

Forecasting Electric consumption & WNN package

Implementation

Time Series project Report

Olivier BOROT

January 2025

Abstract

This report presents the work completed as part of the Time Series Forecasting class of the Master 2 SISE of Lyon 2. The document is organized in two distinct parts : The first part details the implementation of an R package for a non-parametric forecasting method called Weighted Nearest Neighbors (WNN), inspired by the work of Talavera-Llames et al. The second part describes the methodology used to select the best algorithm for predicting electricity consumption, where the WNN algorithm is applied alongside various forecasting methods including Holt-Winters, SARIMA, XGBoost, and SVM. The WNN algorithm achieved an impressive RMSE of 15.51, ranking 3rd among all models tested. The final selected model was SARIMA(5,0,0)(0,1,1), achieving an RMSE of 14.81 on the validation set.

All source code, datasets, tutorials, and exploratory notebooks are available in a fully documented GitHub repository: https://github.com/OlivierBOROT/SISE_WNN_implementation.

Part I

Weighted Nearest Neighbors (WNN) Implementation

1 Introduction

The **Weighted Nearest Neighbors** (WNN) algorithm is a forecasting method for univariate time series based on nearest neighbor search. Proposed by Talavera-Llames et al. [1], this algorithm relies on the principle that past sequences similar to the current observation window can provide relevant information for future forecasting.

This part presents the implementation of a complete R package for the WNN algorithm, designed to be memory-efficient and fully compatible with R's time series ecosystem.

2 Algorithm Overview

2.1 Mathematical Formulation

Let $\mathbf{y} = (y_1, y_2, \dots, y_n)^\top$ denote a univariate time series of length n . We define the following notation:

- $\mathbf{x}_{[a:b]} = (y_a, y_{a+1}, \dots, y_b)^\top$: a subsequence (segment) of \mathbf{y}
- $w \in \mathbb{N}^*$: the lookback window size
- $k \in \mathbb{N}^*$: the number of nearest neighbors ($k \ll n$)
- $h \in \mathbb{N}^*$: the forecast horizon

2.2 Operating Principle

The WNN algorithm proceeds through the following steps:

1. **Reference window extraction:** Extract the most recent w observations as the reference segment:

$$\mathbf{x}_{\text{ref}} = \mathbf{x}_{[n-w+1:n]} \in \mathbb{R}^w \quad (1)$$

2. **Historical segment enumeration:** For each valid starting index $j \in \{1, 2, \dots, n - w - h + 1\}$, extract the historical segment and its associated future values:

$$\mathbf{x}_j = \mathbf{x}_{[j:j+w-1]} \in \mathbb{R}^w \quad (\text{candidate segment}) \quad (2)$$

$$\mathbf{f}_j = \mathbf{x}_{[j+w:j+w+h-1]} \in \mathbb{R}^h \quad (\text{future horizon}) \quad (3)$$

3. **Distance computation:** Compute the Euclidean distance between the reference and each candidate:

$$d_j = \|\mathbf{x}_{\text{ref}} - \mathbf{x}_j\|_2 = \sqrt{\sum_{t=1}^w (x_{\text{ref},t} - x_{j,t})^2} \quad (4)$$

4. **k -nearest neighbor selection:** Identify the set \mathcal{N}_k of indices corresponding to the k smallest distances.

5. **Inverse-square weighting:** Assign weights inversely proportional to squared distances:

$$\omega_j = \frac{1}{(d_j + \varepsilon)^2}, \quad j \in \mathcal{N}_k \quad (5)$$

where $\varepsilon = 10^{-10}$ ensures numerical stability for near-zero distances.

6. **Weight normalization:** Normalize weights to form a proper probability distribution:

$$\tilde{\omega}_j = \frac{\omega_j}{\sum_{\ell \in \mathcal{N}_k} \omega_\ell}, \quad \text{such that } \sum_{j \in \mathcal{N}_k} \tilde{\omega}_j = 1 \quad (6)$$

7. **Weighted forecast aggregation:** The final forecast is the weighted average of future horizons:

$$\hat{\mathbf{y}}_{[n+1:n+h]} = \sum_{j \in \mathcal{N}_k} \tilde{\omega}_j \cdot \mathbf{f}_j \quad (7)$$

2.3 Algorithm Pseudocode

Algorithm 1 Weighted Nearest Neighbors (WNN) Forecasting

Require: Time series $\mathbf{y} = [y_1, \dots, y_n]$, window size w , neighbors k , horizon h

Ensure: Forecast $\hat{\mathbf{y}} = [\hat{y}_{n+1}, \dots, \hat{y}_{n+h}]$

```

1:  $CC_i \leftarrow [y_{n-w+1}, \dots, y_n]$  ▷ Extract reference window
2: Initialize top_k_distances  $\leftarrow [\infty, \dots, \infty]$  (size  $k$ )
3: Initialize top_k_futures  $\leftarrow \emptyset$ 
4: for  $j = 1$  to  $n - w - h + 1$  do
5:    $CC_j \leftarrow [y_j, \dots, y_{j+w-1}]$  ▷ Historical window
6:   future $j$   $\leftarrow [y_{j+w}, \dots, y_{j+w+h-1}]$  ▷ Values after window
7:    $d_j \leftarrow \|CC_i - CC_j\|_2$  ▷ Euclidean distance
8:   if  $d_j < \max(\text{top\_k\_distances})$  then
9:     Replace largest distance with  $d_j$  and store future $j$ 
10:  end if
11: end for
12:  $\alpha_j \leftarrow 1/(d_j + \epsilon)^2$  for all  $j \in \{1, \dots, k\}$ 
13:  $\alpha \leftarrow \alpha / \sum_{j=1}^k \alpha_j$  ▷ Normalize weights
14:  $\hat{\mathbf{y}} \leftarrow \sum_{j=1}^k \alpha_j \cdot \text{future}_j$  ▷ Weighted average
    return  $\hat{\mathbf{y}}$ 

```

3 R6 Implementation

3.1 Design Choices

The R6 implementation employs a selection of the segments for memory optimization. Rather than materializing all $(n - w - h + 1)$ candidate segments simultaneously, the algorithm maintains only the k best neighbors encountered during a single pass through the time series.

The implementation has been thought to be fully compatible with R's native **ts** (time series) objects as well as the **forecast** package, allowing for easy comparison with standard forecasting methods from the **fpp2** framework.

3.2 Hyperparameters

The model hyperparameters are:

- `horizon` (h): number of future values to predict
- `window` (w): size of the comparison window
- `k`: number of nearest neighbors retained

3.3 Usage

The model usage is straightforward with the R6 syntax and accepts `ts` objects as input:

```
wnn <- WNN$new(horizon = 12, window = 24, k = 5)
forecast <- wnn$fit_predict(tsdata)
```

The output is returned as a `ts` object, to maintain temporal consistency with the input series and be able to use it directly with visualization and evaluation tools from the `forecast` package.

Part II

Electricity Consumption Forecast

4 Introduction

The objective of this part is to forecast electricity consumption (in kW) for one full day (96 observations at 15-minute intervals) on February 19, 2010. The training data spans from January 1, 2010 to February 18, 2010, with the outdoor temperature available as an exogenous variable.

5 Data Description

5.1 Data Characteristics

- **Frequency:** 96 observations per day (15-minute intervals)
- **Seasonality:** Strong daily pattern (period = 96), with no particular evolution in variance
- **Trend:** Almost no trend observed
- **Data problems:** Some zero values (like measurement errors or power supply issues) were treated as missing data and interpolated using seasonal interpolation via `na.interp()` from the `forecast` package

5.2 Data Preparation

The dataset contains electricity consumption data from January 1, 2010 (1:15 AM) to February 18, 2010 (11:45 PM), with the outdoor air temperature available including for February 19, 2010.

The raw data was processed as follows:

1. Conversion to time series objects with frequency 96 (96 quarters per day)
2. Replacement of zero values with NA (likely measurement errors or power supply issues)
3. Seasonal interpolation using `na.interp()` from the `forecast` package
4. Train/validation split: last day (96 points) reserved for validation

6 Methodology

6.1 Forecasting Strategy

The forecasting methodology follows these principles:

- **Test set:** 96 data points (1 day) were selected as the test set to reflect the final prediction goal.
- **Cross-validation:** Cross-validation on the training set was implemented for algorithms that can be easily iterated to tune hyperparameters. When implemented, cross-validation used a rolling expanding window approach.
- **Performance metrics:** RMSE (Root Mean Squared Error) was used as the main metric to differentiate the different models, although other metrics have been calculated to have a better understanding of the strengths and weakness of the models.

6.2 Stationarity Analysis

The Ljung-Box test showed significant correlation in the series ($p\text{-value} = 2.2e-16$), confirming there was signal to predict. The analysis indicated:

- Regular differences (d): 0 (no trend)
- Seasonal differences (D): 1 (to eliminate seasonal effect)

7 Models Evaluated

7.1 Holt-Winters Methods

Due to high frequency data, the `stats` package was used instead of `forecast`. Two Holt-Winters methods were tested without damping:

- Additive seasonality
- Multiplicative seasonality

Both models were cross-validated over 4 days of the training set.

Table 1: Holt-Winters Model Comparison

Model	RMSE on test	Mean RMSE on CV
Holt-Winters Additive	18.80	26.65
Holt-Winters Multiplicative	16.00	27.18

Both models showed similar results on the test set as well as the cross-validation score. The multiplicative model should be more robust due to his better score on test and a lesser gap on the RMSE in cross-validation.

7.2 SARIMA Models

The daily seasonality pattern is well suited to SARIMA models. Following Box-Jenkins methodology:

1. Applied seasonal differencing (lag 96) to remove seasonality
2. Analyzed ACF/PACF of differenced series
3. ACF shows significant spike at lag 96 (seasonal MA component)
4. PACF shows significant spikes at early lags (AR component up to lag 5)

Note: SARIMA models on high-frequency datasets are computationally intensive and took well over multiple minutes for each models to be done. So, the cross-validation would have been very expensive both in time and resources; they have not been used for these models.

Table 2: SARIMA Models Comparison

Model	Parameters	RMSE on test	AIC
SARIMA (0,0,0)(0,1,1)	Seasonal MA only	15.89	35821.28
SARIMA (5,0,0)(0,1,1)	AR(5) + Seasonal MA	14.81	31923.24
SARIMA (5,0,1)(0,1,1)	ARMA(5,1) + Seasonal MA	14.93	31920.32
Auto SARIMA	ARIMA(5,0,0)(0,1,0)	22.65	34372.94

Remark: The manual SARIMA(5,0,0)(0,1,1) significantly outperformed the automatic selection, demonstrating the value of careful ACF/PACF analysis following Box-Jenkins methodology.

7.3 Machine Learning Models

7.3.1 Feature Selection

Based on PACF analysis, the following lags were identified as significant:

- Lags 1, 37, 61: Short-term dependencies
- Lags 96, 97: Daily seasonality
- Lag 192: Two-day pattern

Temperature was included as an additional feature for multivariate models.

7.3.2 XGBoost

Feature matrices were built by concatenating lagged values (from $t-1$ up to $t-384$ for 4 periods of seasonality) with temperature. Predictions were made recursively.

Hyperparameter grid search was performed over:

- **nrounds:** 50, 100, 200
- **max_depth:** 1, 3, 6, 9
- **eta:** 0.01, 0.1, 0.25, 0.3, 0.4, 0.5

Best hyperparameters: nrounds = 50, max_depth = 3, eta = 0.1, achieving RMSE = 19.53 on test set. So, we can already know that this model will not be the best one to use.

7.3.3 SVM (Support Vector Machine)

For the SVM, we went with a RBF kernel with grid search over :

- **cost:** 1, 10, 100
- **gamma:** 0.001, 0.01, 0.1
- **epsilon:** 0.01, 0.1, 0.2

We found out after the grid search that the best hyperparameters are: cost = 1, gamma = 0.1, epsilon = 0.01, achieving RMSE = 16.77 on test set. Here also, we can see that the RMSE is too high for the models we have already tested. This model will not be used in the final part of this project either.

Table 3: Machine Learning Models Comparison

Model	Parameters	RMSE on test
SVM (selected lags + temp)	cost: 1, gamma: 0.1, epsilon: 0.01	16.77
XGBoost (selected lags + temp)	nrounds: 50, max_depth: 3, eta: 0.1	19.53

7.4 WNN (Weighted Nearest Neighbors)

The WNN package developed in Part I was applied with cross-validation over 4×96 folds (4 days) to tune hyperparameters:

- Window sizes (w): 96, 192, 384 for ($96 * 15$ minutes = 1 day, $192 * 15$ = 2 days, $384 * 15$ = 4 days)
- Number of neighbors (k): 3, 5, 7, 10, 20

Table 4: WNN Cross-Validation Results

Window Size	k	CV RMSE
384	20	15.83
384	10	16.02
384	7	16.21
192	7	16.42
96	20	16.56

The best WNN configuration was found to be:

- Window size: 384 (4 days)
- Number of neighbors: 20
- **RMSE on test set: 15.51**

While this model is not the best one either, a RMSE of 15.51 is a very good score and can be a very good indicator of what this model is capable of.

8 Results

8.1 Comprehensive Model Comparison

Table 5 presents a comprehensive comparison of all models evaluated on the test set, ranked by RMSE performance.

Table 5: Comprehensive Performance Summary (sorted by RMSE)

Model	RMSE on test	AIC
SARIMA (5,0,0)(0,1,1)	14.81	31923.24
SARIMA (5,0,1)(0,1,1)	14.93	31920.32
WNN (w=384, k=20)	15.51	—
SARIMA (0,0,0)(0,1,1)	15.89	35821.28
Holt-Winters Multiplicative	16.00	—
SVM (selected lags + temp)	16.77	—
Holt-Winters Additive	18.80	—
XGBoost (selected lags + temp)	19.53	—
Auto SARIMA (5,0,0)(0,1,0)	22.65	34372.94

9 Residual Diagnostics

To validate the selected model, comprehensive residual analysis was performed on the SARIMA(5,0,0)(0,1,1) model.

9.1 Statistical Tests

Table 6: Residual Diagnostic Tests for SARIMA(5,0,0)(0,1,1)

Test	Hypothesis	Result
Ljung-Box (lag 96)	No autocorrelation	$p > 0.05$ (Pass)
Shapiro-Wilk	Normality of residuals	$p < 0.05$ (Fail)
Breusch-Pagan	Homoscedasticity	$p > 0.05$ (Pass)

9.2 Interpretation

The residual analysis reveals:

- **No significant autocorrelation:** The Ljung-Box test confirms that residuals behave as white noise, validating the ARMA structure.
- **Non-normal distribution:** While residuals are not perfectly Gaussian, this is common with high-frequency data and does not invalidate point forecasts.
- **Constant variance:** Homoscedasticity is confirmed, supporting the additive error assumption.

10 Model Selection and Final Forecast

The SARIMA(5,0,0)(0,1,1) model achieved the best RMSE of 14.81, closely followed by SARIMA(5,0,1)(0,1,1) with RMSE 14.93 and WNN(w=384, k=20) with RMSE 15.51.

Thus, the **manual SARIMA(5,0,0)(0,1,1) model** was selected for the final prediction, being:

- Statistically sound (uncorrelated residuals confirmed by Ljung-Box test)
- Parsimonious: fewer parameters than SARIMA(5,0,1)(0,1,1) with comparable performance
- Interpretable: AR(5) captures intraday dynamics, seasonal MA(1) handles daily patterns
- Robust: no reliance on exogenous variables that may introduce forecast uncertainty

The optimal SARIMA model was then retrained on the complete dataset (training + test) to generate the final forecast for 96 data points (1-day horizon) for February 19, 2010.

10.1 Key Findings

1. **Seasonality is crucial:** Models that properly account for the 96-period seasonality (SARIMA, Holt-Winters) performed best.
2. **Manual vs Automatic model selection:** The manual SARIMA(5,0,0)(0,1,1) significantly outperformed the automatic SARIMA, demonstrating the value of careful ACF/PACF analysis following Box-Jenkins methodology.
3. **Temperature impact:** Including temperature as an exogenous variable provided marginal improvements but introduced residual correlation issues.
4. **WNN performance:** The WNN algorithm showed excellent performance (RMSE = 15.51), ranking 3rd overall and outperforming Holt-Winters and machine learning models, validating its utility as a non-parametric alternative.

5. **Feature selection matters:** For ML models (XGBoost, SVM), using PACF-based lag selection degraded performance only marginally while producing significantly simpler and more computationally efficient models.

11 Limitations and Future Work

11.1 Current Limitations

1. **Single forecast horizon:** All models were optimized for a 1-day (96-point) horizon. Performance may vary significantly for longer horizons (weekly, monthly forecasts).
2. **Limited exogenous variables:** Only temperature was considered. Other factors (holidays, day-of-week effects, economic indicators) could improve predictions.
3. **Stationarity assumption:** The analysis assumes the underlying consumption patterns remain stable. Structural breaks (e.g., building renovations, occupancy changes) would require model re-estimation.
4. **WNN hyperparameter sensitivity:** While cross-validation identified optimal (w, k) pairs, the algorithm’s performance is sensitive to these choices and will require re-tuning for different datasets.

12 Reproducibility

All experiments were conducted using the following computational environment:

Table 7: Computational Environment

Component	Version
R	4.4.2
forecast	8.21
xgboost	1.7.5
e1071 (SVM)	1.7-13
R6	2.5.1
Random seed	42

All source code, data, and notebooks are available at:

https://github.com/OlivierBOROT/SISE_WNN_implementation

in the **elec-train** folder

13 Conclusion

This study successfully implemented and evaluated the WNN algorithm alongside various forecasting methods for electricity consumption prediction. The manual SARIMA(5,0,0)(0,1,1) model emerged as the optimal choice, achieving an RMSE of 14.81 while maintaining statistical validity through rigorous residual diagnostics.

The WNN algorithm demonstrated excellent performance (RMSE = 15.51), ranking 3rd overall and outperforming traditional Holt-Winters and machine learning approaches. Its non-parametric nature and computational efficiency make it a valuable addition to the forecaster’s toolkit, particularly for datasets exhibiting strong recurring patterns.

The comprehensive comparison demonstrated that while complex models can capture intricate patterns, simpler statistically sound models like SARIMA often provide superior performance for well-structured seasonal data. Future work should explore ensemble approaches combining the strengths of parametric (SARIMA) and non-parametric (WNN) methods.

References

- [1] Talavera-Llames, R.L., Pérez-Chacón, R., Martínez-Ballesteros, M., Troncoso, A., Martínez-Álvarez, F. (2016). A Nearest Neighbours-Based Algorithm for Big Time Series Data Forecasting. In: *Hybrid Artificial Intelligent Systems. HAIS 2016*. Lecture Notes in Computer Science, vol 9648. Springer, Cham.
- [2] Hyndman, R.J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice* (2nd ed.). OTexts: Melbourne, Australia. <https://0Texts.com/fpp2>.