**GSNR Prediction using Supervised Learning**

# **Introduction**

## **Overview of the Problem**

Predicting General Signal-to-Noise Ratio (GSNR) is critical for optimizing the performance of communication systems. Accurate GSNR predictions can enhance signal quality, reduce errors, and improve overall system efficiency. This project aims to leverage supervised learning techniques to develop a robust model that can predict GSNR values based on various input features.

## **Description of the Dataset**

The dataset used in this project, European Topology 6 Paths, contains multiple features that influence the GSNR for 76 channels. It includes multiple features such as power, NLI (non-linear interference), ASE (amplified spontaneous emission), total distance, span and frequency. Each row in the dataset corresponds to a unique observation, providing a comprehensive set of data points for training and evaluating predictive models. The goal is to use this dataset to train various machine learning models and identify the one that delivers the most accurate GSNR predictions.

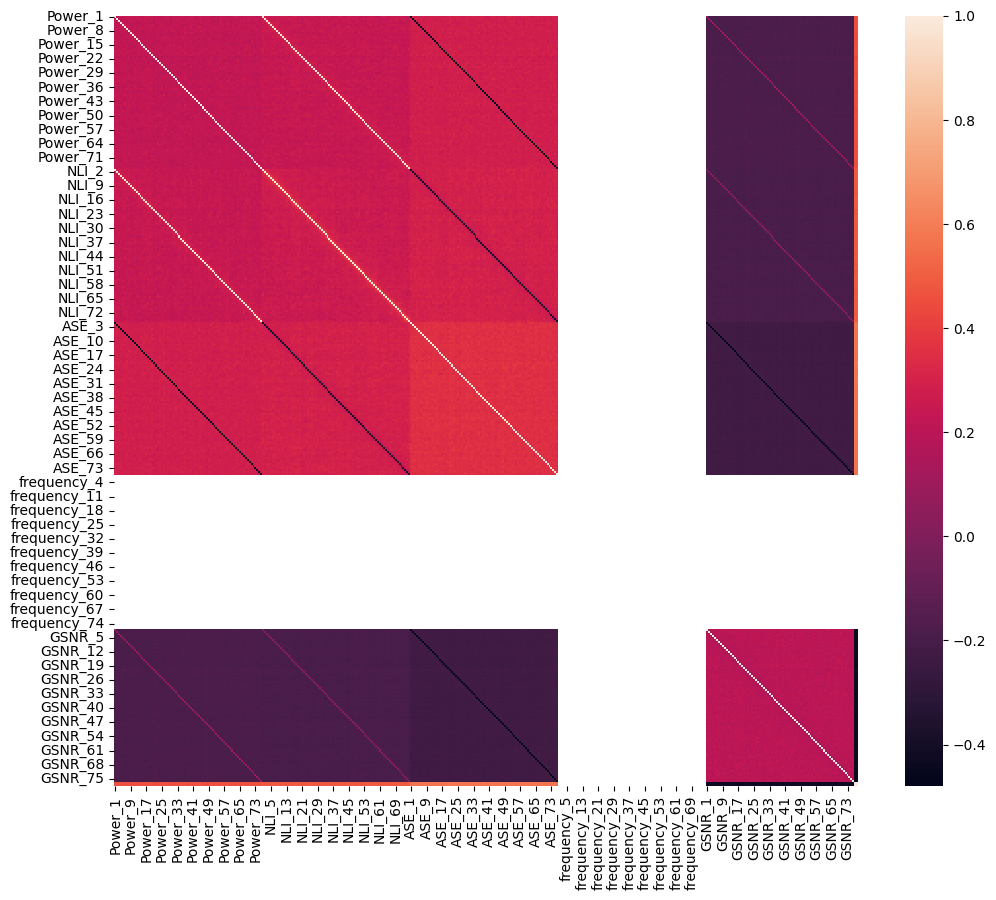
# **Understanding the Data**

We used correlation matrix heatmaps to explore the relationships between different features of the dataset. The provided heatmap of the entire dataset visually represents the correlations between all the features, which can help identify which features are strongly correlated with each other.

## **Correlation Matrix Heatmap**

The correlation matrix heatmap is a useful tool for visualizing the strength and direction of linear relationships between features. In the heatmap:

* **Color Intensity**: Indicates the strength of the correlation, with darker colors representing stronger correlations.
* **Positive Correlations**: Features that increase together are shown in shades of red.
* **Negative Correlations**: Features where one decreases as the other increases are shown in shades of blue or purple.

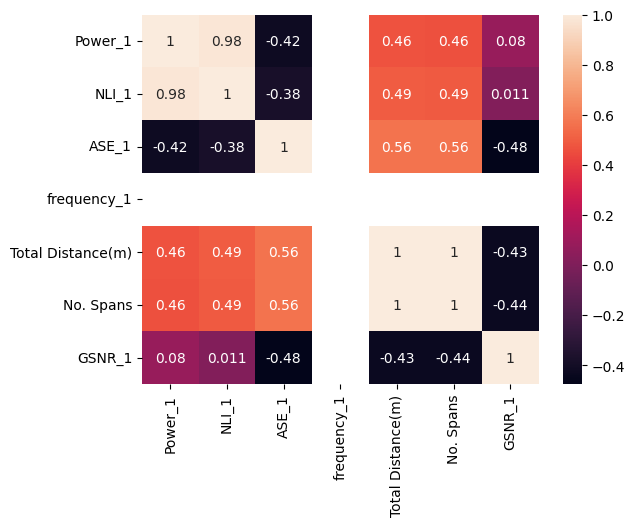


*Figure 1. Correlation Matrix Heatmap for Complete Dataset*

From the above plot we can see that frequency features have almost zero relation with other features. But still we cannot clearly see what is happening here. So for a better understanding, we will be looking at some different channels to get a better view.

## **Channel 1**

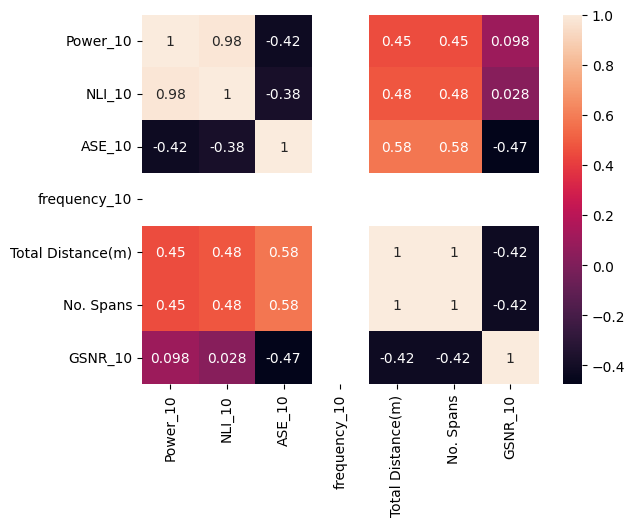
The following heatmap visualizes the correlation matrix for the dataset containing the columns 'Power\_1', 'NLI\_1', 'ASE\_1', 'frequency\_1', 'Total Distance(m)', 'No. Spans', and 'GSNR\_1'. Each cell in the heatmap represents the correlation coefficient between two variables, with values ranging from -1 to 1. Positive values indicate a direct relationship, while negative values indicate an inverse relationship. For example, 'Power\_1' and 'NLI\_1' have a high positive correlation (0.98), indicating that they vary together. On the other hand, 'ASE\_1' and 'GSNR\_1' have a negative correlation (-0.48), suggesting that as one increases, the other decreases. The heatmap uses color coding to visually represent the strength and direction of these correlations, making it easier to identify strong relationships between variables.



*Figure 2. Correlation Matrix Heatmap for Channel 1*

## **Channel 10**

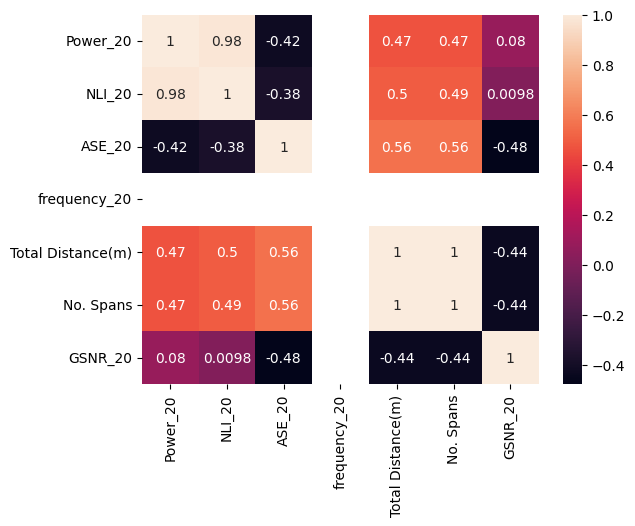
The following heatmap visualizes the correlation matrix for the dataset containing the columns 'Power\_10', 'NLI\_10', 'ASE\_10', 'frequency\_10', 'Total Distance(m)', 'No. Spans', and 'GSNR\_10'. Each cell in the heatmap represents the correlation coefficient between pairs of variables, with the values ranging from -1 to 1. Positive correlations indicate that as one variable increases, the other tends to increase, while negative correlations indicate an inverse relationship. For instance, 'Power\_10' and 'NLI\_10' have a high positive correlation of 0.98, suggesting a strong direct relationship. Conversely, 'ASE\_10' and 'GSNR\_10' have a negative correlation of -0.47, indicating an inverse relationship. The color intensity in the heatmap provides a visual cue for the strength and direction of these correlations, making it easier to quickly identify strong positive or negative relationships between variables.



*Figure 3. Correlation Matrix Heatmap for Channel 10*

## **Channel 20**

The heatmap visualizes the correlation matrix for the dataset containing the columns 'Power\_20', 'NLI\_20', 'ASE\_20', 'frequency\_20', 'Total Distance(m)', 'No. Spans', and 'GSNR\_20'. Each cell in the heatmap represents the correlation coefficient between two variables, with values ranging from -1 to 1. Positive values indicate a direct relationship, while negative values indicate an inverse relationship. For example, 'Power\_20' and 'NLI\_20' have a high positive correlation (0.98), indicating that they vary together. On the other hand, 'ASE\_20' and 'GSNR\_20' have a negative correlation (-0.48), suggesting that as one increases, the other decreases. The heatmap uses color coding to visually represent the strength and direction of these correlations, making it easier to identify strong relationships between variables.



*Figure 4. Correlation Matrix Heatmap for Channel 20*

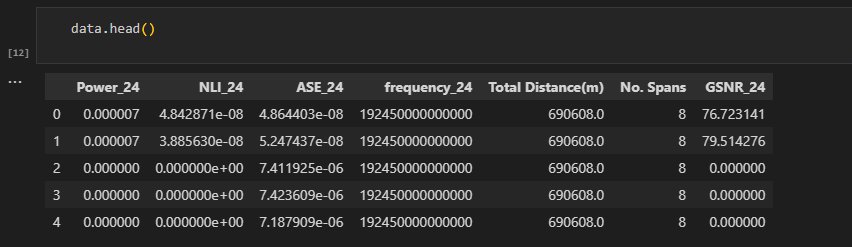
From the above different channel correlation plots, we can conclude that the frequency feature has almost zero significance which means we can discard this feature. But, before we discard any feature, we would like to apply different models and then we will decide whether feature selection is applicable or not.

# **Data Preprocessing**

For our prediction task, we used channel 24 (gsnr\_24) as the target variable.

## **Examine first few rows and column types**

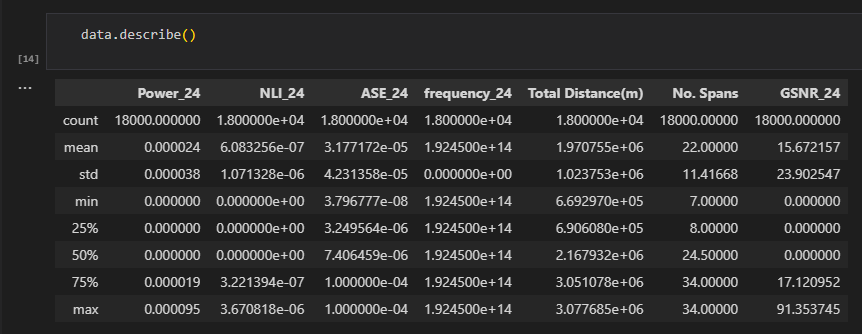
We started by examining the first few rows of the dataset using the data.head() method to get an initial overview of the data. All the columns were float64 except frequency and number of spans that were int.



*Figure 5. First few rows of data*

## **Summary Statistics and look for null values**

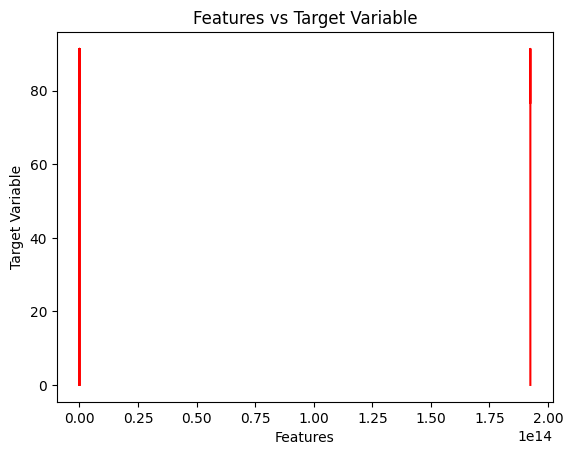
Summary statistics were generated using data.describe(), which provided insights into the central tendency, dispersion, and overall distribution of the data. We also check for null values but there were no null values at all.



*Figure 6. Summary statistics of data*

## **Visualizing gsnr\_24**

We visualize the data using pyplot module from the matplotlib library. The following figure shows that the data is non-linear which means we will be applying only non-linear supervised algorithms for making predictions.

****

*Figure 7. Visual representation of data*

# **Data Splitting**

To evaluate the performance of our predictive models effectively, we split the dataset into training and test sets. This approach helps us to understand how well the models generalize to unseen data.

We used the train\_test\_split function from the scikit-learn library to split the data, with 80% of the data allocated to the training set and 20% to the test set. The random state was set to 42 to ensure reproducibility of the results.

Here are the details of the split:

* **X\_train size**: (14400, 6) - This subset contains 14,400 samples with 6 features each, used to train the models.
* **y\_train size**: (14400,) - This subset contains 14,400 target values corresponding to the training samples.
* **X\_test size**: (3600, 6) - This subset contains 3,600 samples with 6 features each, used to test the models.
* **y\_test size**: (3600,) - This subset contains 3,600 target values corresponding to the test samples.

# **Feature Scaling**

Feature scaling is a crucial preprocessing step in many machine learning workflows. It ensures that all features contribute equally to the model's performance by putting them on a similar scale. This is particularly important for algorithms sensitive to the magnitude of feature values, such as Support Vector Machines, k-Nearest Neighbors, and neural networks.

In our project, we used the StandardScaler from the scikit-learn library to perform feature scaling. The StandardScaler standardizes features by removing the mean and scaling to unit variance, which transforms the data to have a mean of 0 and a standard deviation of 1.

By scaling the features, we enhance the performance of our machine learning models, ensuring they converge faster and perform better by treating all features equally, regardless of their original scale.

# **Model Selection and Training**

In this project, we evaluated several machine learning models to determine which one best predicts the GSNR values. The models selected for comparison included Linear Regression, Support Vector Regression (SVR), Decision Tree, Random Forest, and XGBoost (XGB).

## **Evaluation Results**

After training, the models were evaluated on the test data (X\_test\_scaled and y\_test). The performance of each model was assessed using Mean Squared Error (MSE) and R-squared (R2) metrics. The results were as follows:

|  |  |  |
| --- | --- | --- |
| Model | MSE (Mean Squared Value) | R-squared |
| Linear Regression | 363.289 | 0.352 |
| Support Vector Regression (SVR) | 63.489 | 0.887 |
| Decision Tree | 0.194 | 1.0 |
| Random Forest | 0.074 | 1.0 |
| XGB | 0.058 | 1.0 |

# **Conclusion**

On the basis of the above study, we have concluded the following results:

* Support Vector Regression (SVR) showed a significantly lower MSE and higher R2 compared to Linear Regression, indicating much better performance.
* Decision Tree, Random Forest, and Gradient Boosting (XGB) models all achieved near-perfect performance with extremely low MSE and R2 values of 1.
* Among these models, Gradient Boosting (XGB) had the lowest MSE, making it the best performing model overall.
* Although feature selection could be performed to remove less significant features using a heatmap, the excellent results obtained suggest that there is no immediate need for feature removal.

**THE END**