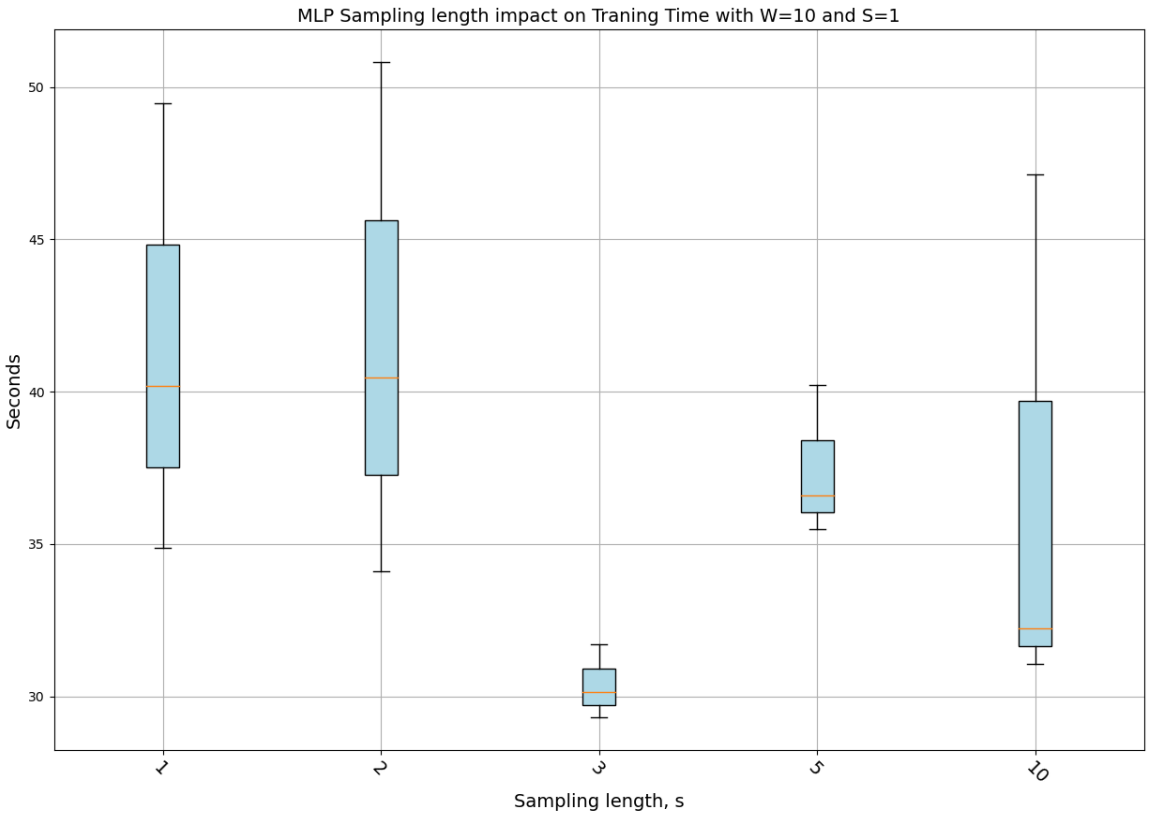
For the following analysis the window length and the spacing have been fixed.

The plots below show the impact of a decreasing sampling interval on the training time. As for the previous analysis, reducing the number of observations has a positive impact on the training time. Thus the training time decreases as the sampling range increases.

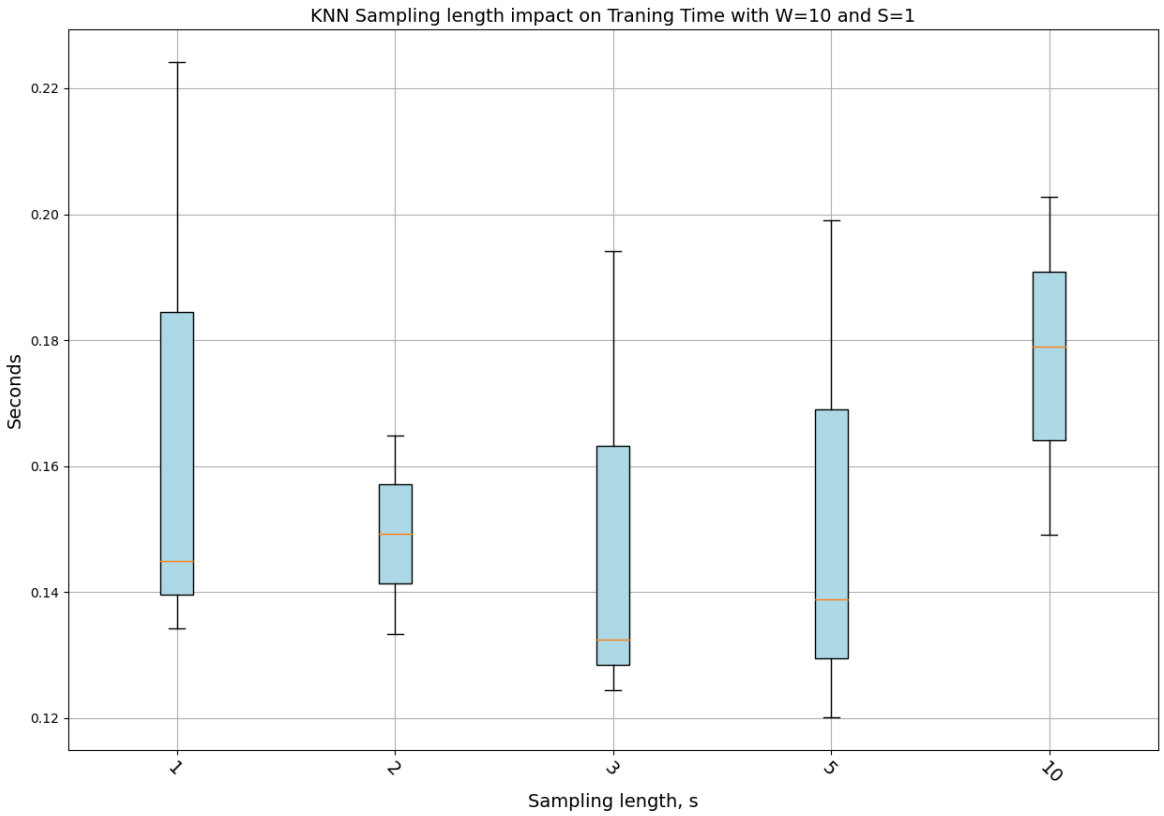
XGB Training Time Comparison



MLP Training Time Comparison



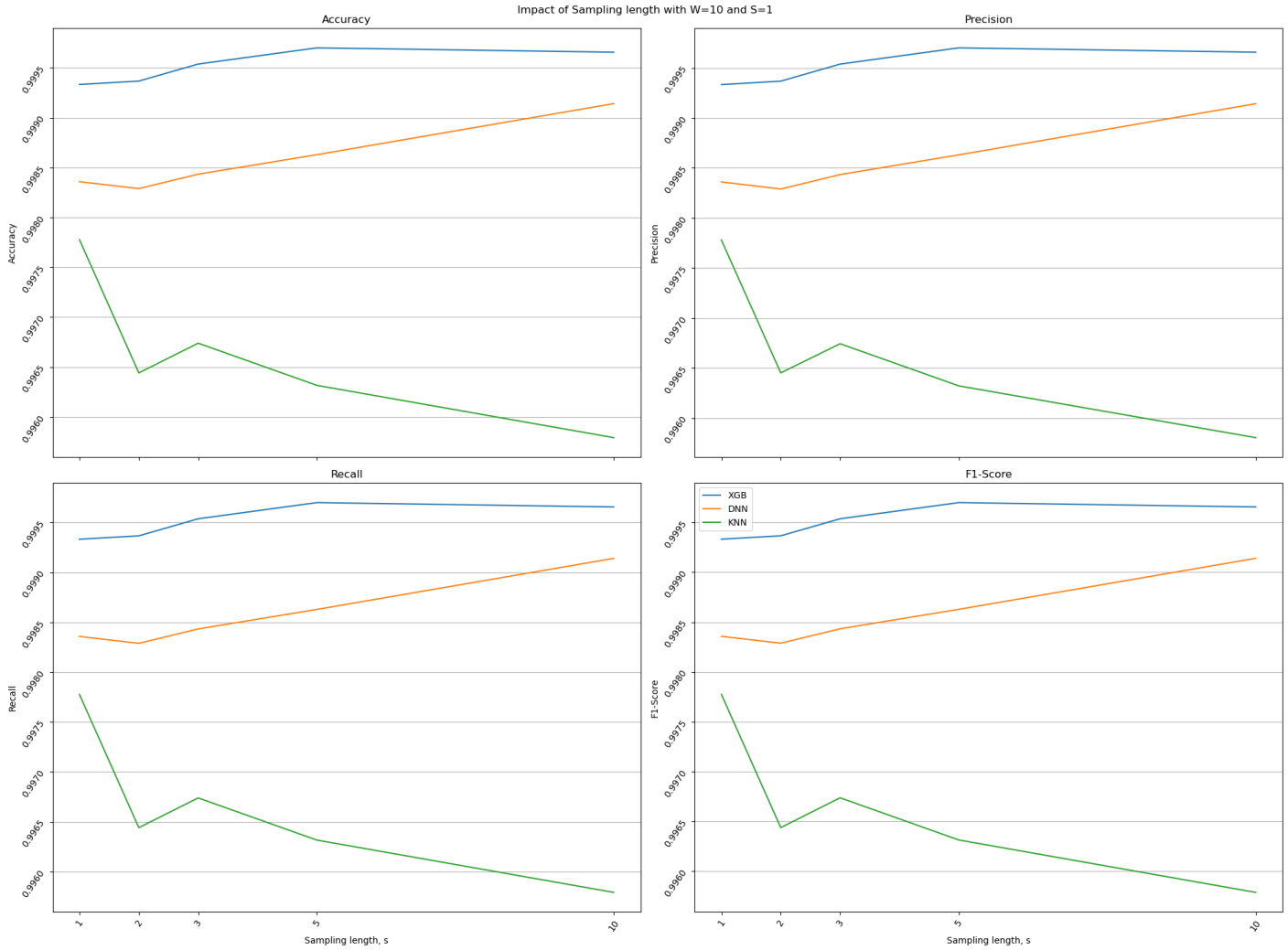
KNN Training Time Comparison



The KNN algorithm showed the expected behavior with a constant training time because of the constant complexity of the training phase.

Accuracy, Precision, Recall, and F-1 score

The results of the other metrics performances are shown in the graphs below. As for the previous points, even with heavy dataset degradation as one-tenth of the raw dataset, all the metrics stay above 99%.

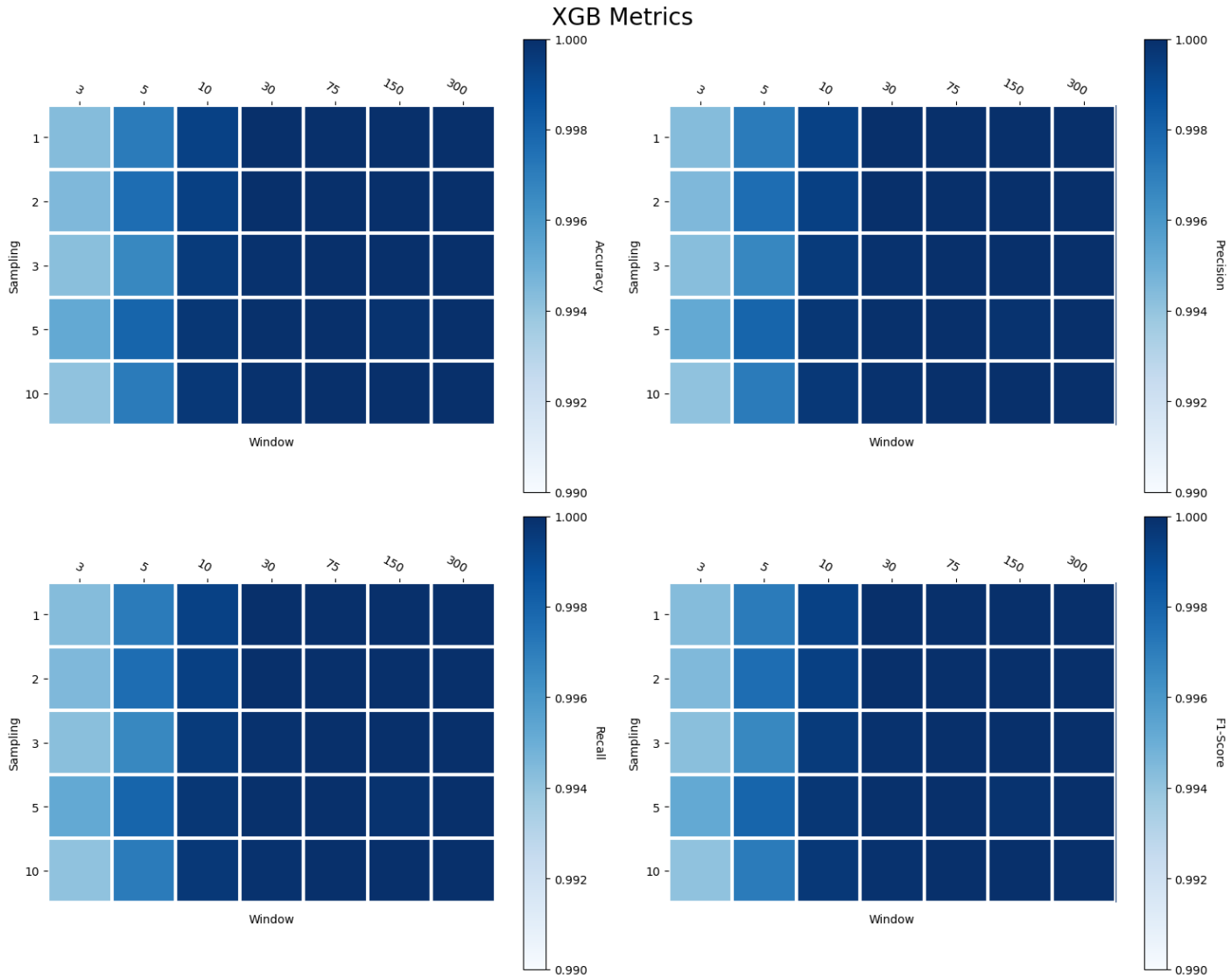


XGB Accuracy, Precision, Recall, and F-1 score

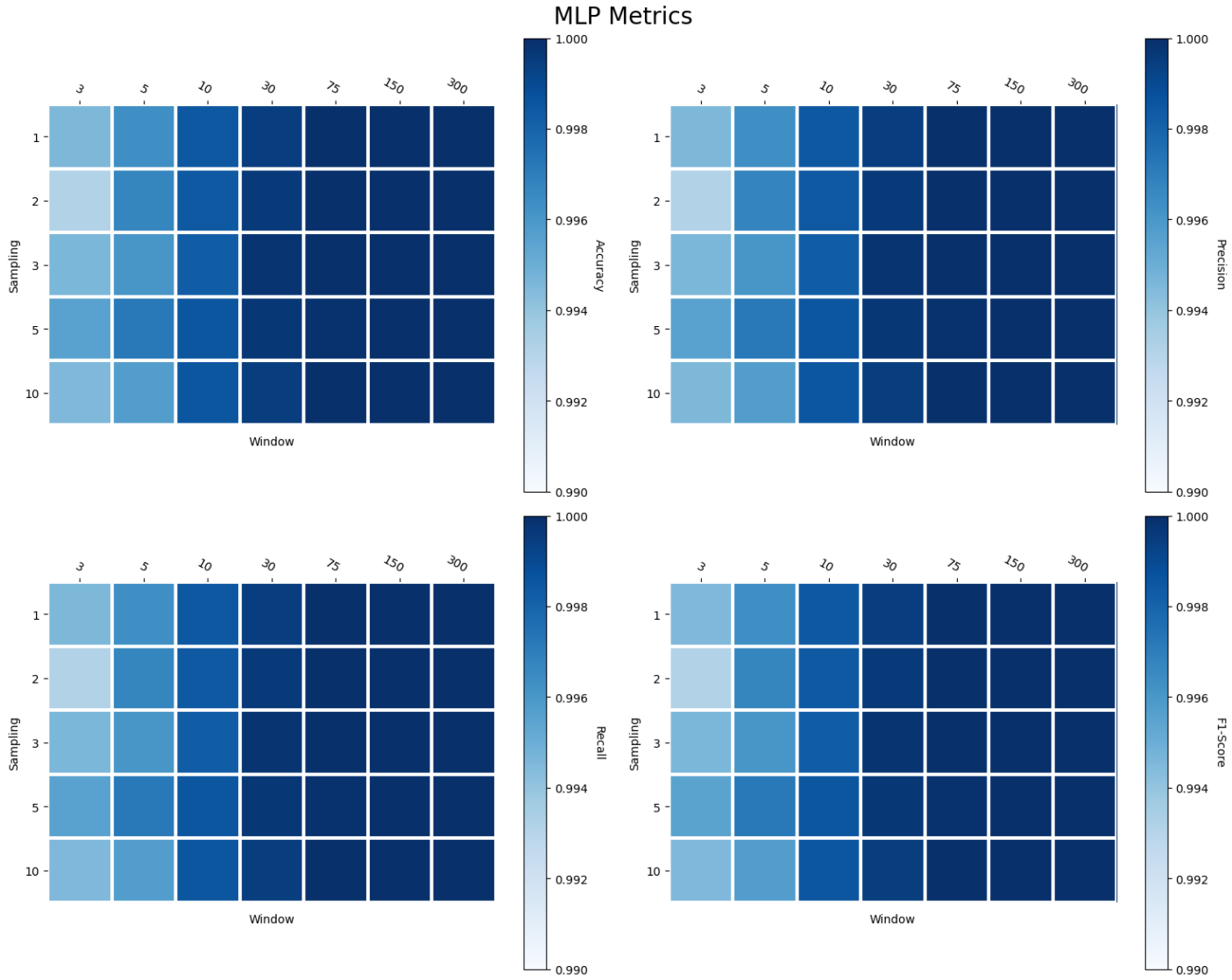
A similar approach to the previous point was held to find an appropriate trade-off between **sampling** and **window length**, for the sake of resource optimization.

The heat maps below show the influence of these parameters on the metrics and on the training time.

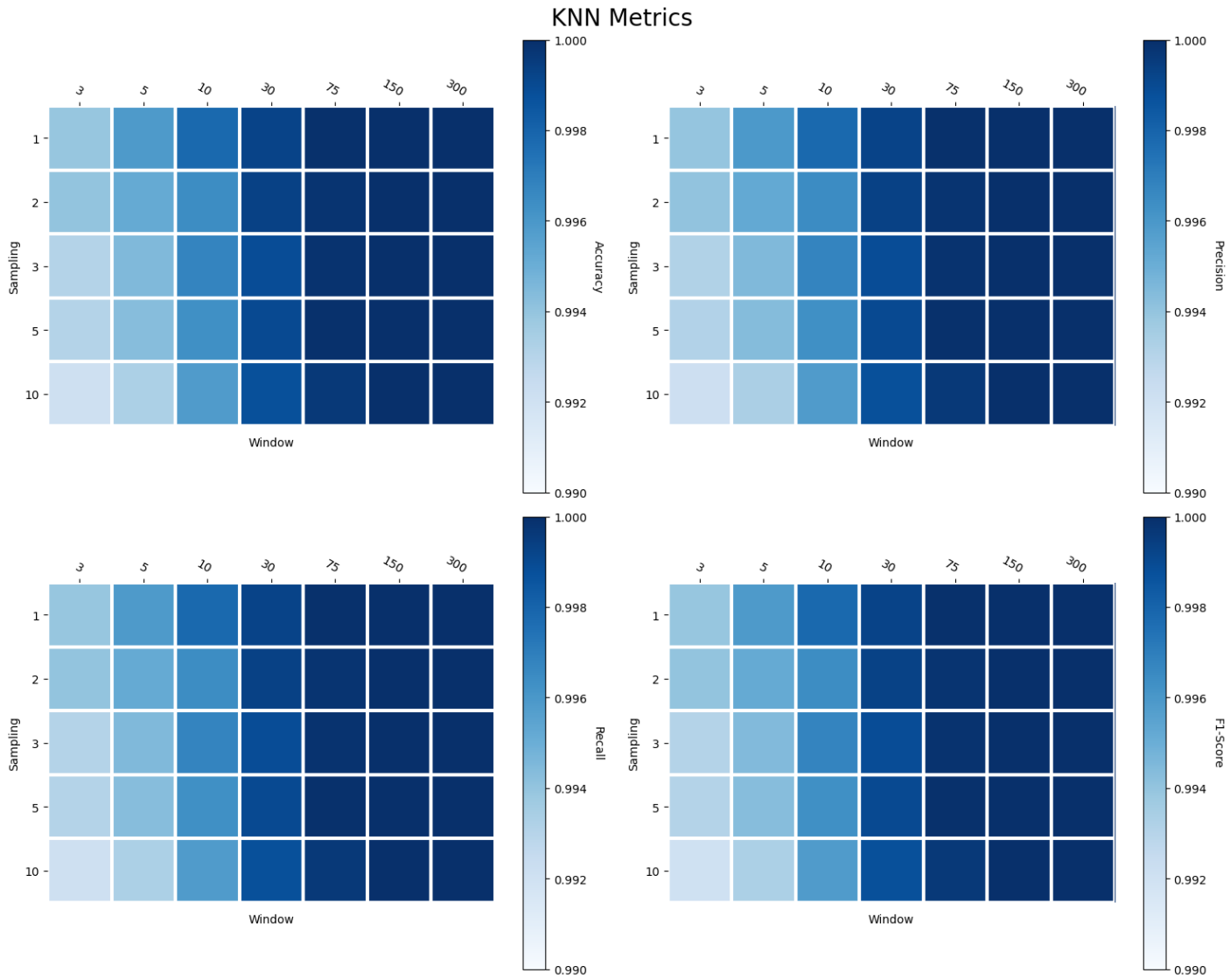
Cross-checking these metrics-related plots is quite intuitive to find an optimal solution.



MLP Accuracy, Precision, Recall, and F-1 score



KNN Accuracy, Precision, Recall, and F-1 score

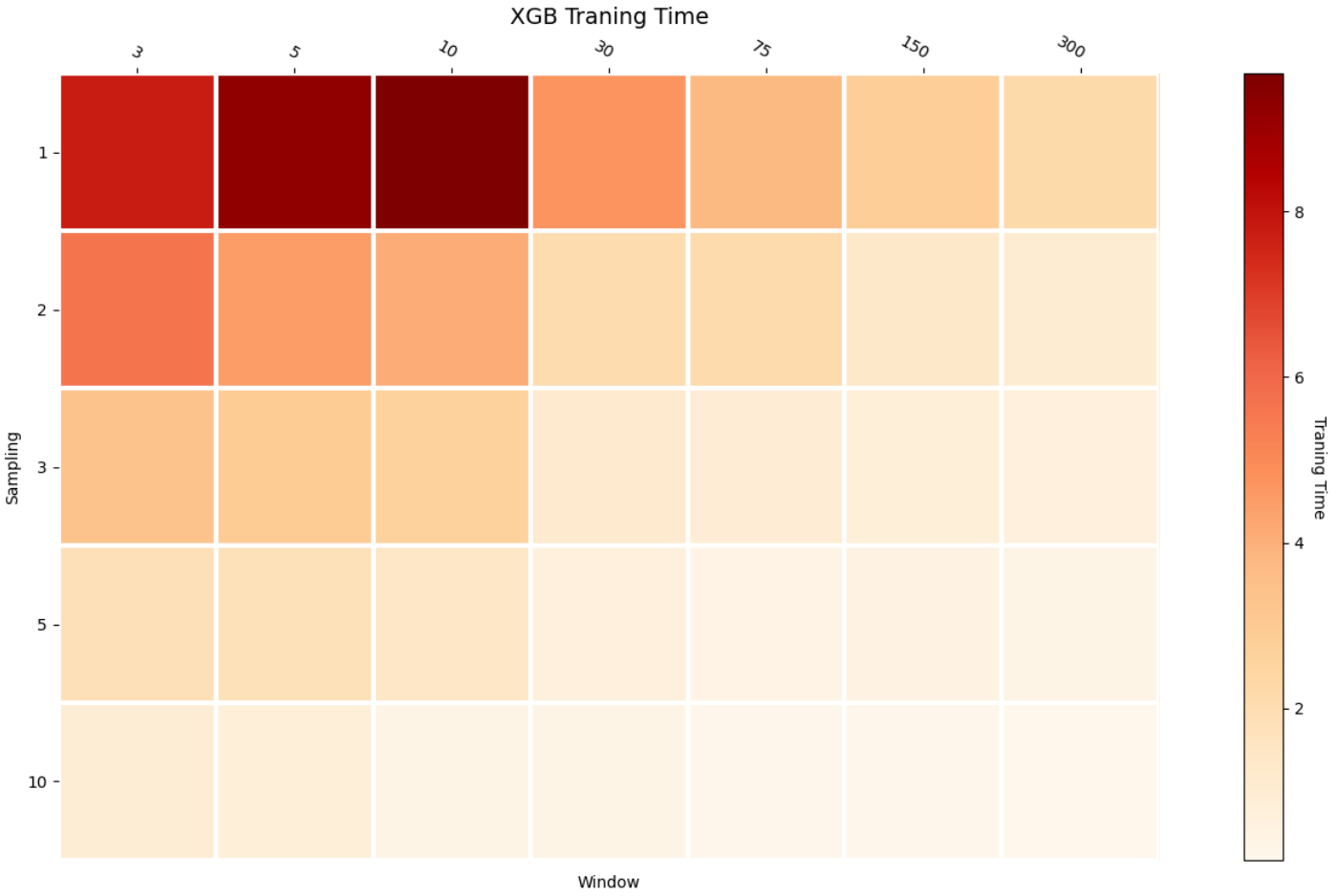


Regardless of the model used and the sampling period nearly the highest performances, in this graph, can be achieved with a window length of 300.

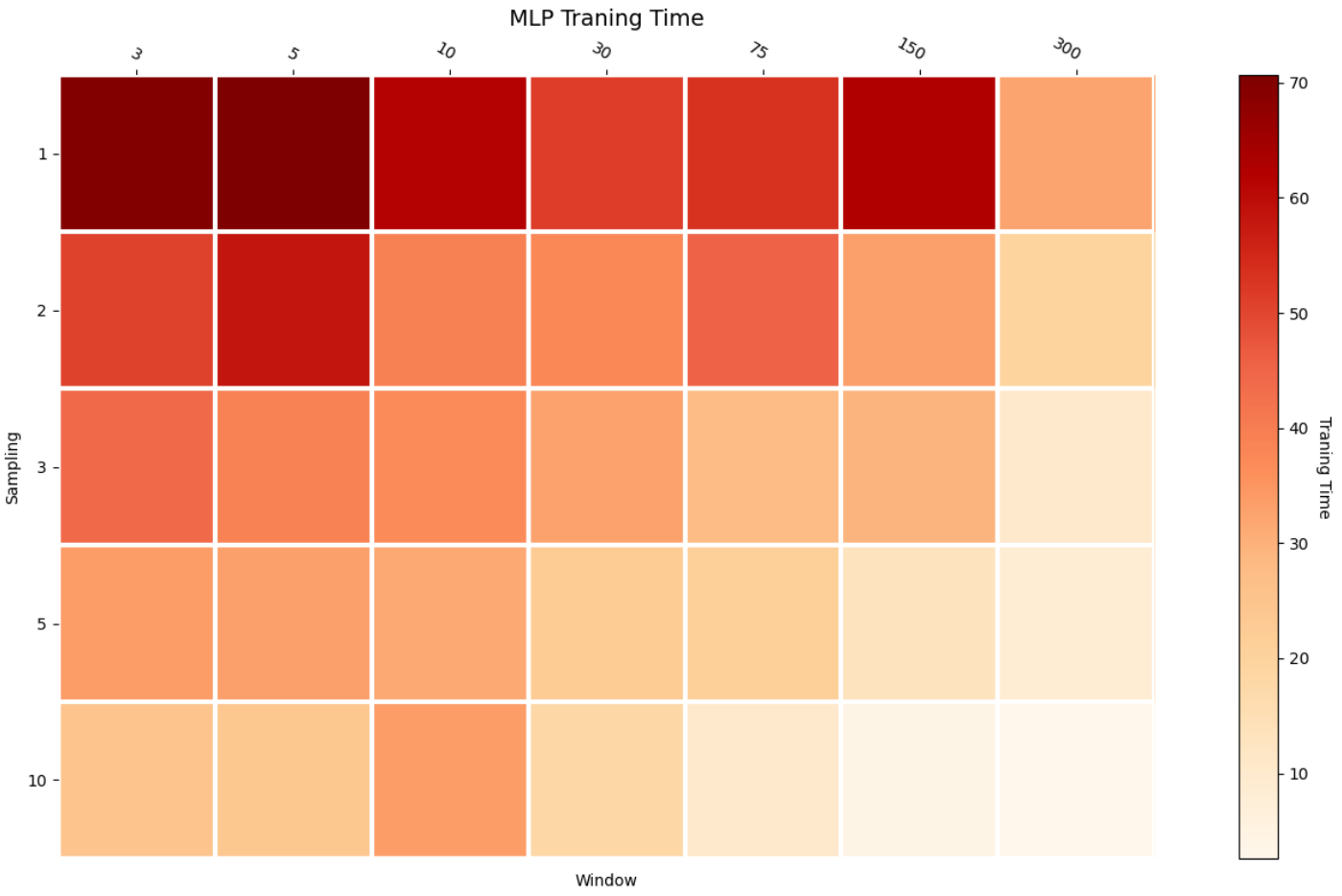
However, higher performances can be obtained by picking a higher sampling period, as the plots below show. Thus a good combination of these two parameters would be W = 300, S = 10.

In this way, a both memory and time-optimized model is created, which also achieves high performances.

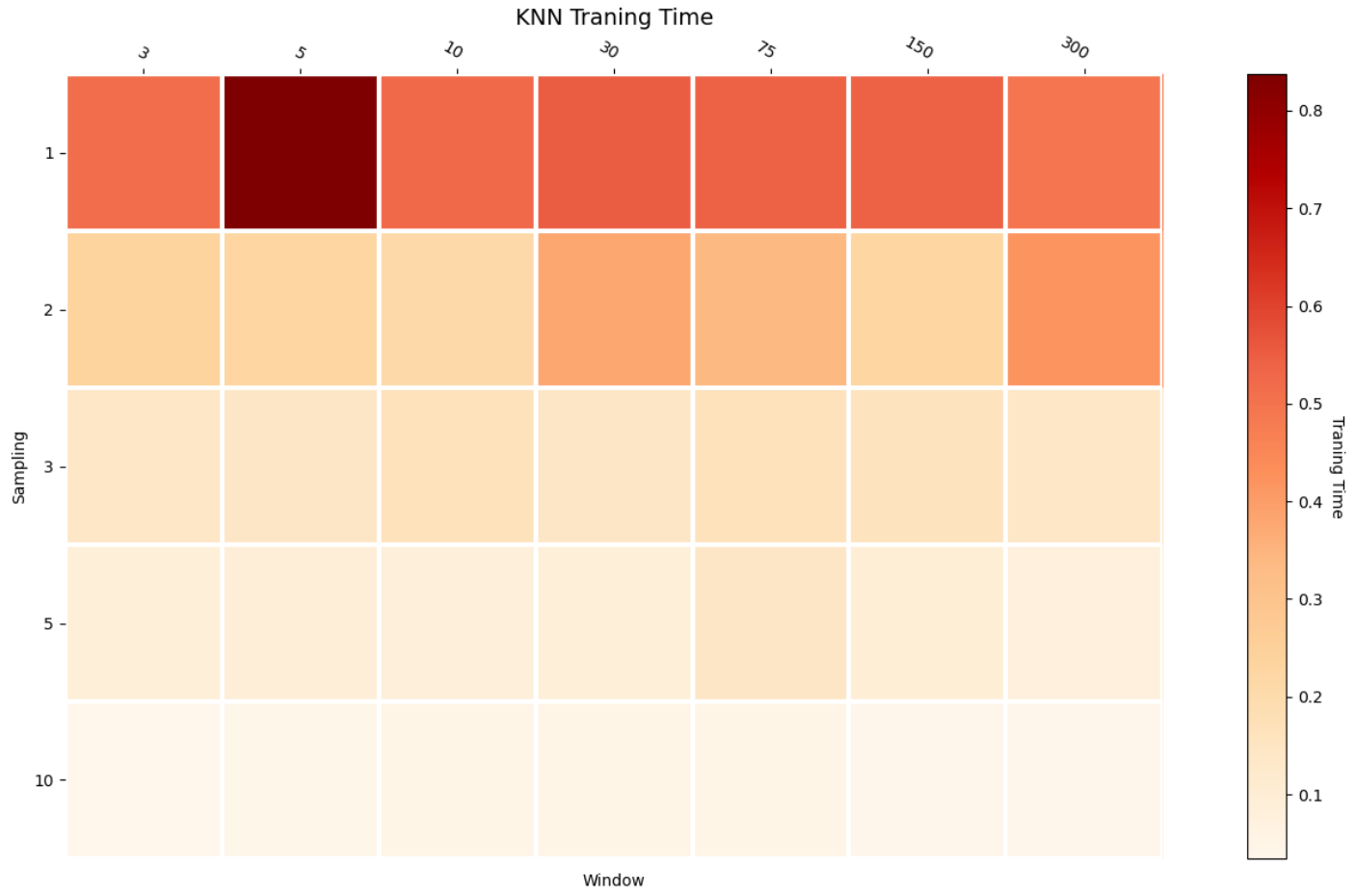
XGB Training Time



MLP Training Time



KNN Training Time



As expected from the above graphs it is clear how we obtain short training time for large Sampling periods and large Window sizes.

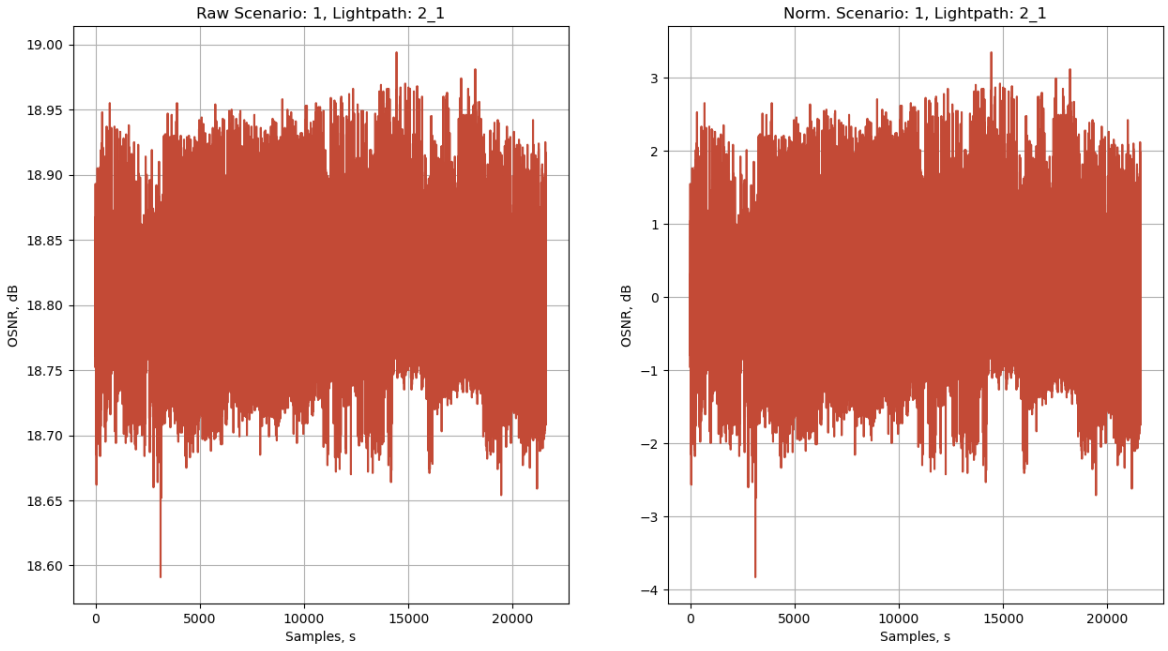
1.8 - What is the impact of OSNR normalization?

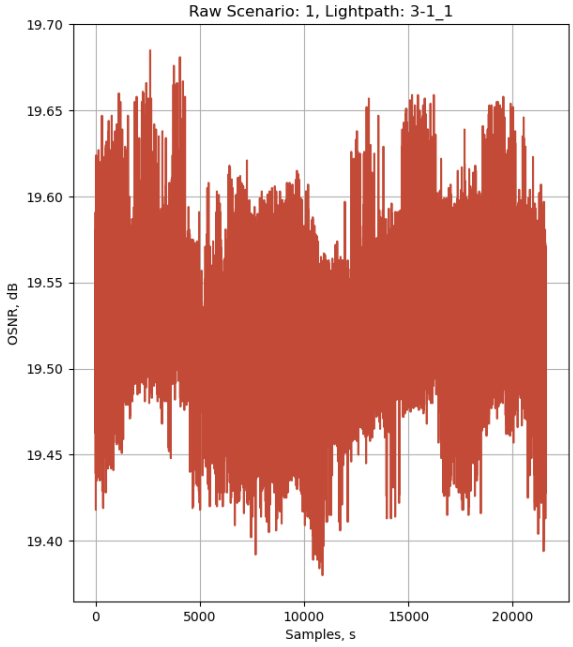
The case in which the OSNR values are normalized before extracting the features has also been taken into account.

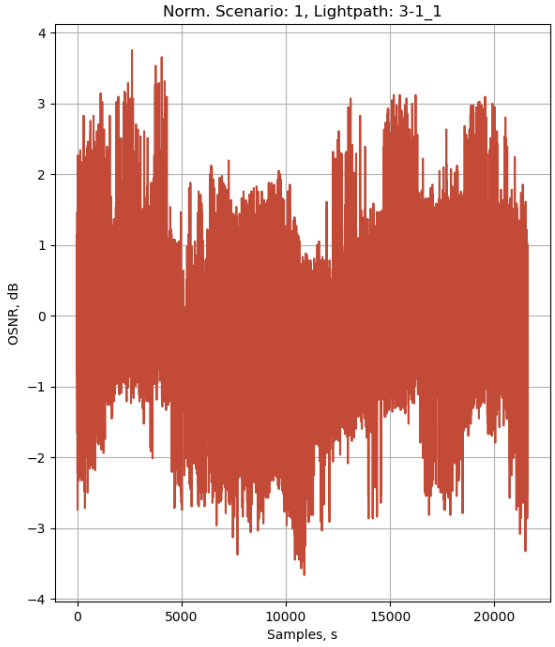
In order to perform this analysis a new dataset has been created and the raw values of the OSNR have been normalized.

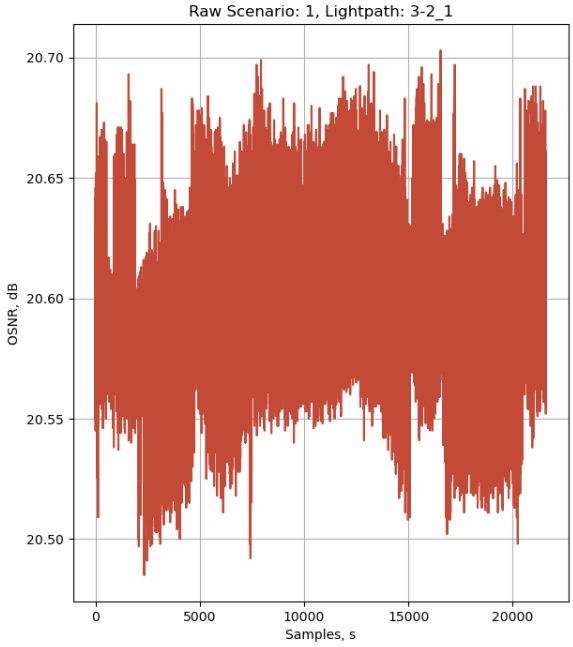
The impact of this action on the metrics and training time has then been computed.

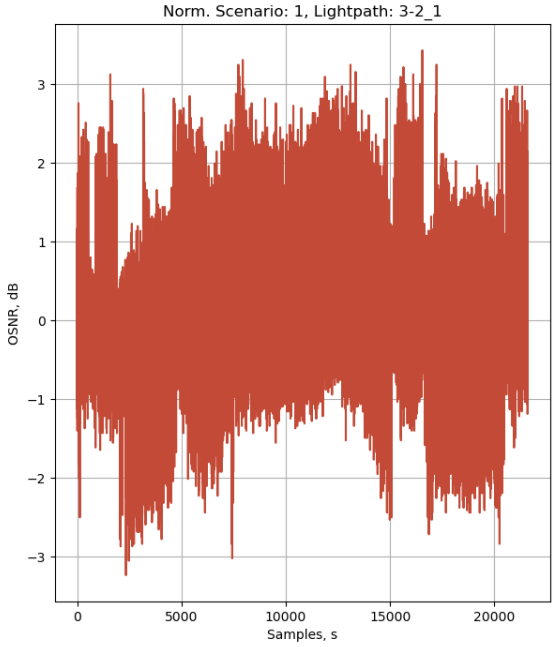
Scenarios with Attenuation Failure

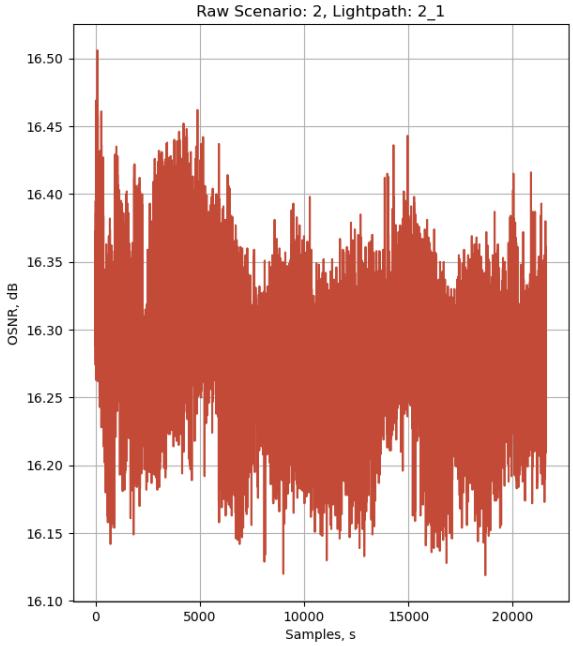


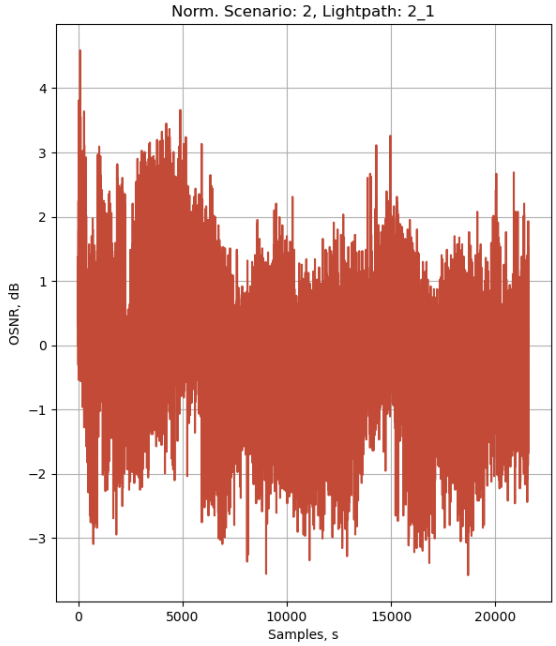

Raw Scenario: 1, Lightpath: 3-1_1
OSNR, dB
Samples, s

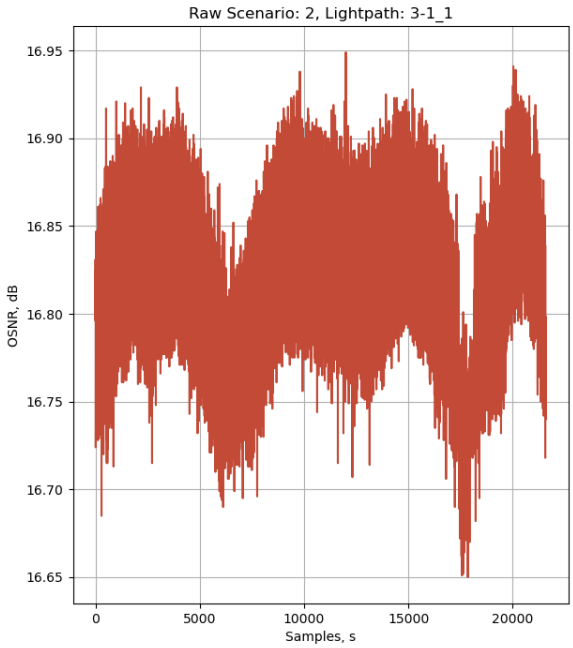

Norm. Scenario: 1, Lightpath: 3-1_1
OSNR, dB
Samples, s

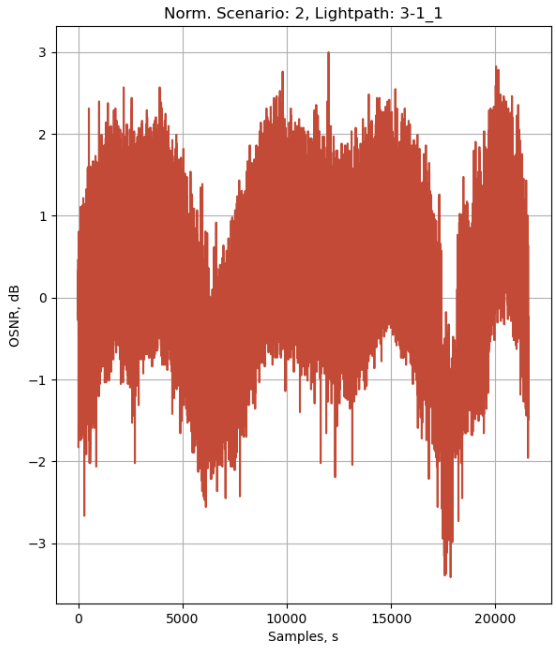

Raw Scenario: 1, Lightpath: 3-2_1
OSNR, dB
Samples, s

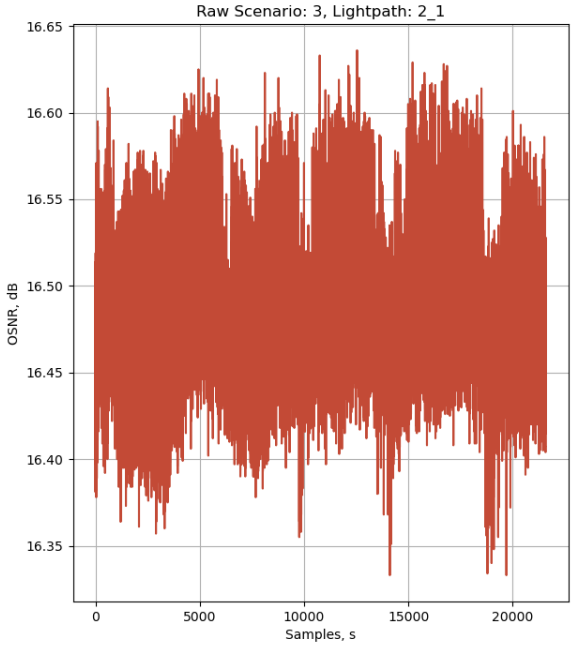

Norm. Scenario: 1, Lightpath: 3-2_1
OSNR, dB
Samples, s

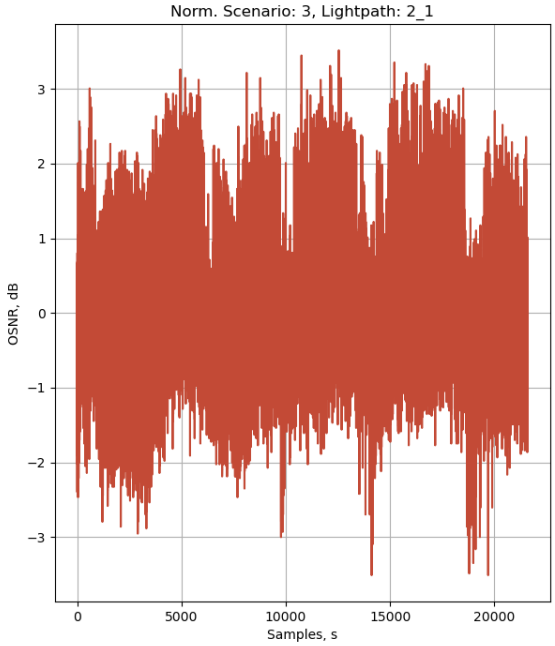

Raw Scenario: 2, Lightpath: 2_1
OSNR, dB
Samples, s

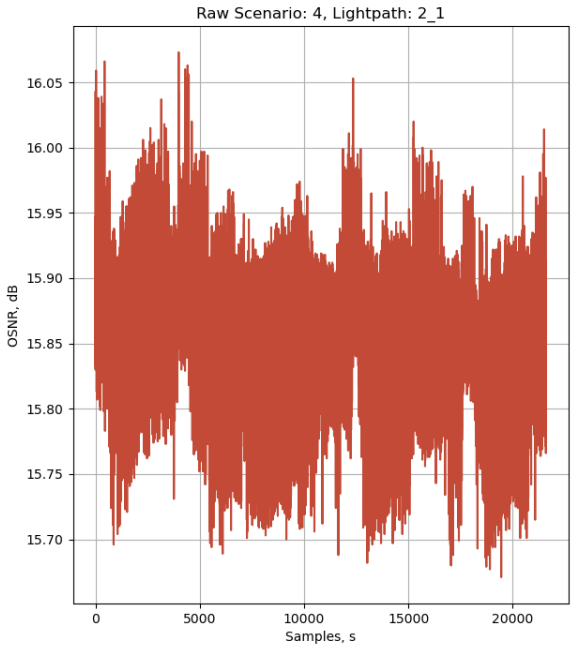

Norm. Scenario: 2, Lightpath: 2_1
OSNR, dB
Samples, s

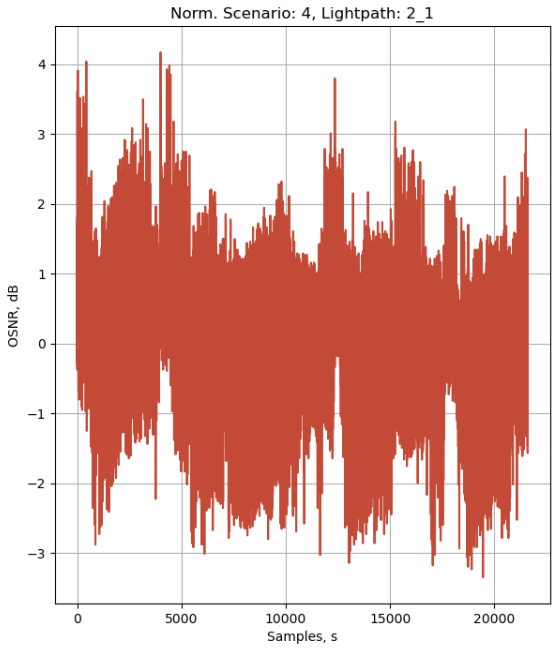

Raw Scenario: 2, Lightpath: 3-1_1
OSNR, dB
Samples, s

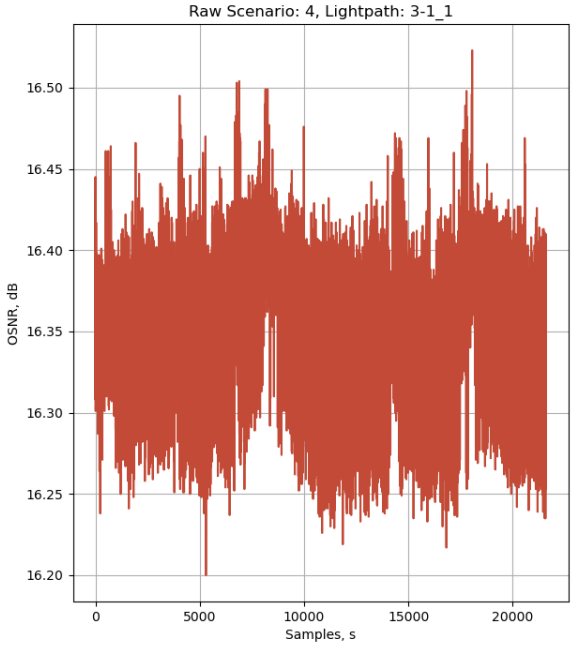

Norm. Scenario: 2, Lightpath: 3-1_1
OSNR, dB
Samples, s

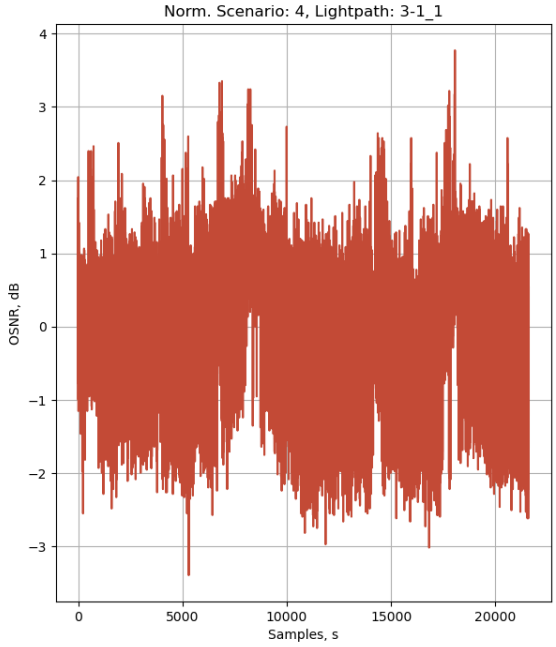

Raw Scenario: 3, Lightpath: 2_1
OSNR, dB
Samples, s

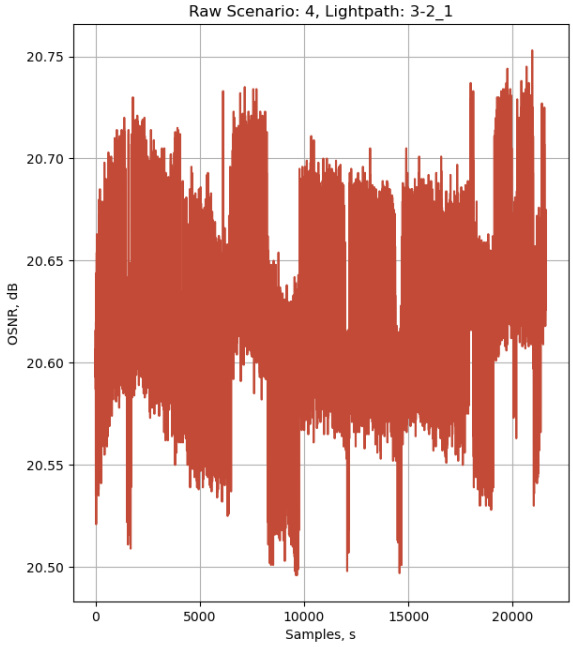

Norm. Scenario: 3, Lightpath: 2_1
OSNR, dB
Samples, s

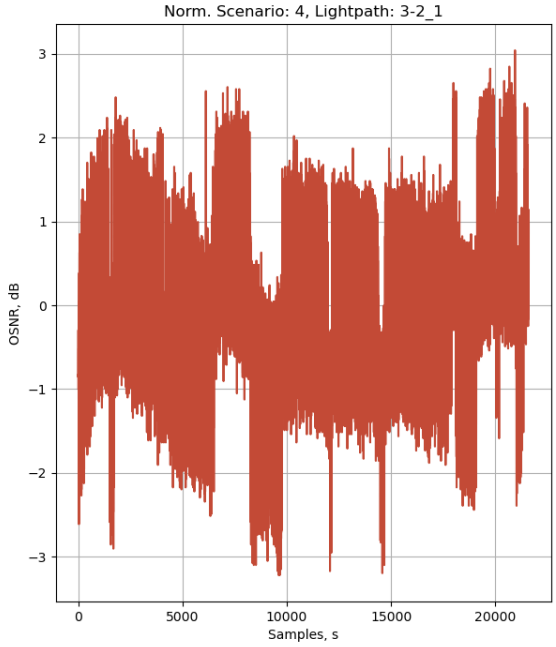

Raw Scenario: 4, Lightpath: 2_1
OSNR, dB
Samples, s

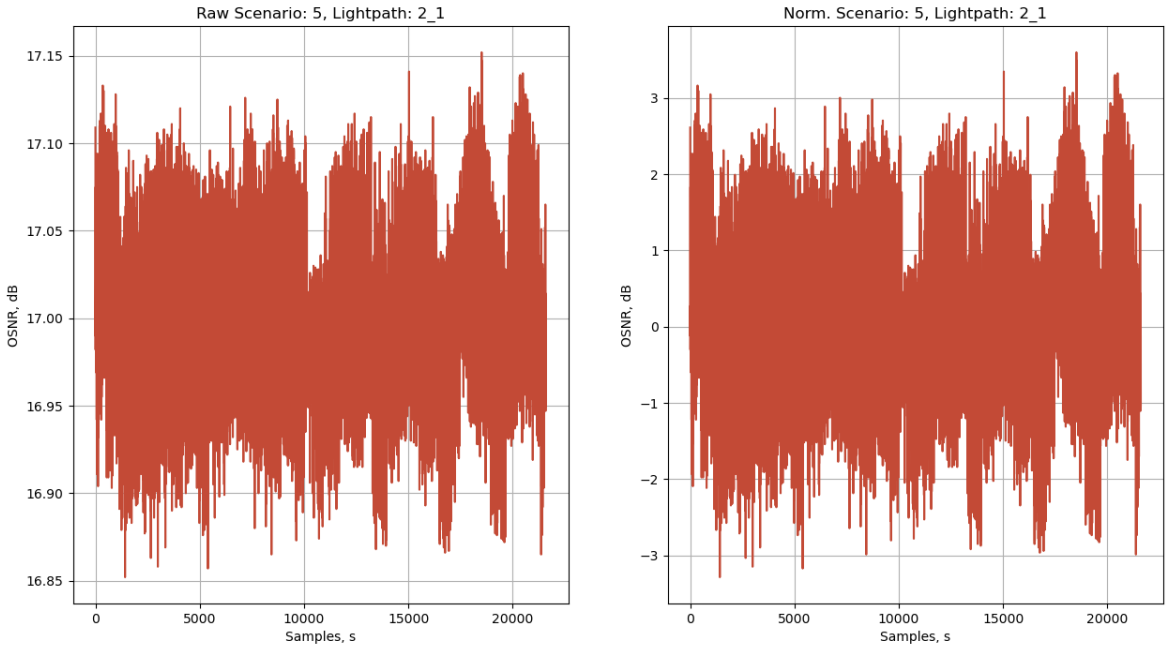

Norm. Scenario: 4, Lightpath: 2_1
OSNR, dB
Samples, s

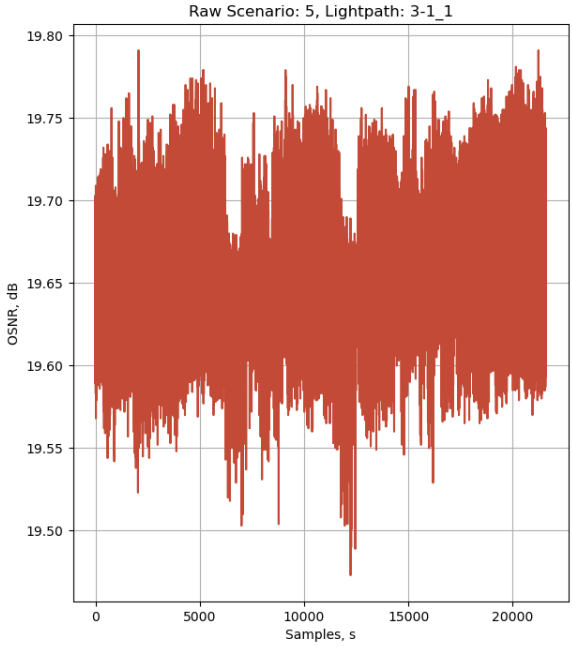

Raw Scenario: 4, Lightpath: 3-1_1
OSNR, dB
Samples, s

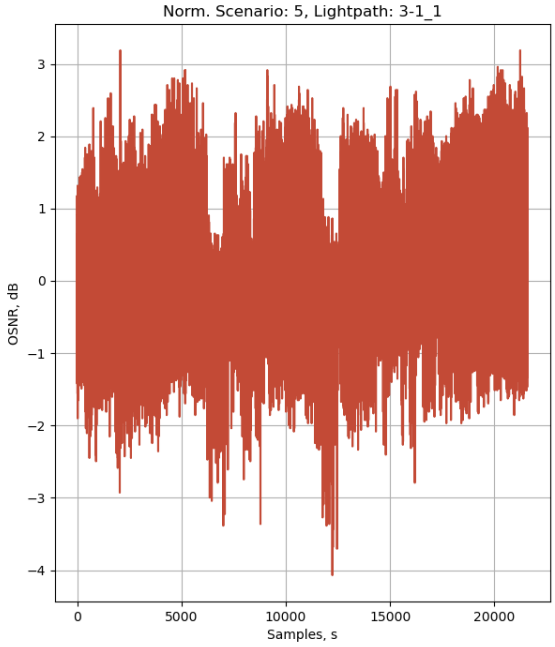

Norm. Scenario: 4, Lightpath: 3-1_1
OSNR, dB
Samples, s

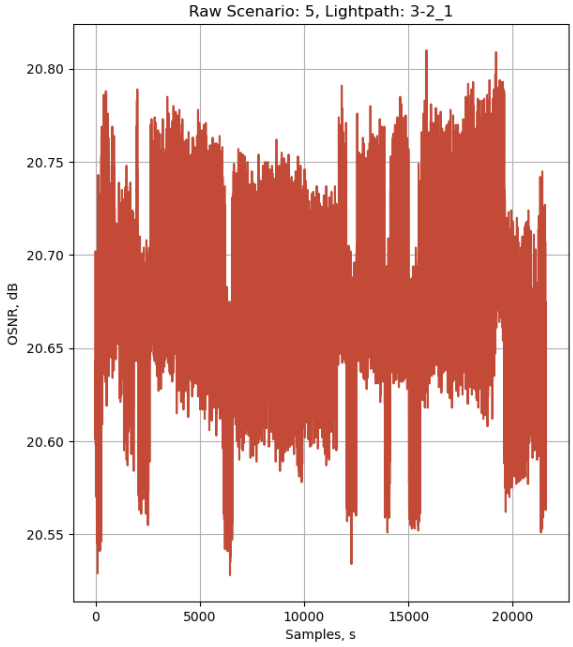

Raw Scenario: 4, Lightpath: 3-2_1
OSNR, dB
Samples, s

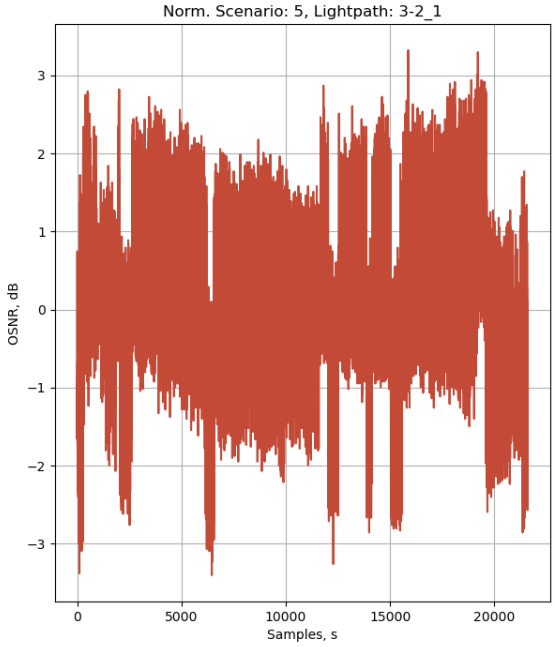

Norm. Scenario: 4, Lightpath: 3-2_1
OSNR, dB
Samples, s


Raw Scenario: 5, Lightpath: 2_1
OSNR, dB
Samples, s
17.15
17.10
17.05
17.00
16.95
16.90
16.85
0
5000
10000
15000
20000
Norm. Scenario: 5, Lightpath: 2_1
OSNR, dB
Samples, s
3
2
1
0
−1
−2
−3
0
5000
10000
15000
20000

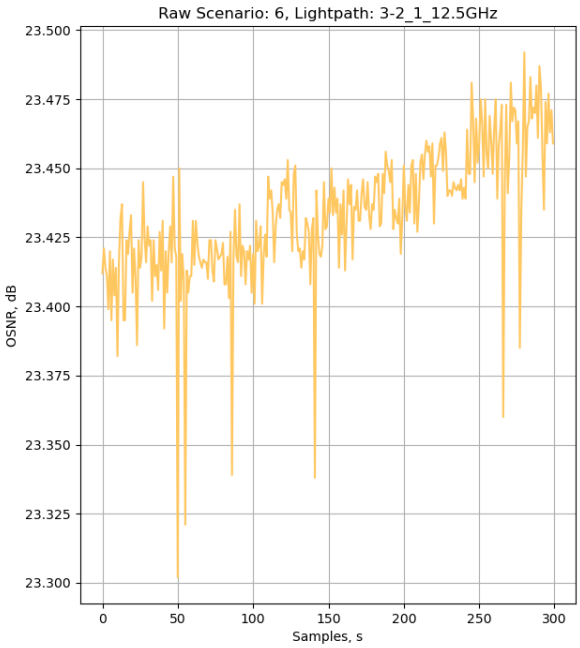

Raw Scenario: 5, Lightpath: 3-1_1
OSNR, dB
Samples, s

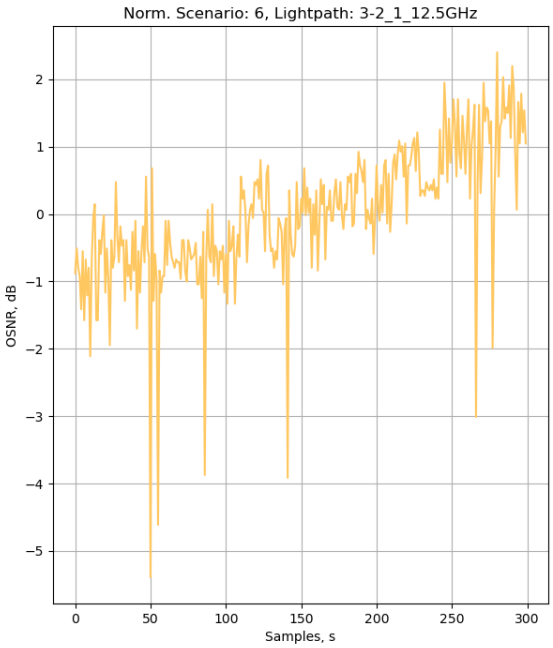

Norm. Scenario: 5, Lightpath: 3-1_1
OSNR, dB
Samples, s

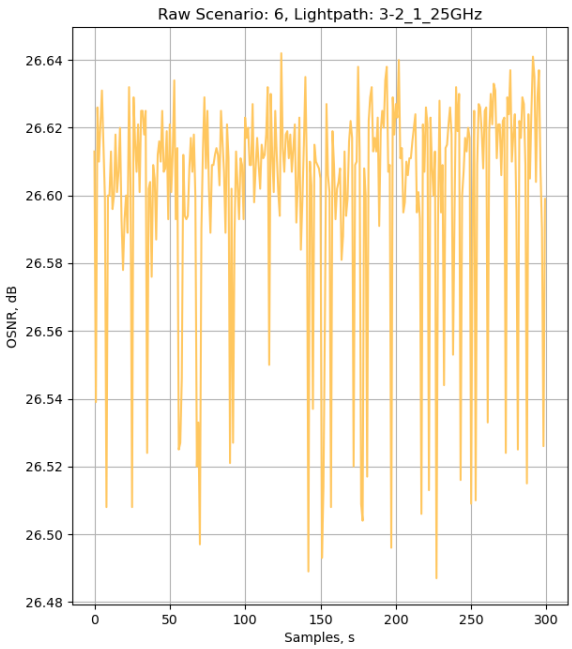

Raw Scenario: 5, Lightpath: 3-2_1
OSNR, dB
Samples, s

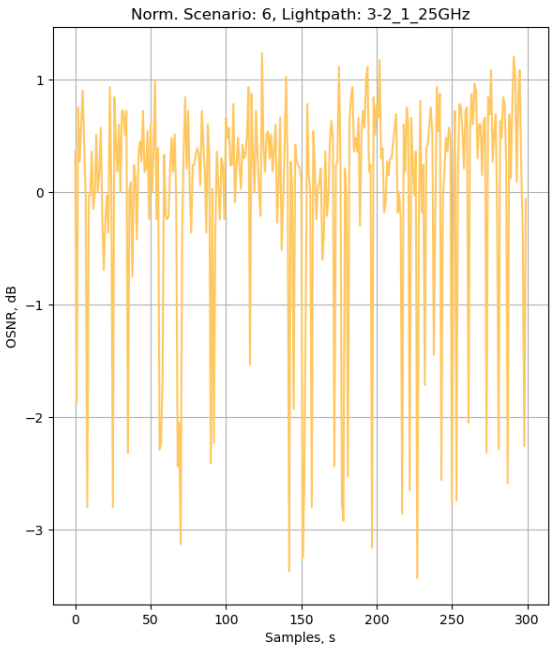

Norm. Scenario: 5, Lightpath: 3-2_1
OSNR, dB
Samples, s

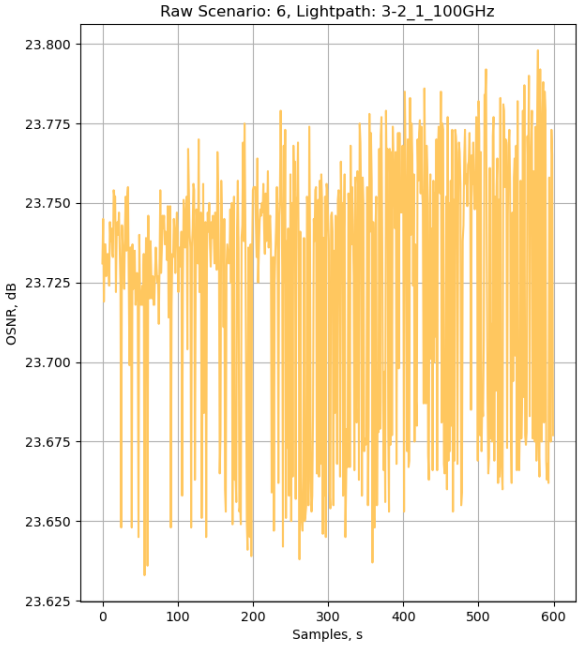
Scenarios with Filtering Failure

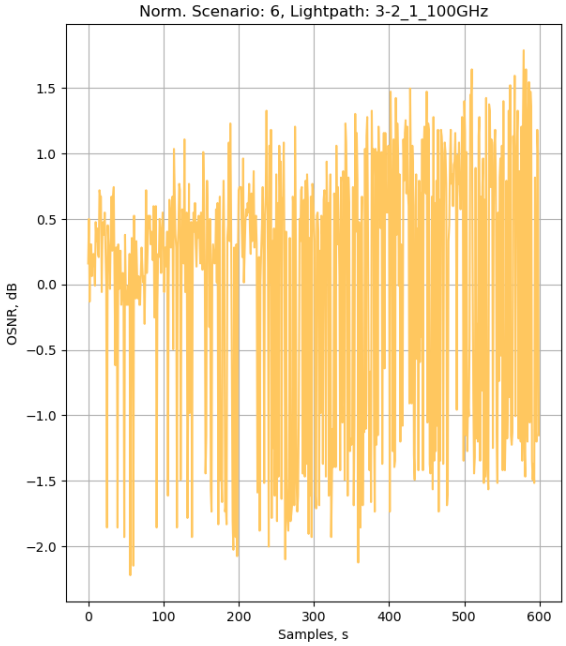


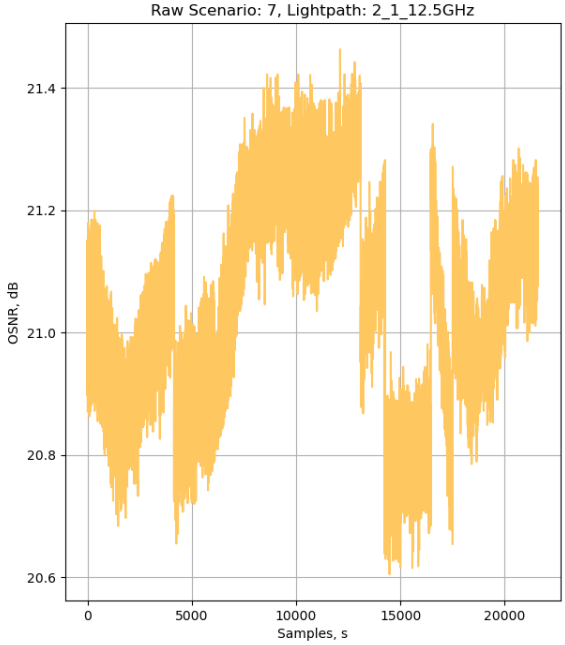


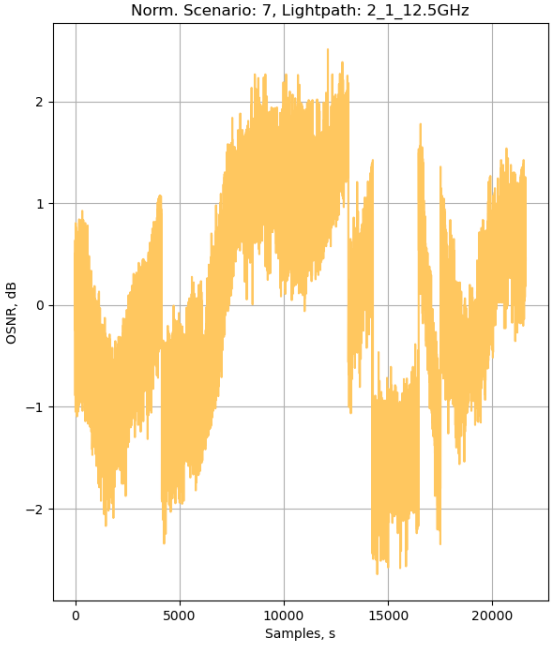


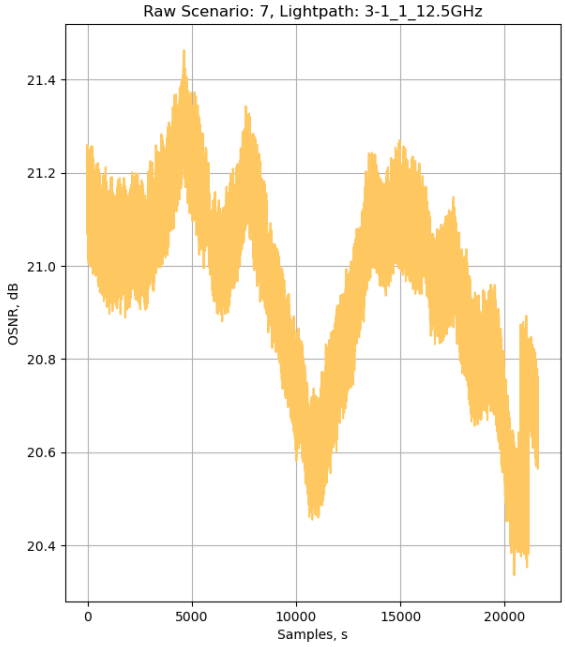


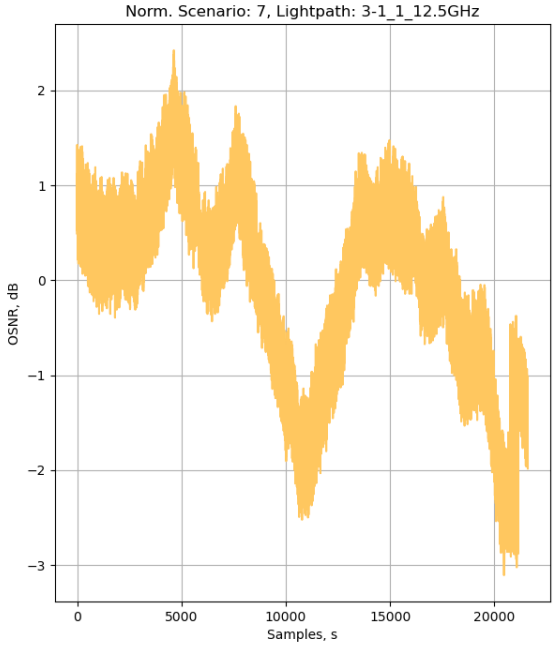

Raw Scenario: 6, Lightpath: 3-2_1_100GHz
OSNR, dB
Samples, s
23.800
23.775
23.750
23.725
23.700
23.675
23.650
23.625
0
100
200
300
400
500
600

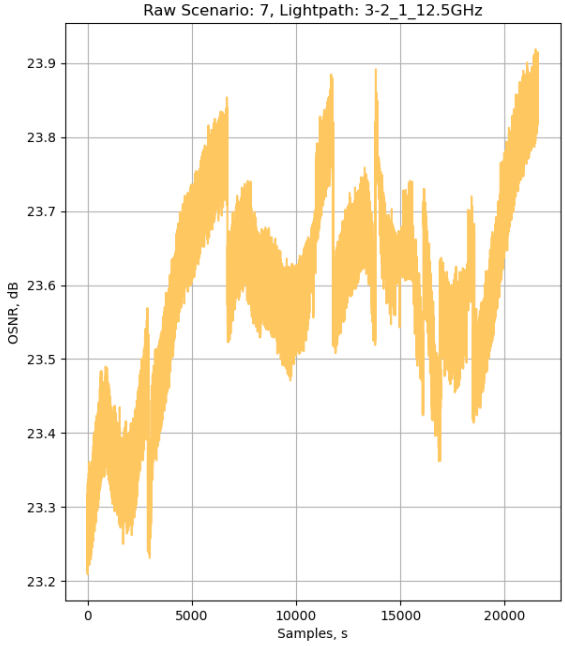

Norm. Scenario: 6, Lightpath: 3-2_1_100GHz
OSNR, dB
Samples, s
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
100
200
300
400
500
600

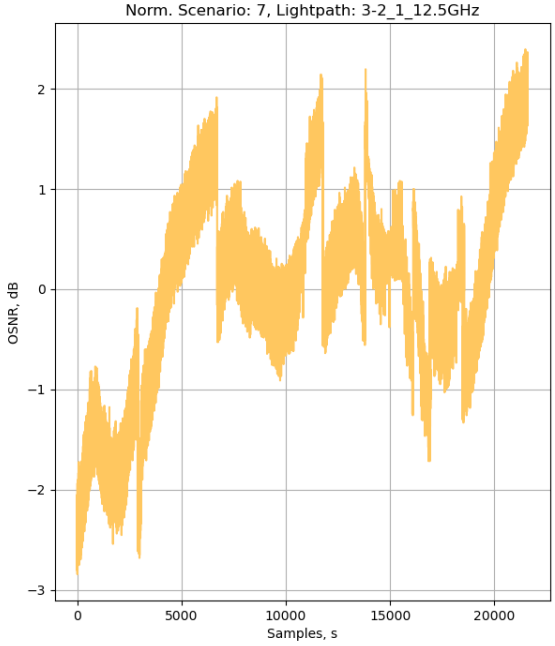

Raw Scenario: 7, Lightpath: 2_1_12.5GHz
OSNR, dB
Samples, s
21.4
21.2
21.0
20.8
20.6
0
5000
10000
15000
20000

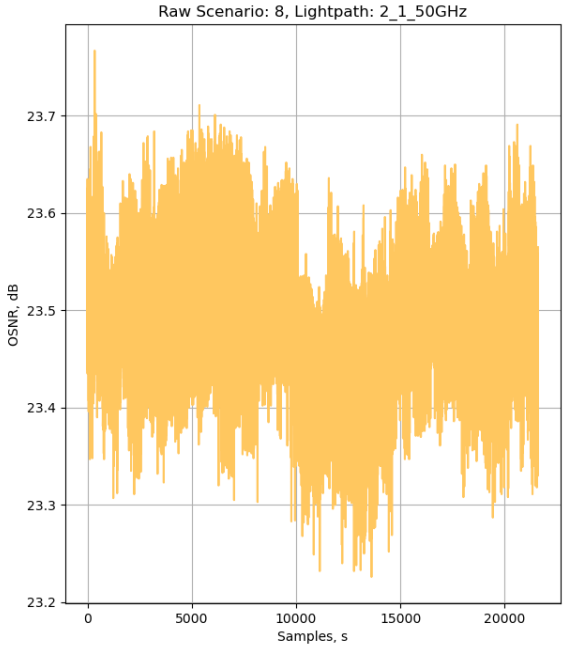

Norm. Scenario: 7, Lightpath: 2_1_12.5GHz
OSNR, dB
Samples, s
2
1
0
−1
−2
0
5000
10000
15000
20000

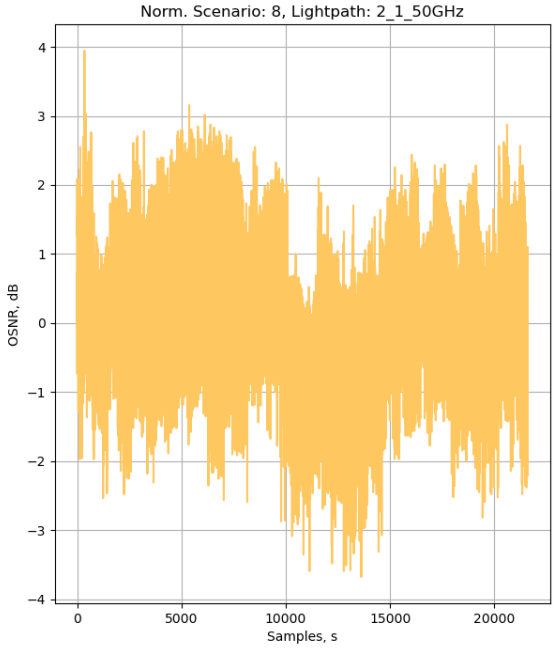

Raw Scenario: 7, Lightpath: 3-1_1_12.5GHz
OSNR, dB
Samples, s

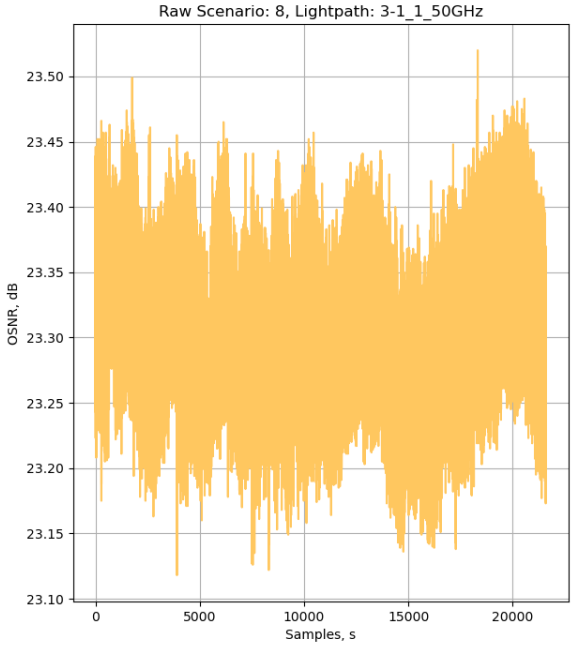

Norm. Scenario: 7, Lightpath: 3-1_1_12.5GHz
OSNR, dB
Samples, s

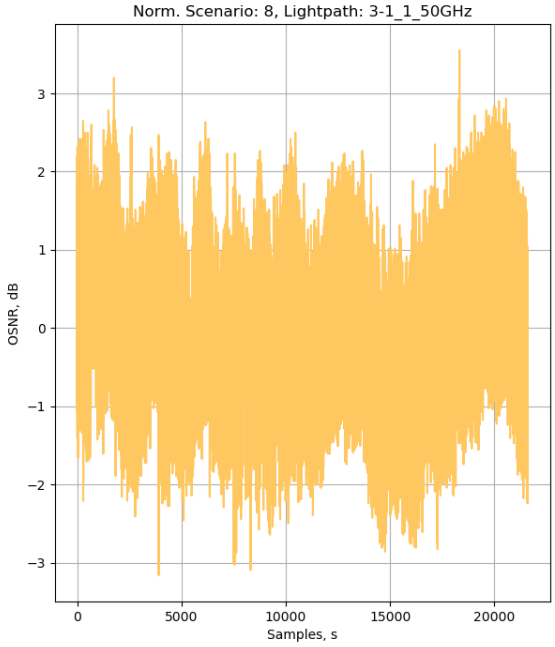

Raw Scenario: 7, Lightpath: 3-2_1_12.5GHz
OSNR, dB
Samples, s

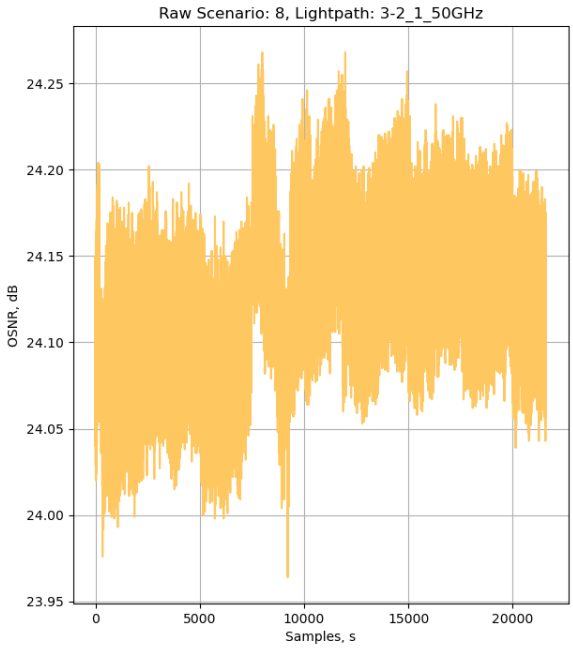

Norm. Scenario: 7, Lightpath: 3-2_1_12.5GHz
OSNR, dB
Samples, s

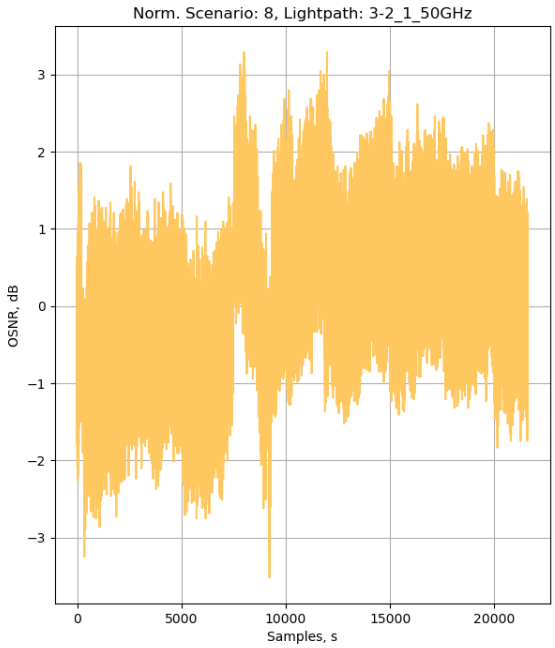

Raw Scenario: 8, Lightpath: 2_1_50GHz
OSNR, dB
23.7
23.6
23.5
23.4
23.3
23.2
0
5000
10000
15000
20000
Samples, s


Norm. Scenario: 8, Lightpath: 2_1_50GHz
OSNR, dB
4
3
2
1
0
−1
−2
−3
−4
0
5000
10000
15000
20000
Samples, s


Raw Scenario: 8, Lightpath: 3-1_1_50GHz
OSNR, dB
Samples, s


Norm. Scenario: 8, Lightpath: 3-1_1_50GHz
OSNR, dB
Samples, s

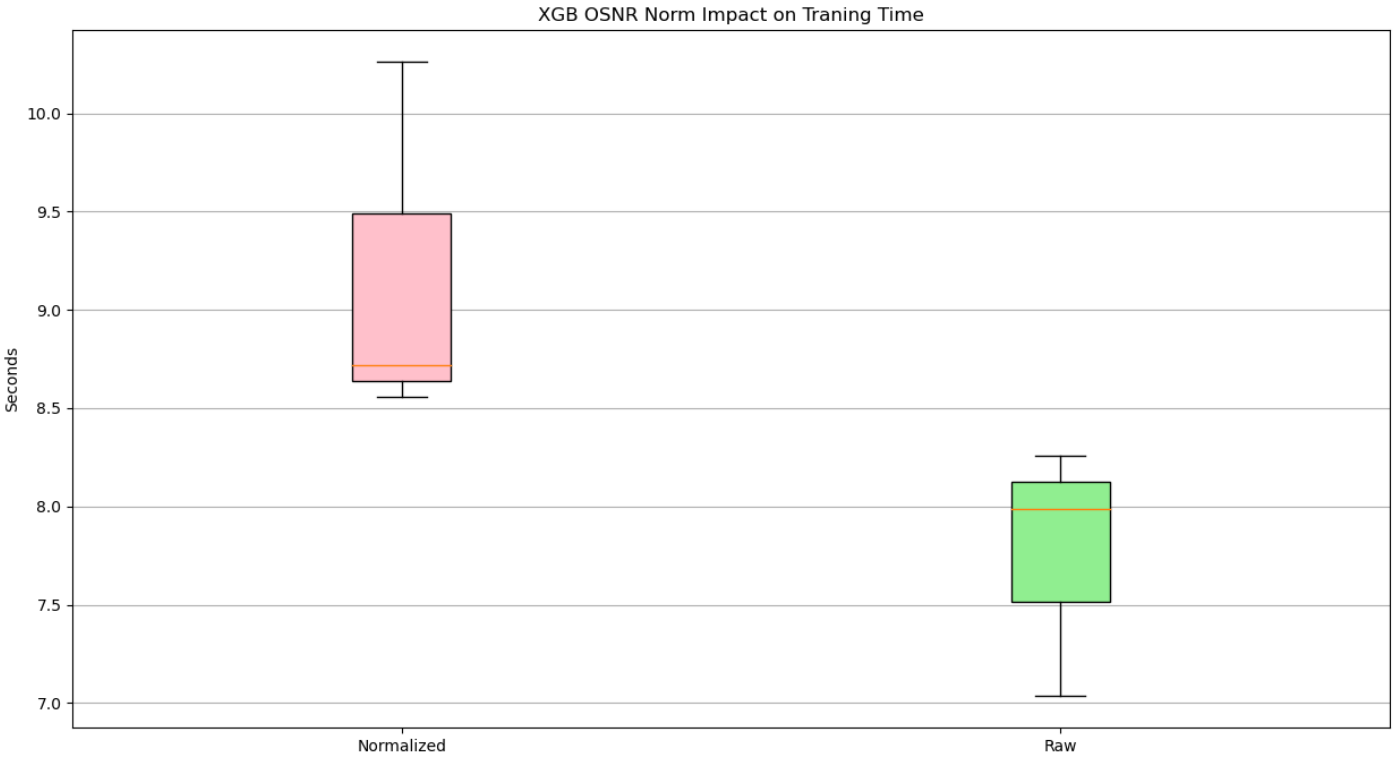

Raw Scenario: 8, Lightpath: 3-2_1_50GHz
OSNR, dB
Samples, s


Norm. Scenario: 8, Lightpath: 3-2_1_50GHz
OSNR, dB
Samples, s

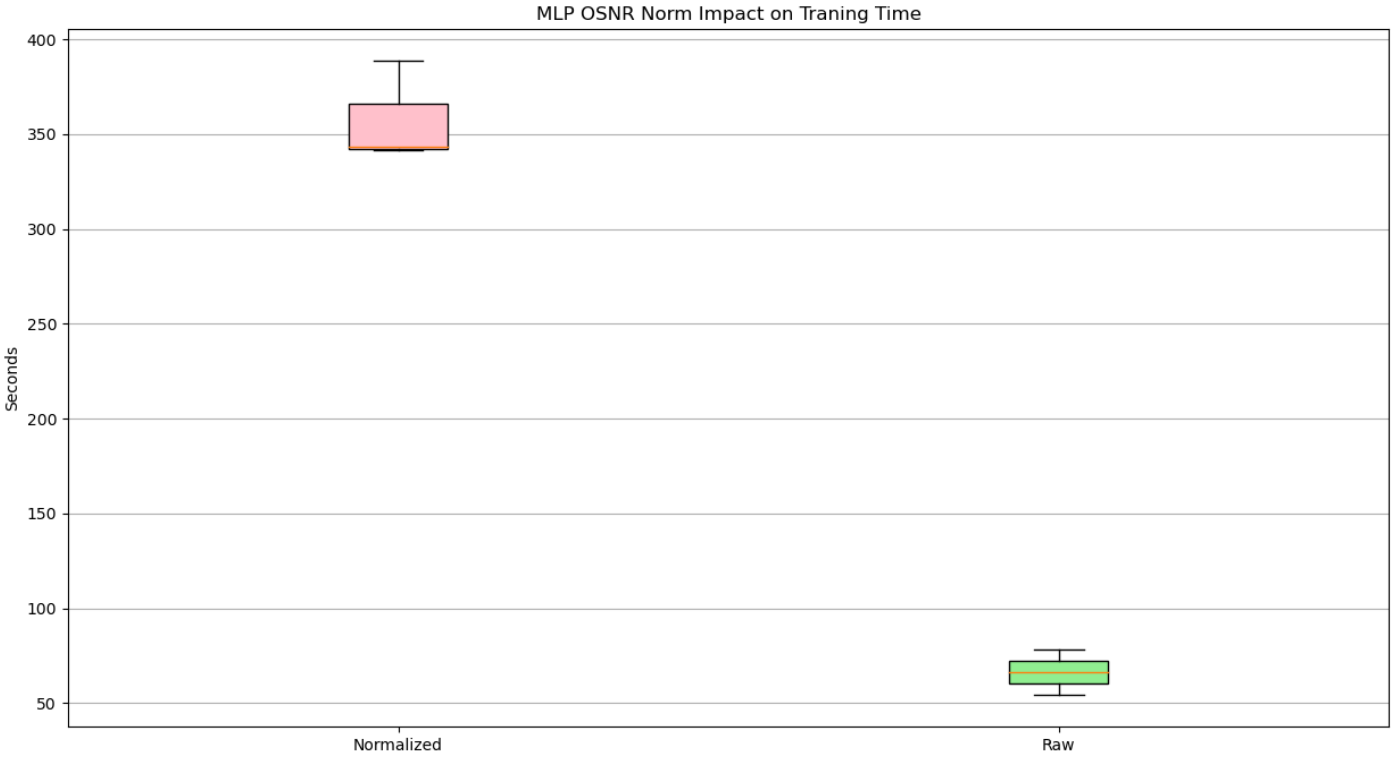
Training Time Comparison

XGB Training Time Comparison

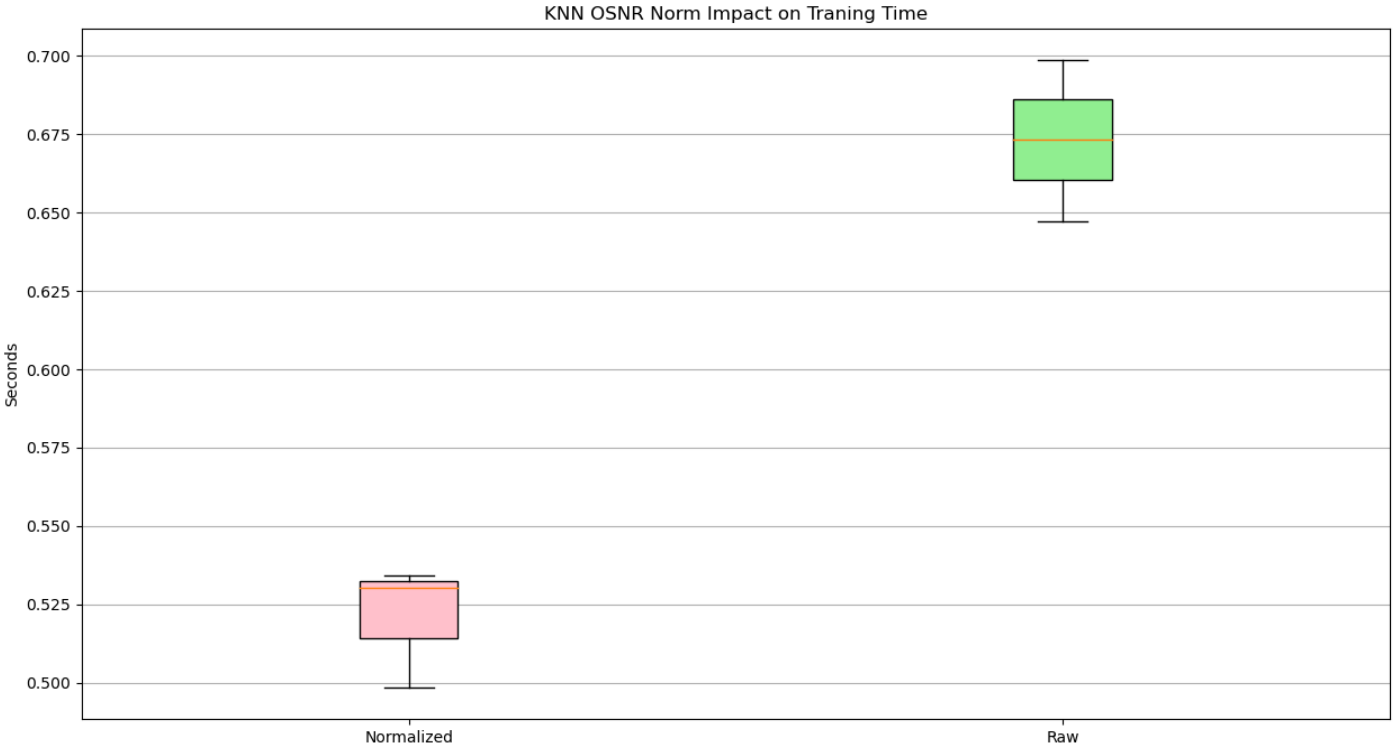
The normalization impacts negatively on the XGB and MLP classifiers while positively on the KNN.



MLP Training Time Comparison



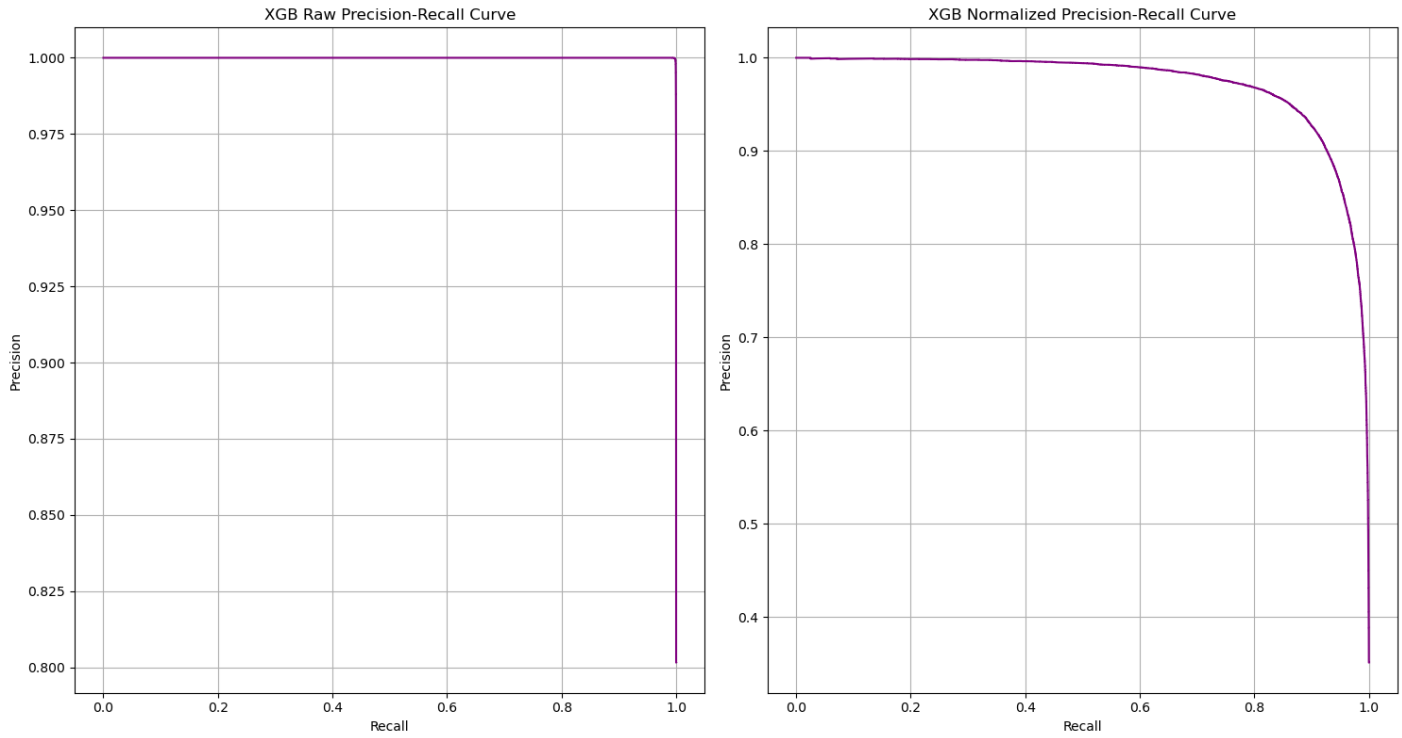
KNN Training Time Comparison



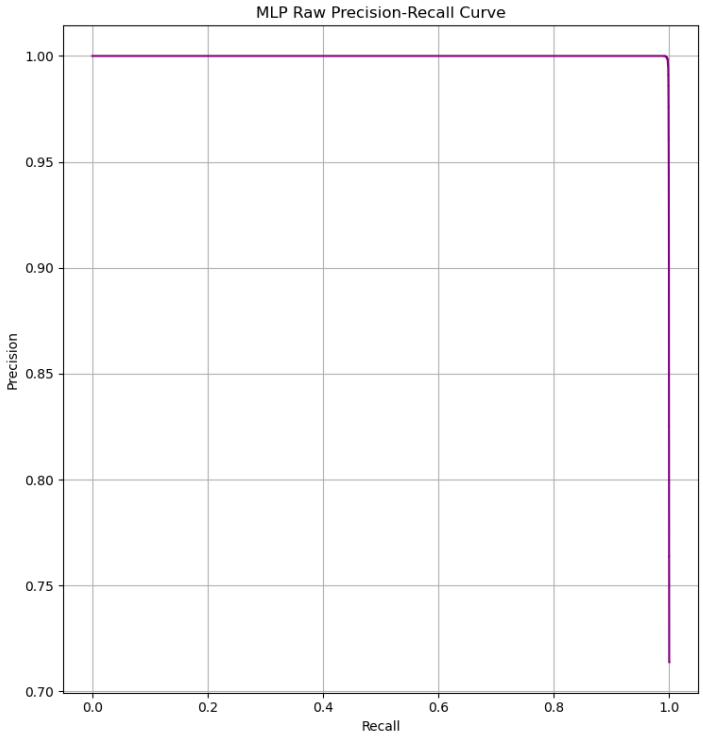
Precision-Recall Curve Comparison

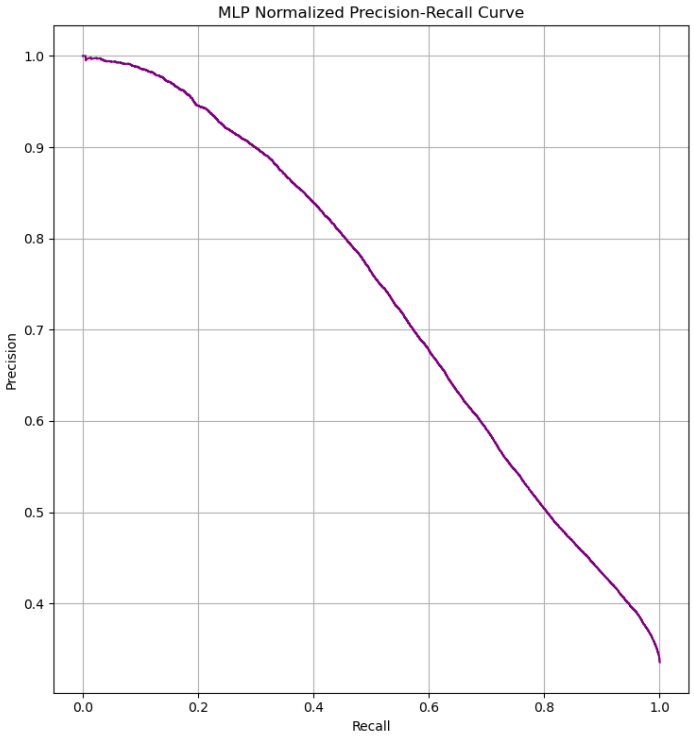
The impact of normalization on precision-recall and ROC has also been analyzed and plotted in the graphs below. The curves plotted below show how severely negative was the impact of normalization on the performances.

XGB Precision-Recall Curve

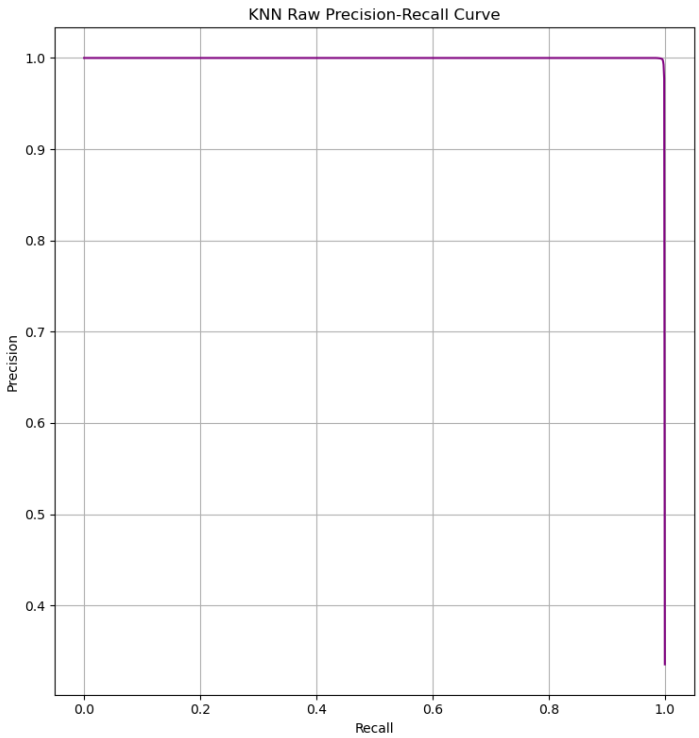


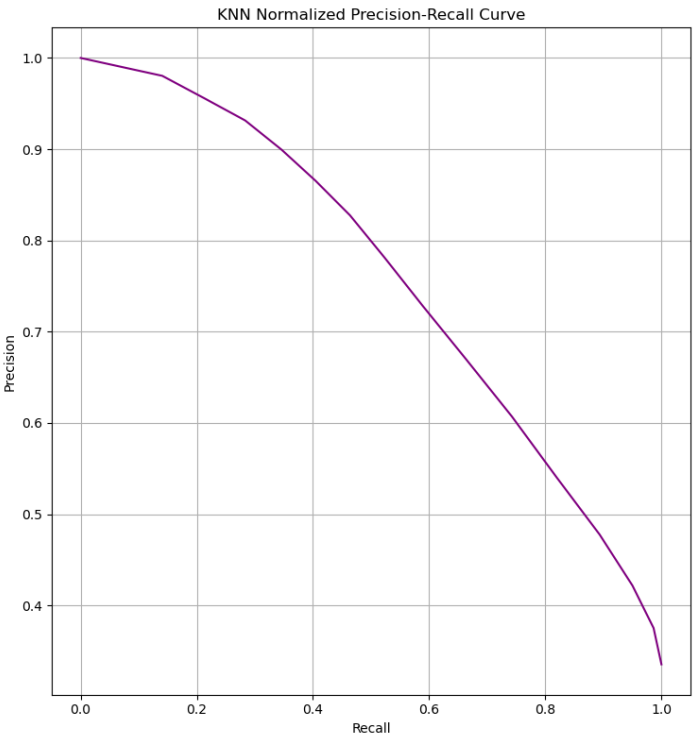
MLP Precision-Recall Curve





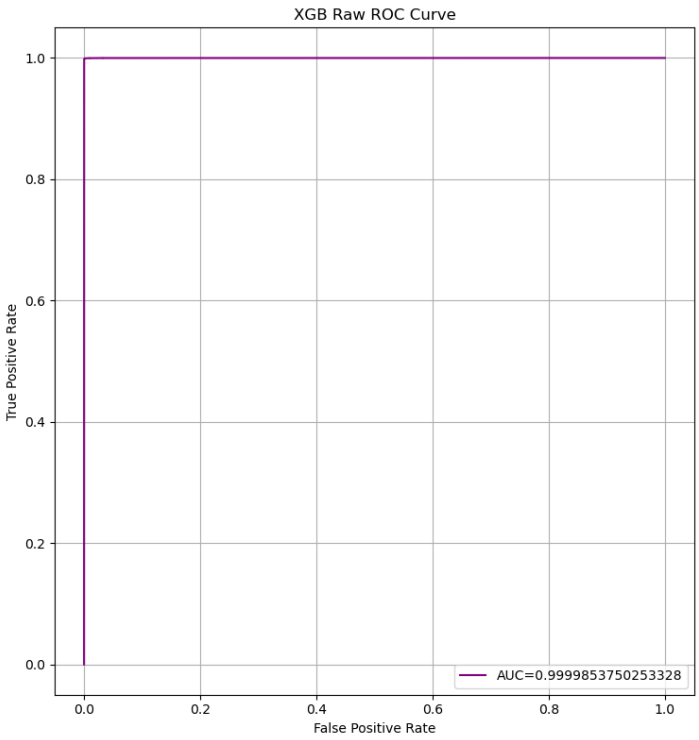
KNN Precision-Recall Curve

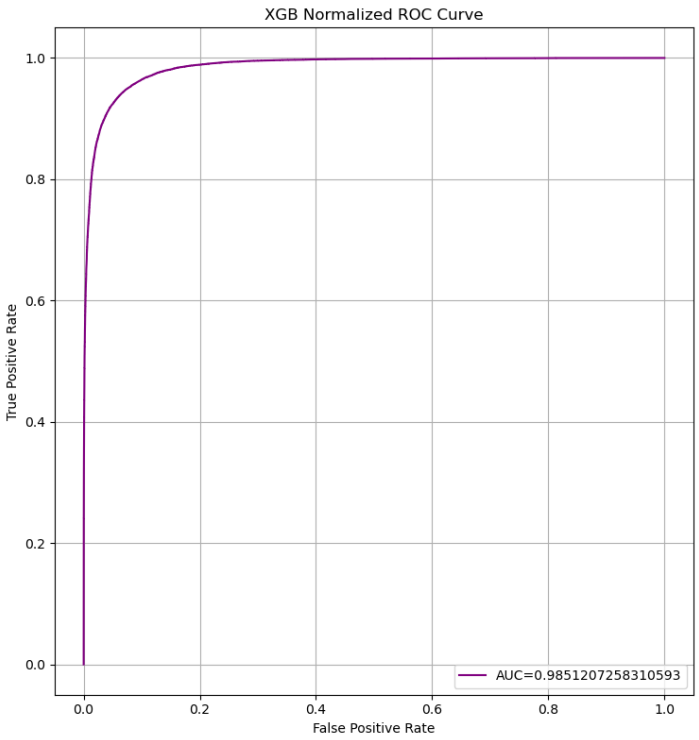




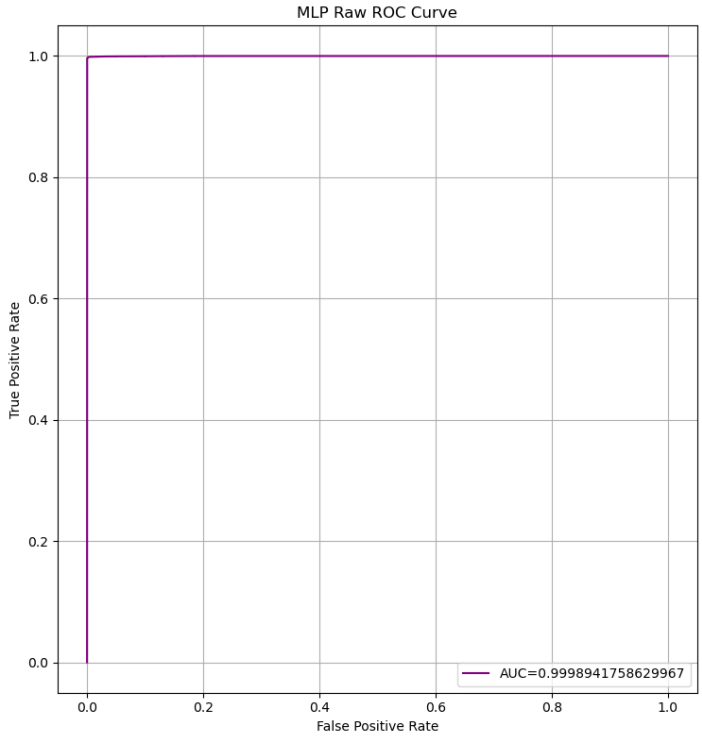
ROC Curve Comparison

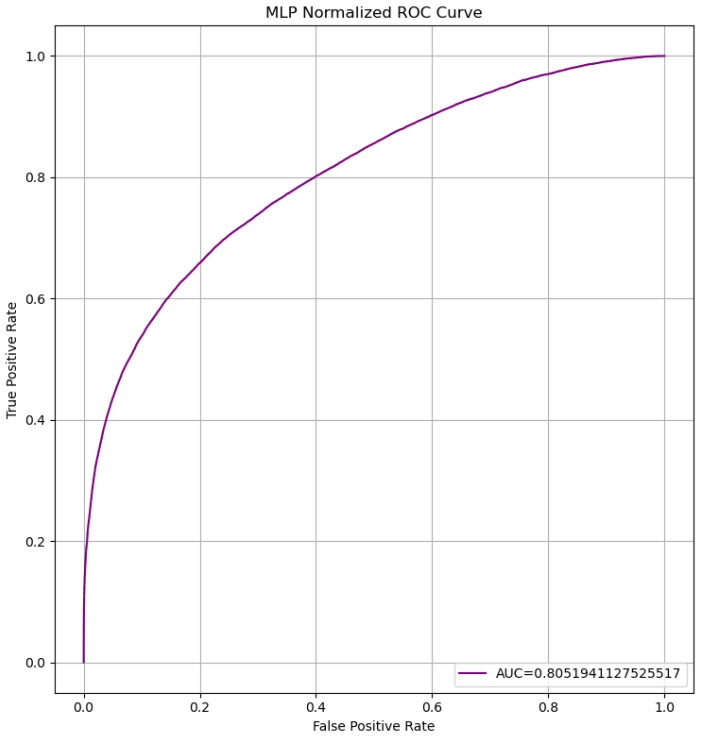
XGB ROC Curve



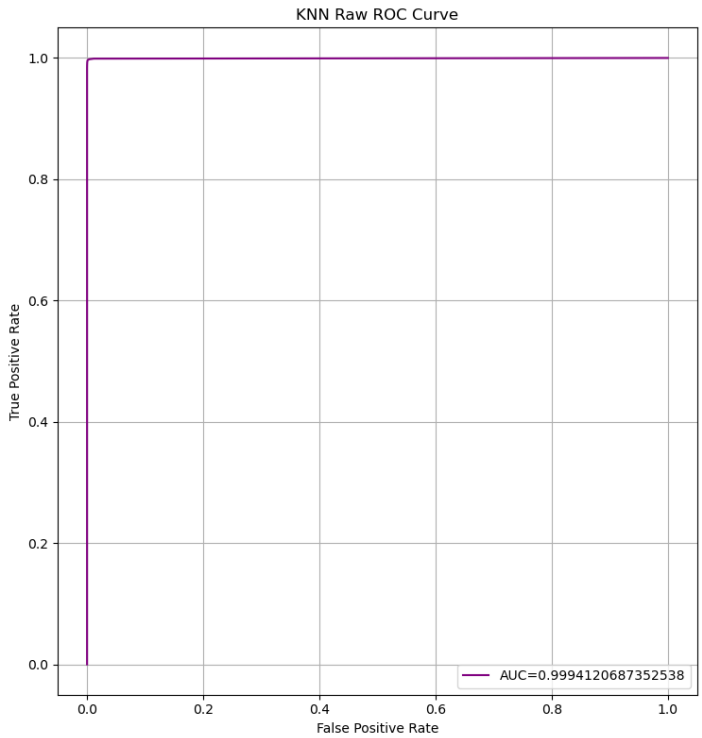


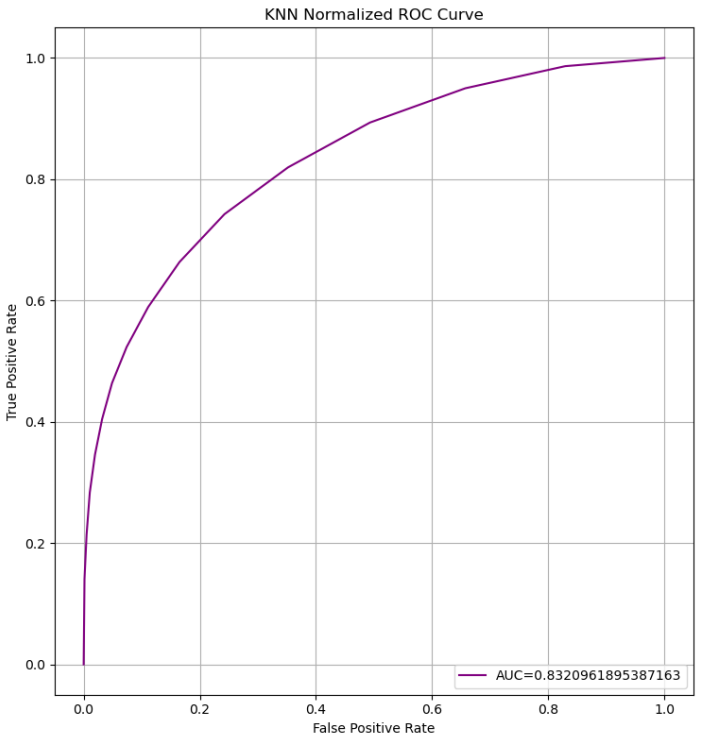
MLP ROC Curve





KNN ROC Curve

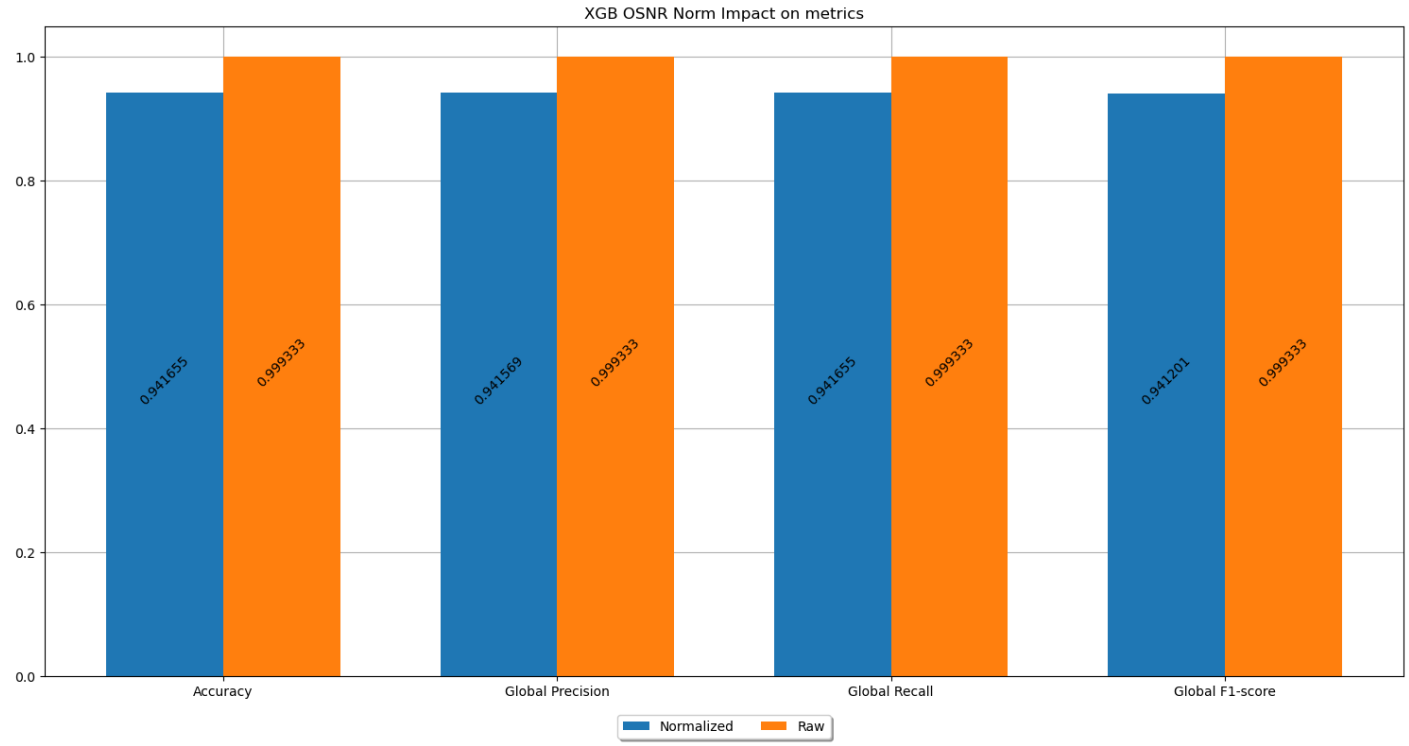




Accuracy, Precision, Recall, and F-1 score Comparison

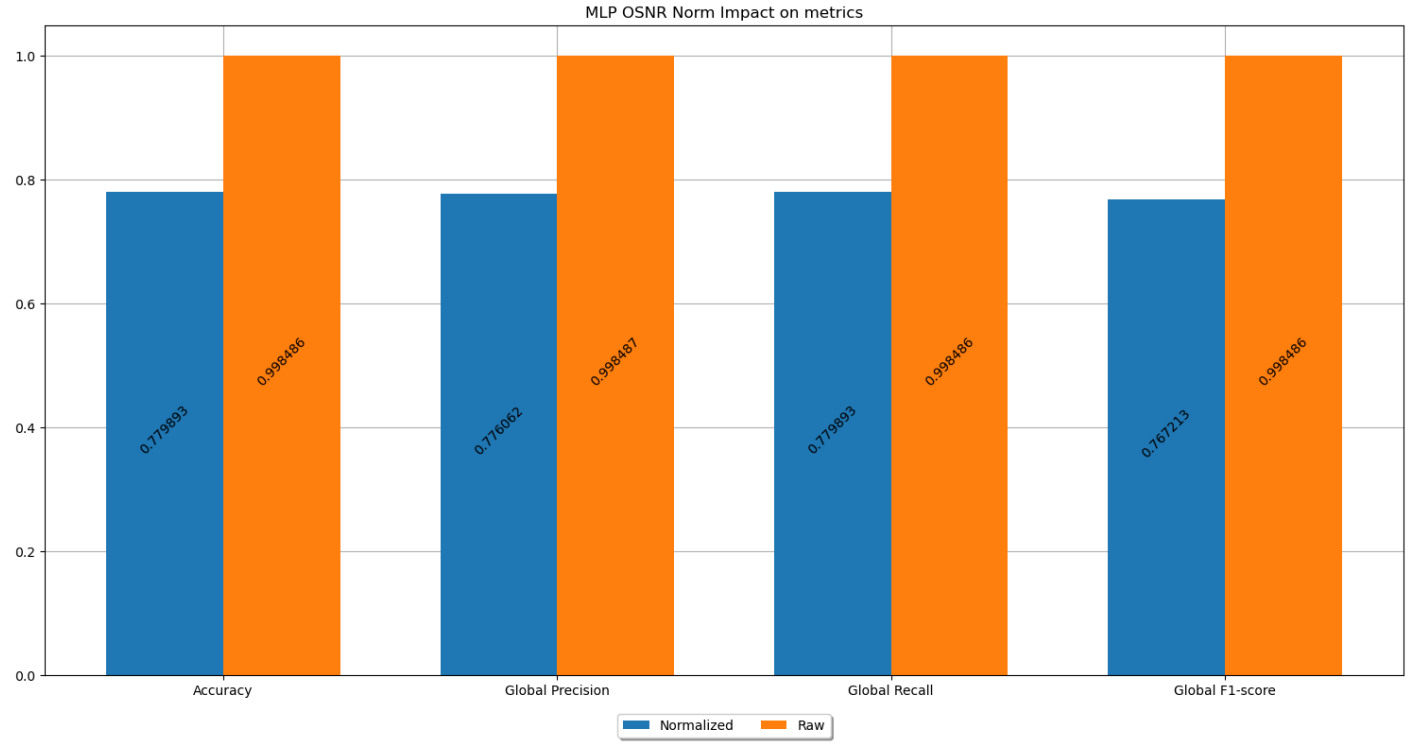
The normalization had on average a negative impact on the metrics as shown in the graph below.

XGB Metrics Comparison

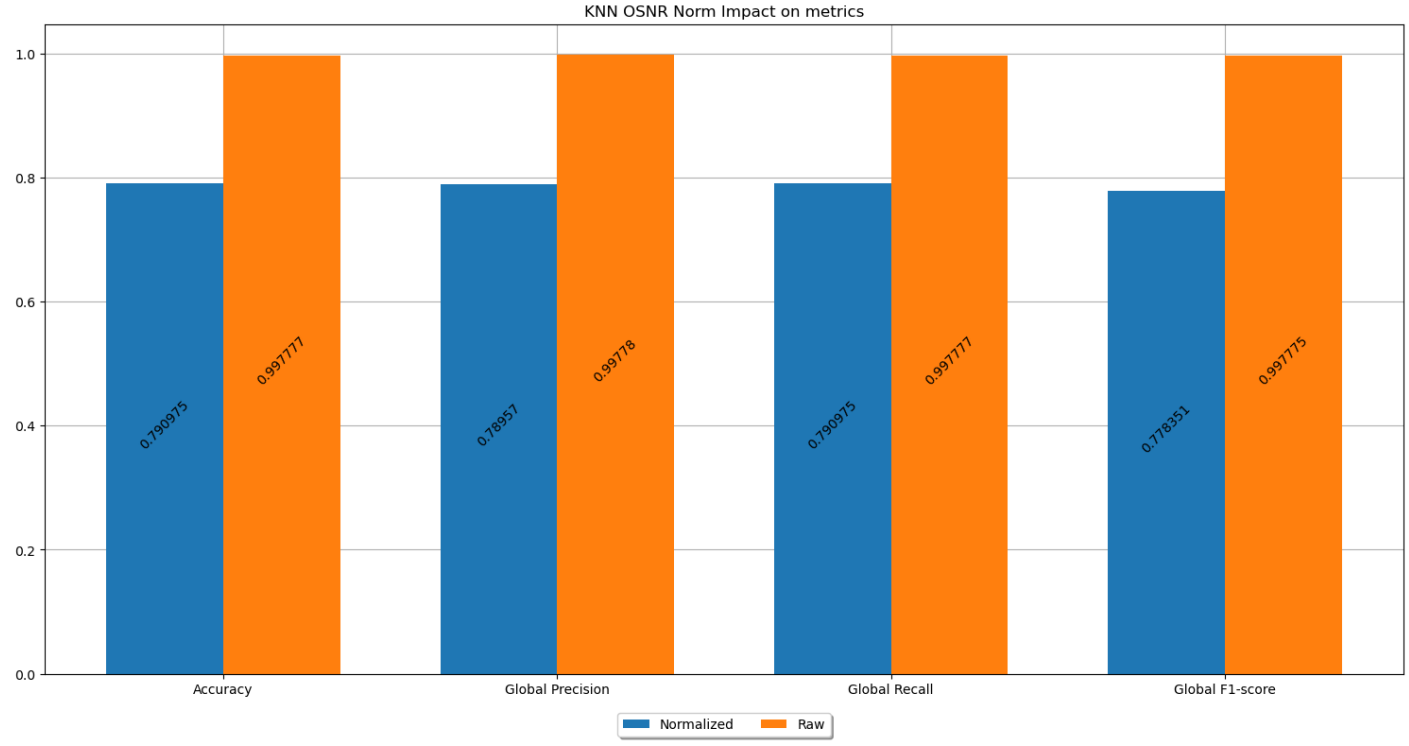


This happens because normalizing the OSNR before extracting the features results in a loss of entropy and data dispersion, especially in the case of attenuation errors for the new considered range. Thus for instance in the XGBoost classifier, the branches of the tree will have less distance and less or in the KNN case, the centroids will be less distant.

MLP Metrics Comparison



KNN Metrics Comparison



Sitography

[1] “How to Use StandardScaler and MinMaxScaler Transforms in Python” - Jason Brownlee

https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

[2] “Understanding AUC - ROC Curve” - Sarang Narkhede
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[3] “Why the time complexity for training K-Nearest Neighbors is O(1)”
https://medium.com/nerd-for-tech/why-the-time-complexity-for-training-k-nearest-neighbors-is-o-1-5b8f417104cf