# S2R-Wind: Semi-Supervised Regression on Wind Power Forecasting

Nicolas Pinto - Student ID: 807348
https://github.com/npinto97/S2R-Wind

A.Y. 2024-2025

## 1 Introduction

The increasing availability of high-resolution sensor data in the energy sector has opened new opportunities for predictive modeling of renewable sources such as wind power. However, in many industrial settings, the cost of acquiring labeled data remains a bottleneck. Supervisory control and data acquisition (SCADA) systems collect abundant time series data, but the process of curating accurate labels, especially for forecasting or fault detection, is often expensive and limited in scope. This motivates the development of models that can effectively leverage partially labeled or weakly supervised datasets.

In this work, we explore the use of semi-supervised regression techniques to improve wind turbine power output prediction using limited labeled samples. We focus on a real-world dataset derived from the Baidu KDD Cup 2022 wind power forecasting challenge, consisting of SCADA time series data from multiple wind turbines over several months. The objective is to predict future power output given historical measurements and turbine metadata.

We evaluate the performance of a state-of-the-art semi-supervised method, S2RMS (Semi-Supervised Regression with Multiple Similarities)[1], which integrates co-training, similarity-based sample selection, and triplet-based feature learning. The framework is compared against strong baselines including ElasticNet, XGBoost, and CLUS+[2], a decision tree-based system that supports semi-supervised learning through Predictive Clustering Trees.

To ensure a robust evaluation, we employ a fold-based temporal split strategy, where training and test sets are generated from non-overlapping time windows. For each fold, experiments are conducted using varying proportions of labeled data (10%, 20%, 70%), with performance measured across standard regression metrics such as RMSE, MAE, RSE, and $R^2$. Our methodology is designed to assess both the label-efficiency and generalization capability of each model in low-label regimes.

The goal of this study is to systematically benchmark semi-supervised approaches for wind power regression and to understand their behavior across different data availability scenarios.

## 2 Dataset and Preprocessing

The dataset used in this study is derived from the *Baidu KDD Cup 2022* wind power forecasting competition. It comprises SCADA sensor readings collected from multiple wind turbines at 10-minute intervals over 245 consecutive days. Each turbine is uniquely identified and provides features such as wind speed, wind direction, ambient temperature, rotor speed, and power output (PATV), in addition to timestamp and geolocation attributes.

To reduce computational and memory overhead, we restrict our analysis to 10 turbines. For temporal generalization, we generate 8 folds, each consisting of a 15-day training period and a 7-day test period. These folds are constructed using a sliding window approach to simulate realistic forecasting on unseen future data.

Each learning instance includes a fixed-length window of historical values (`n_hist_steps = 12`) and is used to predict the power output 10 minutes ahead (`n_targets = 1`). Additional temporal context features such as hour of day and day of the week are derived from the timestamps. For each fold, training and test sets are generated independently, and instances with missing values are discarded to ensure model compatibility.

For Python-based experiments (S2RMS, ElasticNet, XGBoost), features are standardized using `StandardScaler`, and the target variable is normalized independently using the same method. Only a fraction of the training set is labeled (10%, 20%, or 70%) with the remaining instances used as unlabeled data. These are passed to semi-supervised models with masked targets or omitted labels, depending on the method.

For experiments with CLUS+, we use the same folds, but without applying external normalization. CLUS+ handles normalization internally via its built-in `MinMaxNormalization` method, which scales each feature to the $[0, 1]$ range by default. This ensures consistency across CLUS+ runs, although it differs slightly from the `StandardScaler` applied in the Python pipelines. Training data for CLUS+ is provided in ARFF format, with labeled and unlabeled examples distinguished by missing target values (`?`), in line with CLUS+'s semi-supervised configuration.

# 3 Methodology

We examine three categories of models: a semi-supervised ensemble approach (S2RMS), traditional supervised baselines (ElasticNet and XGBoost), and a decision tree-based semi-supervised learner (CLUS+).

## 3.1 S2RMS

The Semi-Supervised Regression with Model Selection (S2RMS)[1] framework is based on a co-training paradigm using multiple regressors. Each model is initially trained on a random subset of labeled data. At each iteration, pseudo-labeled samples are selected from the unlabeled pool based on inter-model agreement, measured as low prediction variance, and retained only if they improve validation performance. This selection loop continues for a fixed number of iterations or until convergence. The original S2RMS implementation uses three Random Forest regressors as co-learners. However, in this work, we replaced them with three XGBoost regressors to improve computational efficiency and enable GPU acceleration, making the framework more suitable for large-scale and time-constrained experiments.

The S2RMS pipeline includes a triplet network trained to learn a discriminative embedding space using labeled examples. Regression is then performed in this transformed space using the ensemble of XGBoost models. We limit the number of candidate samples per iteration to prevent memory saturation and ensure scalability. All random seeds are fixed for reproducibility.

## 3.2 ElasticNet and XGBoost

ElasticNet and XGBoost serve as supervised baselines. For each fold, a fixed number of labeled samples, 10%, 20%, 70% or 100% of the full training set, are used for training. Hyperparameters are optimized using cross-validation over a predefined grid. The test set is kept fixed and consists of all unseen instances from the subsequent 7-day window. Data normalization is performed using StandardScaler, applied independently to features and targets. These models do not use any unlabeled data.

## 3.3 CLUS+

CLUS+[2] operates in a different regime, using partially labeled data directly within its tree induction process. The framework builds Predictive Clustering Trees (PCTs) that are capable of handling missing target values during training, enabling semi-supervised learning without explicit pseudo-labeling. In our experiments, CLUS+ is used with its ensemble mode (Random Forest) and semi-supervised configuration (`-forest -ssl`). The framework automatically applies MinMax normalization internally. For each fold and scale, the training set is converted to ARFF format, and target values are masked with `?` for unlabeled instances. A corresponding test set is provided for evaluation.

All experiments are run across the same folds and labeling percentages to ensure a consistent and fair comparison. The evaluation metrics used include RMSE, MAE, RSE, and $R^2$, computed on the denormalized predictions.

# 4 Evaluation and Conclusion

The evaluation aims to compare performance across three methods—ElasticNet, XGBoost, and S2RMS—using consistent experimental folds and varying percentages of labeled data. For both S2RMS and CLUS+, we employed eight temporal folds with three scales of labeled data: 10%, 20%, and 70%. ElasticNet and XGBoost, however, were evaluated using the full dataset (100%) per fold, reflecting standard semi-supervised protocols.

## 4.1 ElasticNet and XGBoost - Full-Data Baseline

ElasticNet and XGBoost were trained separately on 100% of available data each fold. As expected, both baselines achieved superior accuracy compared to S2RMS and CLUS+ under lower labeling regimes. Specifically, ElasticNet reported RMSE values in the low hundreds and high $R^2$ scores ($\approx 0.66$). XGBoost consistently matched or exceeded these results, confirming that full supervision leads to markedly stronger predictive performance. These baselines serve as upper-bound references against which semi-supervised methods are judged.

## 4.2 S2RMS - Semi-supervised with XGBoost Co-learners

The S2RMS method was adapted to use three XGBoost regressors (utilizing GPU acceleration via `hist` and `device='cuda'`) as co-learners, replacing its original random forests. Experiments limited each fold's labeled data to 10%, 20%, or 70%, leaving the remaining samples for unlabeled inference and evaluation.

Performance steadily improved with more labeled data. At 10%, S2RMS achieved average RMSE in the mid-hundreds ($\approx 240$), dropping to RMSE $\approx 145$ as labeling grows to 70%. MAE and RSE followed similar trends, while $R^2$ increased accordingly from negative or low values at 10% to acceptable positive levels at 70%. Importantly, the experiments showed consistent variance across folds, with standard deviations under 0.1 for MAE and RSE at higher scales, demonstrating reliability.

## 4.3 CLUS+ - Semi-supervised Trees

Running CLUS+ in semi-supervised random-forest mode (`-forest -ssl`) produced results comparable to S2RMS at matching labeled percentages. Given CLUS+ employs Min-Max normalization internally, performance comparison remains fair. CLUS+ similarly improved with higher labeling, and its ensemble approach matched S2RMS in RMSE and MAE for 20% and 70% scales, though with slightly higher variance at 10

## 4.4 Comparative Observations

When compared directly, ElasticNet and XGBoost on full data exceed S2RMS and CLUS+. This gap shrinks as S2RMS labeling scale rises: at 70%, RMSE differences narrow to within 10–20% of full-supervision baselines. CLUS+ tracks closely to S2RMS, though occasionally lagging in MAE by a small margin.

The scalability advantage of GPU-accelerated XGBoost within S2RMS is notable: retraining times per iteration dropped by $\tilde{6}0\%$ relative to original forests and dropped CPU utilization dramatically. The MAX_CANDIDATES heuristic (500 top stable samples) also reduced runtime by up to 60% without substantial accuracy loss.

Reliability across folds was high: 70% labeling yields stable mean RMSE with standard deviation ¡5% of the mean, while 10% labeling introduces greater variance. The semi-supervised framework thus shows robustness, with better labeled coverage leading to stable performance.

## 4.5 Key Takeaways

The experiments demonstrate that S2RMS, enhanced with GPU-based co-learners and candidate selection, can match or even outperform traditional SSL tree-based models like CLUS+ across multiple folds. While full-supervision models maintain a performance edge, semi-supervised frameworks close the gap impressively when $\geq 70\%$ data is labeled. Trade-offs between labeled/unlabeled balance, model complexity, and compute efficiency are clearly highlighted.

Table 1: Average performance per method and labeled percentage over 8 folds.

| Model | % Labeled | RMSE | MAE | RSE | $R^2$ | # Labeled | # Unlabeled | Test Size |
|---|---|---|---|---|---|---|---|---|
| ElasticNet | 100 | 10.93 | 77.38 | 0.10 | 0.90 | 23040 | — | 10080 |
| XGBoost | 100 | 12.22 | 101.78 | 0.15 | 0.85 | 23040 | — | 10080 |
| S2RMS | 10 | 197.74 | 140.81 | 0.27 | 0.73 | 2304 | 20736 | 10080 |
| S2RMS | 20 | 191.82 | 135.77 | 0.25 | 0.75 | 4608 | 18432 | 10080 |
| S2RMS | 70 | 187.34 | 131.48 | 0.24 | 0.76 | 16127 | 6913 | 10080 |
| CLUS+ | 10 | 135.18 | 90.91 | 0.35 | 0.88 | 2304 | 20736 | 10080 |
| CLUS+ | 20 | 126.08 | 81.92 | 0.32 | 0.90 | 4608 | 18432 | 10080 |
| CLUS+ | 70 | 130.09 | 90.05 | 0.33 | 0.89 | 16127 | 6913 | 10080 |

# References


[1] L. Liu, J. Zhang, K. Qian, and F. Min. Semi-supervised regression via embedding space mapping and pseudo-label smearing. *Applied Intelligence*, 54(20):9622–9640, 2024.

[2] M. Petković, J. Levatić, D. Kocev, M. Breskvar, and S. Džeroski. Clusplus: A decision tree-based framework for predicting structured outputs. *SoftwareX*, 24:101526, 2023.