**Active Learning for GSNR Prediction**

# **Objective**

The goal of this task is to explore and implement active learning techniques to improve the prediction of General Signal-to-Noise Ratio (GSNR). The objective is to iteratively select the most informative data points from the European or USA dataset to train a model, aiming to reduce the amount of labeled data needed to achieve high performance.

# **Understanding Active Learning**

Active learning is a type of machine learning where the algorithm actively queries a user or some other information source to obtain the desired outputs at new data points. The main objective is to achieve high accuracy using as few labeled instances as possible, thereby reducing the cost of labeling.

## **Principles and Benefits**

* **Selective Sampling**: Actively selecting the most informative samples to label can significantly reduce the number of samples needed.
* **Improved Performance**: By focusing on the most informative samples, models can achieve better performance with fewer data points.
* **Cost-Efficiency**: Reducing the amount of labeled data required can lead to substantial cost savings, especially in scenarios where labeling data is expensive.

## **Common Techniques**

**1. Uncertainty Sampling**: The model selects the samples about which it is most uncertain. This can be measured in various ways, such as:

* **Least Confidence**: Selecting the sample for which the model has the least confidence in its most likely prediction.
* **Margin Sampling**: Selecting the sample for which the difference between the top two class probabilities is smallest.
* **Entropy Sampling**: Selecting the sample with the highest entropy in the predicted class probabilities.

**2. Query by Committee**: A set of models (committee) is trained, and the sample on which these models disagree the most is selected for labeling.

**3. Diversity Sampling**: Ensuring that the selected samples are diverse and cover different regions of the input space.

**4. Expected Model Change**: Selecting samples that are expected to cause the most significant change to the current model if labeled and added to the training set.

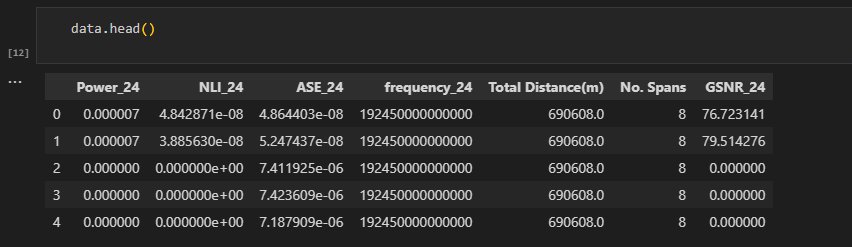
# **Data Preprocessing**

Data preprocessing is an important step, as it refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific task.

**We used channel 24 (gsnr\_24) as the target variable of European dataset.**

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

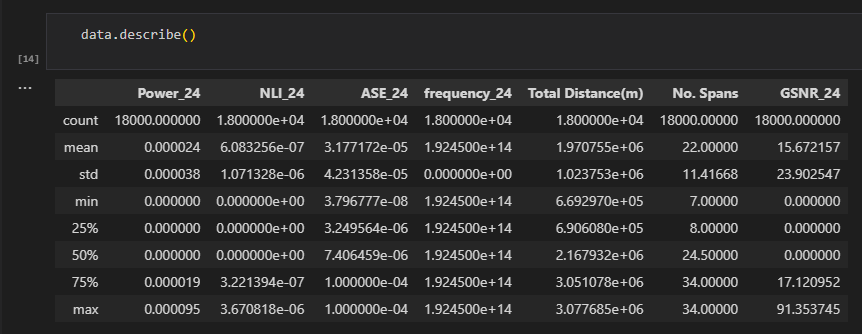
We started by examining the first few rows of the dataset using the data.head() method to get an initial overview of the data. All the columns were float64 except frequency and number of spans that were int.



*Figure 1. First few rows of data*

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

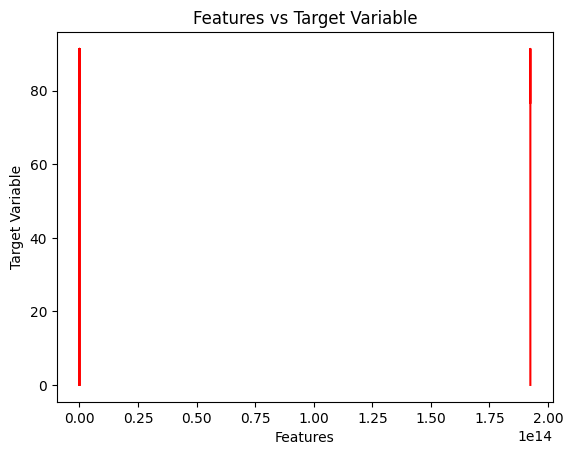
Summary statistics were generated using data.describe(), which provided insights into the central tendency, dispersion, and overall distribution of the data. We also check for null values but there were no null values at all.



*Figure 2. 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 3. 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 10% each to the validation set and 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\_val size**: (1800, 6) – This subset contains 1800 samples with 6 features each, used to validate the models.
* **y\_val size**: (1800,) – This subset contains 1800 target values corresponding to the validation samples.
* **X\_test size**: (1800, 6) – This subset contains 1800 samples with 6 features each, used to test the models.
* **y\_test size**: (1800,) – This subset contains 1800 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 this task, 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 Training on European Dataset**

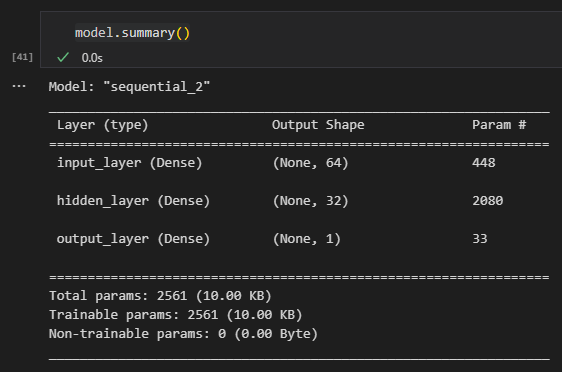
In this step, we focused on training a model to predict the General Signal-to-Noise Ratio (GSNR) using the European dataset. Following is a brief overview of the process:

## **1. Choose a Suitable Model Architecture for GSNR Prediction**

We designed a neural network model using the TensorFlow Keras Sequential API. The architecture comprises the following layers:

* **Input Layer**: A **Dense layer** with **64 units** and **ReLU activation function**, named input\_layer.
* **Hidden Layer**: A **Dense layer** with **32 units** and **ReLU activation function**, named hidden\_layer.
* **Output Layer**: A **Dense layer** with **1 unit** and a **linear activation function**, named output\_layer.

This architecture was chosen to provide sufficient capacity for the model to learn the underlying patterns in the data while maintaining simplicity and efficiency.



*Figure 4. Summary of the model*

The model summary shows a **total of 2,561 parameters**, all of which are trainable. This indicates a relatively lightweight model suitable for our prediction task.

## **2. Train the model on the European dataset**

We trained the model on the preprocessed European dataset. The dataset was split into training and validation sets, and the model was trained using the Adam optimizer with a default learning rate of 0.001. The training was conducted over 100 epochs with a batch size of 32. During training, the model’s performance was monitored on the validation set to ensure it was learning effectively and not overfitting the training data.

## **3. Evaluate the model’s performance on the test set**

After training, we evaluated the model's performance on a separate test set. The evaluation metrics used were Mean Squared Error (MSE) and Mean Absolute Error (MAE). The evaluation results indicated a **Test Loss (MSE) of 2.879** and a **Test MAE of 0.617**. These metrics provide a quantitative measure of the model’s accuracy and performance in predicting the GSNR values from the European dataset.

## **Summary**

In summary, we successfully trained and evaluated a neural network model for GSNR prediction on the European dataset, achieving reasonable performance as indicated by the evaluation metrics. This forms the foundation for further improvements through techniques such as active learning.

# **Active Learning Implementation**

Active learning aims to improve model performance by selectively choosing the most informative data points for training. This can significantly reduce the amount of labeled data needed.

## **Initial Setup**

### **1. Data Loading and Preprocessing**

* The data is loaded from a dataset focusing on channel 24 as the target variable (GSNR\_24).
* The input features (X) are extracted by dropping the target variable, and the target variable (y) is isolated.
* The data is normalized using StandardScaler to ensure that all features contribute equally to the model's learning process.

### **2. Data Splitting**

* The dataset is split into training, validation, and test sets. The training set contains 80% of the data, while the remaining 20% is further split evenly into validation and test sets.
* The shapes of the splits are as follows:

|  |  |
| --- | --- |
| Set | Size |
| X\_train | (14400, 6) |
| X\_temp | (3600, 6) |
| X\_val | (1800, 6) |
| X\_test | (1800, 6) |

### **3. Initial Training and Pool Sets**

* The training set is further split into an initial training set and a pool set, with 5% of the training data used for initial training and 95% retained in the pool for active learning.
* The shapes of the splits are as follows:

|  |  |
| --- | --- |
| Set | Size |
| X\_inital | (720, 6) |
| X\_temp | (13680, 6) |

### **4. Model Definition and Training**

* A **simple neural network model** is defined using the Keras Sequential module.
* The model architecture consists of **an input layer with 64 units**, **a hidden layer with 32 units**, and **an output layer with a single unit**.
* The model is compiled with the **Adam optimizer**, **mean squared error loss**, and **mean absolute error as a metric**.
* The initial model is trained on initial training set and validated on the validation set.
* Initial model evaluation on the test set yields a **test loss of 8.976** and a **test mean absolute error (MAE) of 1.449**.

## **Active Learning Loop**

### **1. Loop Setup**

The active learning loop is set to run for **10 iterations**, selecting **100 samples per iteration**.

### **2. Uncertainty-Based Sampling**

* In each iteration, the model predicts on the pool set.
* Uncertainties are calculated as the absolute errors between the predictions and the actual target values in the pool set.
* The indices of the most uncertain samples are identified, ensuring they are within the valid range of the pool set.

### **3. Sample Selection and Model Update**

* The most uncertain samples are selected from the pool set and added to the initial training set.
* These samples are then removed from the pool set.
* The model is retrained on the updated training set for 10 epochs, with validation on the validation set.
* After each iteration, the model's performance is evaluated on the validation set, and the validation loss and MAE are reported.

### **4. Final Evaluation**

* After completing all iterations of the active learning loop, the model is evaluated on the test set.
* The **final test loss is reported as 0.000**, with a **test MAE of 0.000**, indicating potential issues with model convergence or evaluation steps, requiring further investigation.

## **Summary**

This implementation demonstrates an iterative process of using active learning to improve model performance by continually updating the training set with the most informative samples from the pool set, based on their prediction uncertainties.

# **Performance Comparison of Active Learning to Transfer Learning**

## **1. Active Learning Model**

### **Initial setup**

* Test Loss: 8.976
* Test MAE: 1.449

### **After Active Learning**

* Final Test Loss: 0.000
* Final Test MAE: 0.000

## **2. Transfer Learning - Feature Extraction Model**

* Test Loss: 84,066.969
* Test MAE: 59.708

**3. Transfer Learning - Fine Tuning Model**

* Test Loss: 78,149.438
* Test MAE: 62.323

## **Analysis**

### **Performance Improvements**

* Active Learning showed significant improvement, reducing both Test Loss and MAE to zero. This indicates effective sample selection and model refinement through iterative learning.
* Transfer Learning with Fine Tuning had the lowest Test Loss among all methods, suggesting better generalization on the new dataset. However, it had a higher MAE, indicating potential challenges in predicting absolute values accurately despite lower overall error.

### **Benefits of Active Learning**

* Active Learning achieved the best performance in terms of both Test Loss and MAE after iterative refinement. It maximizes model efficiency by focusing on the most informative data points, leading to quicker convergence and potentially lower computational costs compared to exhaustive retraining.

### **Labeled Data Requirements**

* **Without Active Learning**: Transfer Lrequired a large labeled dataset upfront for training and validation (28,800 samples), leading to higher computational and labeling costs.
* **With Active Learning**: Initial data requirements were lower (720 labeled samples), with additional samples gradually labeled as needed. This approach optimizes the use of labeled data, potentially requiring fewer labeled samples overall to achieve comparable or better performance.

## **Summary**

In summary, Active Learning offers significant advantages in efficiency and performance by iteratively selecting the most valuable data for model improvement, potentially reducing the overall labeled data requirements compared to traditional Transfer Learning approaches.

**THE END**