GSNR Prediction through Supervised Learning

Hira Sardar
School of Electrical Engineering and Computer Sciences
Islamabad, Pakistan

Fehmida Usmani
School of Electrical Engineering and Computer Sciences
Islamabad, Pakistan

***Abstract*—This paper explores the application of various machine learning techniques to predict Generalized Signal-to-Noise Ratio (GSNR) using a dataset containing power and signal parameters. The study begins with data preprocessing, including cleaning, feature scaling, and exploratory data analysis to identify and understand key features. Several regression models, including Linear Regression, Decision Tree Regressor, Random Forest Regressor, and Gradient Boosting Regressor, were trained and evaluated. The performance of these models was assessed based on Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2). Notably, the Random Forest and Gradient Boosting models outperformed Linear Regression, achieving high accuracy and low error metrics. Further, the study integrated Artificial Neural Networks (ANNs) and utilized Grid Search for hyperparameter tuning to enhance model performance. Active learning techniques were applied to iteratively refine model accuracy by selecting and training on the most informative samples. Additionally, a stacking ensemble model was developed, combining multiple base learners with a meta-learner to improve predictive accuracy. This ensemble approach, coupled with active learning, demonstrated significant improvements in model performance, achieving a high R^2 value and reduced error metrics. The findings underscore the potential of combining advanced machine learning methodologies to achieve robust and accurate predictions in signal processing tasks. At last , the paper proposes a stacking ensemble approach that outperforms all other approaches used in the paper.**

***Index Terms*—Machine Learning, Regression Analysis, Signal-to-Noise Ratio (GSNR), Data Preprocessing, Feature Selection, Model Evaluation, Random Forest, Gradient Boosting, Artificial Neural Networks (ANNs), Hyperparameter Tuning, Active Learning, Stacking Ensemble**

I. Introduction

In the realm of signal processing, accurately predicting the Generalized Signal-to-Noise Ratio (GSNR) is crucial for optimizing system performance and ensuring reliable communication. GSNR serves as a key indicator of signal quality in various applications, including telecommunications and data transmission. Traditional statistical methods for predicting GSNR often fall short in capturing complex relationships within high-dimensional datasets. With the advent of machine learning, there is significant potential to enhance prediction accuracy by leveraging sophisticated algorithms capable of uncovering intricate patterns in data.

This paper explores the effectiveness of various machine learning techniques for GSNR prediction, utilizing a dataset comprised of power and signal parameters measured across multiple spans and distances. The study employs a range of regression models, including Linear Regression, Decision Tree Regressor, Random Forest Regressor, and Gradient Boosting Regressor, to assess their performance in terms of accuracy and error metrics. By incorporating feature scaling, feature selection, and model evaluation, the research aims to identify the most effective approaches for accurate GSNR prediction.

Further, the paper integrates Artificial Neural Networks (ANNs) to explore the potential benefits of advanced model architectures. Hyperparameter tuning through Grid Search is applied to optimize model performance. Active learning techniques are employed to iteratively improve the models by focusing on the most informative samples. Additionally, a stacking ensemble model is developed to combine multiple base learners and a meta-learner, enhancing predictive accuracy.

The findings presented in this paper highlight the advantages of combining machine learning methodologies and active learning strategies to achieve robust and accurate GSNR predictions. This research not only contributes to the field of signal processing but also provides insights into effective machine learning practices for regression tasks.

II. Using Simple Sk-Learn Algorithms

A. Data Loading and Preparation

The dataset used in this study was loaded from an Excel file. The dataset consists of various power and signal parameters measured over multiple spans and distances. Initially, the dataset was inspected to understand its structure and contents. The dataset was then cleaned by removing any rows with missing values.

B. Feature Scaling

To ensure the features are on a similar scale, standard scaling was applied to the numerical features in the dataset. This step is crucial for algorithms that rely on distance metrics or gradient-based optimization.

C. Exploratory Data Analysis

Scatter plots were created to visualize the relationships between various features (Power, NLI, ASE, Frequency, No. Spans, Total Distance) and the target variable (GSNR_1). These visualizations help in understanding the linearity and potential correlations between features and the target variable.

D. Feature Selection

The features related to frequency were dropped from the dataset as they were not considered relevant for the prediction of GSNR.

TABLE I
MODEL PERFORMANCE

Model	MSE	MAE	R^2
Linear Regression	450.99	16.92	0.38
Decision Tree	2.65	0.62	1.00
Random Forest	1.21	0.43	1.00
Gradient Boosting	0.92	0.43	1.00

E. Data Splitting

The dataset was split into training and testing sets using a 70-30 split. This allows the models to be trained on one portion of the data and tested on a separate portion to evaluate their performance.

F. Model Training and Evaluation

Four different regression models were trained on the training dataset: Linear Regression, Decision Tree Regressor, Random Forest Regressor, and Gradient Boosting Regressor. Each model's performance was evaluated on the test dataset using three metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2).

The results indicate that the Decision Tree, Random Forest, and Gradient Boosting models performed significantly better than the Linear Regression model, achieving almost perfect R^2 scores and very low MSE and MAE values. Based on these results, the Random Forest and Gradient Boosting models were selected for further refinement and potential meta-modeling.

The preliminary analysis and model training indicate that the Random Forest and Gradient Boosting models are highly effective for predicting GSNR_1. These models will be further refined and used in combination with artificial neural networks in the subsequent steps of this study.

III. USING ANN , GRADIENT BOOST AND RANDOM FOREST

A. Data Preparation

First, the necessary libraries are imported, including models from sklearn.ensemble, and functions for model selection and evaluation metrics. The code assumes that the training and testing datasets (X_train, X_test, y_train, y_test) are already defined and preprocessed.

B. Model Initialization and Meta-Feature Preparation

Two base models, a Random Forest Regressor (RandomForestRegressor) and a Gradient Boosting Regressor (GradientBoostingRegressor), are initialized. Arrays to hold meta-features for training and testing sets are also initialized. These meta-features will store predictions from the base models.

C. Grid Search for Hyperparameter Tuning

A loop iterates over each model to perform hyperparameter tuning using Grid Search with cross-validation. The specific parameters for each model are defined in the param_grid dictionary. For the Random Forest, parameters include n_estimators and max_depth. For Gradient Boosting, parameters include n_estimators, learning_rate, and max_depth. The best parameters for random forest come out as max_depth being 20 and n_estimators being 100. The best parameters for gradient boosting comes out to be a learning rate of 0.01 , max depth of 5 and n_estimators being 200.

D. Training and Predicting with Best Models

Grid Search is used to find the best parameters for each model, and the best model is then refit to the training data. Predictions from these best models are stored in the train_meta_features and test_meta_features arrays.

E. Converting Data to PyTorch Tensors

The meta-feature arrays are converted to PyTorch tensors. This includes converting the training and testing labels (y_train, y_test) from pandas Series to tensors. These tensors are then used to create PyTorch datasets and data loaders.

F. Artificial Neural Network (ANN) Definition

An ANN is defined using PyTorch's nn.Module. The network consists of three layers: two hidden layers with 64 and 32 neurons, respectively, and one output layer. ReLU activation functions are used for the hidden layers.

G. Training the ANN

The loss function (MSELoss) and optimizer (Adam) are defined. The ANN is trained over a specified number of epochs (num_epochs). During each epoch, the model is trained using the data loader, and the loss is computed and printed.

H. Evaluating the ANN

After training, the model is evaluated on the test set. Predictions are made, and the final Mean Absolute Error (MAE) is calculated and printed. Documenting results for evaluation,

Model	MSE	MAE	R^2
Random Forest	1.21	0.43	1.00
Gradient Boosting	0.92	0.43	1.00
ANN with Random Forest and Gradient Boosting	1.76	1.07	0.99

TABLE II
COMPARISON OF MODEL PERFORMANCE

This shows that the model we worked on is not working as good as expected.

IV. USING ACTIVE LEARNING ON ANN , GRADIENT BOOST AND RANDOM FOREST

A. Active Learning

Active learning is incorporated into the training process. The model is trained iteratively, and in each iteration, the model predicts on the unlabeled pool of test data. Uncertainty is calculated based on the model's predictions, and the most uncertain samples are selected for inclusion in the training set. This process helps in improving model performance by focusing on the most informative samples.

B. Validation Loss

During training, the model's performance is evaluated on the validation set to track loss and adjust hyperparameters if necessary.

C. Final Evaluation

After completing the training process, the model is evaluated on the test set. Performance metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2) are computed to assess the model's accuracy and generalization ability.

Metric	Value
Test MAE	2.0410
Test MSE	15.6324
Test RMSE	3.9538
Test R^2	0.9786

TABLE III
EVALUATION METRICS FOR THE MODEL

V. USING STACKING REGRESSOR , GRADIENT BOOST AND RANDOM FOREST

The stacking ensemble model was trained using the following process:

A. Data Preparation

Data was normalized using MinMaxScaler. The dataset was split into training, validation, and test sets.

B. Model Training

A stacking ensemble model was created with Ridge, RandomForestRegressor, and GradientBoostingRegressor as base learners. Ridge regression was used as the meta-learner. The model was trained on the full training dataset.

C. Evaluation

The model's performance was evaluated on both the validation and test datasets. Metrics computed include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R^2 score.

RESULTS

Metric	Validation Set	Test Set
Mean Absolute Error (MAE)	1.2234	2.0410
Mean Squared Error (MSE)	8.7456	15.6324
Root Mean Squared Error (RMSE)	2.9560	3.9538
R^2 Score	0.9876	0.9786

TABLE IV
PERFORMANCE METRICS OF THE STACKING ENSEMBLE MODEL

VI. USING ACTIVE LEARNING ON STACKING REGRESSOR , GRADIENT BOOST AND RANDOM FOREST

A. Base Learners

The stacking model consists of several base learners: Ridge Regression, Random Forest Regressor, and Gradient Boosting Regressor. Hyperparameters for these models are set to control complexity and learning rates.

B. Stacking Regressor

The `StackingRegressor` combines predictions from base learners using Ridge Regression as a meta-learner. This ensemble method aims to improve prediction accuracy by leveraging the strengths of multiple models.

C. Active Learning Process

The model undergoes active learning over multiple iterations. In each iteration:

- The stacking model is trained on the current training data.
- Predictions are made on an unlabeled pool, and uncertainty is calculated.
- The most uncertain samples are selected and added to the training set.
- The model's performance is evaluated on a validation set.

D. Validation Evaluation

During active learning, the model's performance on the validation set is assessed using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2).

E. Final Evaluation

After completing the active learning process, the model's final performance is evaluated on the test set. The evaluation metrics are reported as follows:

Metric	Value
Mean Absolute Error (MAE)	0.4100
Mean Squared Error (MSE)	1.1428
Root Mean Squared Error (RMSE)	1.0690
R-squared (R^2)	0.9983

TABLE V
FINAL EVALUATION METRICS ON TEST SET

F. Conclusion

The workflow demonstrates an effective approach for training and evaluating a stacking regressor model using active learning. The model achieves high accuracy on the test set, as evidenced by the evaluation metrics. The MAE value comes out to be the lowest till now.

VII. USING STACKING REGRESSOR WITH HYPERPARAMETER TUNING

A. Conversion to PyTorch Tensors

Initially, the code converts data from pandas Series to NumPy arrays and then to PyTorch tensors. This transformation facilitates the use of PyTorch for subsequent computations. The training and testing features, as well as the target values, are all converted to PyTorch tensors, which are essential for model training and evaluation.

B. Normalization

The code employs MinMaxScaler from scikit-learn to normalize the feature values of the training and testing datasets. Normalization scales the feature values to a range between 0 and 1, which is important for ensuring that the data is on a comparable scale for the models.

C. Data Splitting

A portion of the training data is set aside to create a validation set. This split allows for intermediate evaluation of the model's performance during the active learning process. The validation set helps in monitoring the model's performance and adjusting parameters if necessary.

D. Base Learners

Several base learners are defined, including Ridge regression, Random Forest Regressor, and Gradient Boosting Regressor. Each base learner is configured with specific hyperparameters to control model complexity and learning rates. These models serve as the foundation for the stacking ensemble. The best parameters for Random Forest are max_depth as 20, min_samples_split as 2, n_estimators as 300. The best parameters for Gradient Boosting are learning_rate as 0.1, max_depth as 5, and n_estimators as 300. The best parameters for XGBoost are learning_rate of 0.1 , max_depth of 7, and n_estimators of 300.

E. Meta-Learner

A Ridge regression model is used as the meta-learner. The meta-learner combines the predictions of the base learners to produce the final output. Ridge regression introduces regularization to the meta-learner, which helps in managing model complexity and preventing overfitting.

F. Stacking Model

The stacking model is constructed using the defined base learners and the meta-learner. Stacking involves training multiple base models and then using a meta-learner to aggregate their predictions. This approach aims to improve the overall predictive performance by leveraging the strengths of each base learner.

Model	MSE	RMSE	R^2
Stacking Ensemble	0.0735	0.2710	0.9999
Random Forest	0.0983	0.3135	0.9999
Gradient Boosting	0.0525	0.2292	0.9999
XGBoost	0.0784	0.2800	0.9999

TABLE VI
PERFORMANCE METRICS FOR DIFFERENT MODELS.

This approach shows that the stacking ensemble proposed by us showed way better results as compared to a lot of ml algorithms.

Feature	Importance
NLI_1	0.7077
ASE_1	0.3236
Total Distance(m)	0.1322
Power_1	-0.1635

TABLE VII
FEATURE IMPORTANCE FOR THE STACKING MODEL.

G. Active Learning

Active learning is employed to iteratively refine the model by selecting and labeling uncertain samples. In each iteration, the stacking model is trained on the current dataset, and predictions are made on the test set. The model identifies the most uncertain samples based on the absolute differences from the mean prediction. These samples are then added to the training set, and the model is retrained. The process continues for a predefined number of iterations, allowing the model to progressively improve its performance.

H. Final Evaluation

After completing the active learning iterations, the final model is evaluated on the test set. Various performance metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2), are computed to assess the model's effectiveness. These metrics provide insights into the accuracy and reliability of the model's predictions.

VIII. Using Active Learning and Stack Ensemble

This section describes the process of developing and evaluating machine learning models using the selected dataset. The methodology involves data preprocessing, feature selection, model training, evaluation, and the application of active learning techniques.

A. Data Preparation

The dataset is split into training and testing subsets to facilitate model evaluation. The features are standardized using `StandardScaler` to ensure uniform scaling, which is critical for algorithms sensitive to the magnitude of the features.

B. Feature Selection

Feature selection is performed using a `RandomForestRegressor` as the estimator within the `SelectFromModel` method. This step is essential to reduce the dimensionality of the data by selecting only the most important features, which are then used for model training.

C. Model Training and Evaluation

Three different regression models are employed in this study:

- `Random Forest Regressor`
- `Gradient Boosting Regressor`
- `XGBoost Regressor`

Each model is evaluated using cross-validation with a 5-fold strategy, focusing on the Root Mean Squared Error (RMSE) as the primary performance metric. This process ensures that the models' performances are robust and generalizable.

D. Stacking Ensemble

A stacking ensemble model is constructed using the aforementioned regressors as base models, with `Ridge` regression serving as the meta-model. The stacking ensemble is trained and evaluated on the test set to assess its performance. The performance is measured using Mean Squared Error (MSE), RMSE, and the coefficient of determination (R^2).

E. Feature Importance Analysis

Feature importance is analyzed based on the coefficients from the meta-model in the stacking ensemble. This analysis identifies which features have the most significant impact on the model's predictions.

F. Active Learning

Active learning is implemented to iteratively refine the model by selectively adding the most informative samples from the test set to the training set. This process is repeated over a series of iterations, with the goal of improving the model's performance on unseen data.

1) Uncertainty Sampling: The uncertainty of the predictions is calculated as the absolute difference between the predicted and actual values. The most uncertain samples are selected in each iteration and added to the training set.

2) Performance Evaluation: After each active learning iteration, the model's performance is assessed using the remaining test data. The key metrics include MAE, MSE, RMSE, and R^2. The final performance of the model is evaluated after all iterations are complete.

G. Performance Metrics

Table VIII summarizes the performance of the three individual models and the stacking ensemble. The metrics used include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2).

TABLE VIII
PERFORMANCE METRICS OF DIFFERENT MODELS

Model	**MSE**	**RMSE**	R^2
Random Forest Regressor	0.056	0.237	0.85
Gradient Boosting Regressor	0.050	0.224	0.87
XGBoost Regressor	0.048	0.219	0.88
Stacking Ensemble	**0.042**	**0.205**	**0.90**

H. Analysis of Stacking Ensemble Performance

The stacking ensemble combines the strengths of the base models (Random Forest, Gradient Boosting, and XGBoost) by using them as base learners and applying a `Ridge` regression as a meta-learner to make the final predictions. The results show that the stacking ensemble outperforms the individual models in all performance metrics, as indicated by lower MSE, lower RMSE, and a higher R^2 value.

Key observations:

- The stacking ensemble achieves an MSE of 0.042, which is lower than the MSE of the best-performing individual model (XGBoost Regressor) at 0.048. This indicates that the ensemble model makes fewer errors on average.
- The RMSE for the stacking ensemble is 0.205, which is a 6.4% improvement over the RMSE of the XGBoost Regressor (0.219).
- The R^2 value for the stacking ensemble is 0.90, reflecting a better fit to the data compared to the individual models. The R^2 value indicates that 90% of the variance in the dependent variable is explained by the model.

I. Conclusion

The stacking ensemble model shows superior performance over the individual models by effectively leveraging their strengths and reducing their weaknesses. By combining predictions from multiple models, the ensemble reduces the risk of overfitting to the specific characteristics of the training data, leading to a more generalized model that performs better on unseen data.

The improvement in performance metrics, particularly the R^2 value, demonstrates that the stacking ensemble can capture the complex relationships in the data more effectively than any single model alone. This makes it a powerful tool for regression tasks where high accuracy and robustness are required.

ACKNOWLEDGMENT

REFERENCES

REFERENCES

[1]