DDBI Lecture 4: Machine Learning Methods for Multivariate Time Series Forecasting

A Hands-On Introduction & Preparation for Lab

Dr. Stavros K. Stavroglou

December 23, 2024

Outline

1. Introduction & Goals
2. Support Vector Regression (SVR)
3. K-Nearest Neighbors (KNN)
4. Decision Trees & Random Forests
5. Simple vs. Rolling Forecasts
6. Pitfalls & Best Practices
7. Conclusion & Next Steps

Lecture Objectives

In this lecture, you will:

- Explore the use of various **Machine Learning (ML)** techniques for forecasting a **target climate index (ENSO)** using multiple regressors (e.g. AAO, AO, NAO, PNA, and lagged ENSO).
- Understand each model's **theoretical foundation**, including strengths and weaknesses:
  5Support Vector Regression (SVR)K-Nearest Neighbors (KNN)Decision TreeRandom Forest
- Compare **simple out-of-sample forecasts** vs. **rolling forecasts**, discussing the trade-offs of each approach.
- Prepare for the **coding lab session**, where you will implement and evaluate these methods.

SVR: Core Concept

Support Vector Regression (SVR) is an adaptation of Support Vector Machines (SVM) for regression tasks:

- Attempts to fit a function within an ϵ-tube around the data (i.e. an ϵ margin of tolerance).
- Minimizes model complexity (via $\|w\|^2$) subject to penalties for points lying outside the ϵ-tube.
- Can use **kernel functions** (RBF, polynomial, sigmoid, etc.) for non-linear relationships.

Pros:

- Handles **high-dimensional** or complex feature spaces well.
- Often robust to outliers, depending on **C** and ϵ settings.

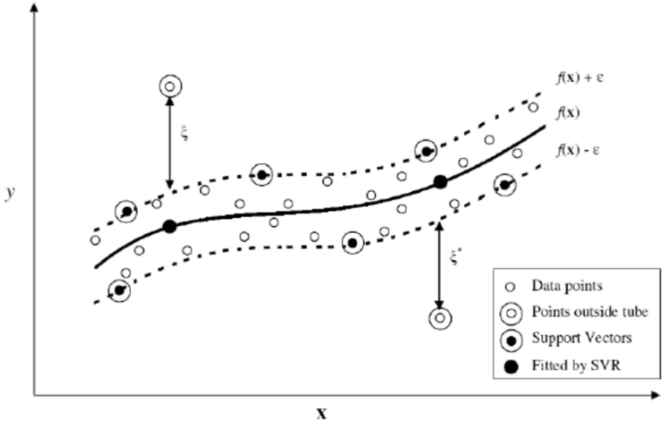
Cons:

- **Hyperparameter tuning** (e.g. kernel type, C, γ) can be non-trivial.
- Performance may suffer on **very large** datasets due to complexity.

SVR Forecasting Workflow (1/2)

- **Data Preparation:**
  - Select regressors: AAO, AO, NAO, PNA, possibly ENSO with lags.
  - Split into **train** and **test** sets (e.g. 80-20 split).
- **Model Training:**
  - Choose kernel (`rbf`, `linear`, `poly`, etc.).
  - Tune **C** (penalty factor), ϵ, and kernel-specific parameters (e.g. γ for RBF).
- **Evaluation:**
  - Generate **out-of-sample** predictions for the test period.
  - Possibly compute **confidence intervals** via residual-based bootstrapping.
- *We'll see an example of the final forecasts on the next slide.*

SVR Forecasting Workflow (2/2)



(SVR regression diagram.)

KNN: Intuition

- **Distance-Based** approach:
  - For a new sample, find its K closest neighbors (by Euclidean or other distance).
  - Predict by **averaging** (or weighting) the neighbors' target values.
- Completely **non-parametric** (stores training data and uses it at prediction time).

Pros:

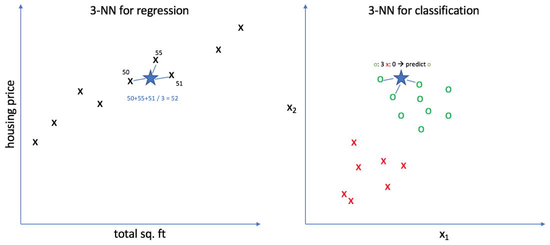
- Conceptually **simple** and easy to explain.
- No explicit model parameterization (beyond **K**).

Cons:

- **Slow** for large datasets (distance computations).
- **Scaling** matters: Variables on larger scales can dominate distance.
- No built-in handling of time-based **lags** unless you explicitly engineer them.

KNN Forecast Example

- Steps:
  1. Choose K (e.g. 5, 10).
  2. Scale features if needed.
  3. Compute predictions for test set; observe and compare the results.
- Can produce **chunky** or **step-like** forecasts if the data is smooth.



(KNN regression/classification illustration.)

Decision Tree: Basics

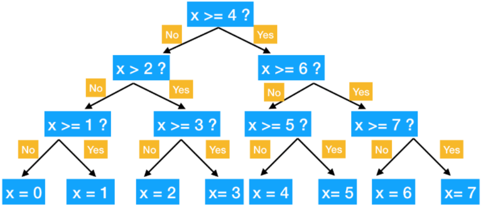
- Splits the feature space at various **thresholds** (e.g. AO < 0.5, etc.).
- Final prediction is often the **mean** of target values in the corresponding leaf node.

Pros:

- Very **interpretable** (you can visualize the tree).
- Automatically captures **non-linearities** and interactions.

Cons:

- High variance, can **overfit** if too deep.
- Sensitive to small changes in training data.



Random Forest: Why Ensemble?

- **Random Forest** is an **ensemble** of many Decision Trees:
  - Each tree is built on a **bootstrap sample** of the training data.
  - Random subsets of features are used at each split.
- **Reduces variance**: By averaging many diverse trees, we get a smoother and often more accurate prediction.

Pros:

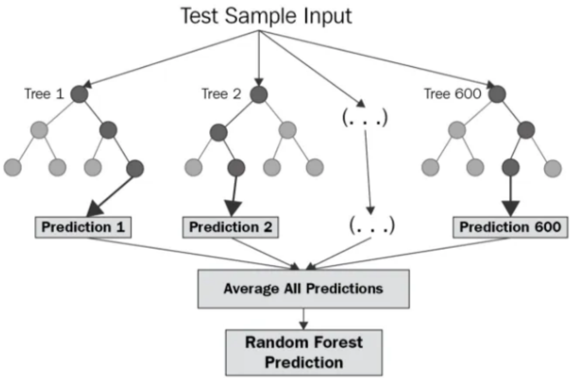
- Typically **more robust** than a single tree.
- Good at capturing complex interactions among regressors.

Cons:

- **Less interpretable** (we lose the simple tree structure).
- Still possible to overfit if not properly tuned (e.g. `max_depth`, `n_estimators`).

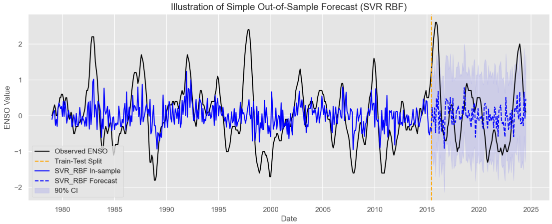
Forest Forecast

- Steps in training:
  1. Choose `n_estimators` (# of trees) and `max_depth`.
  2. Train on the historical data, then generate out-of-sample predictions.
- Often yields **smooth** but flexible forecasts.



Simple Out-of-Sample Forecast

- **Procedure**:
  1. Fit the model on the entire training set.
  2. Predict the *entire* test set *in one shot*.
- **Pros**:
  - Simpler, faster to compute.
  - Good for an initial benchmark.
- **Cons**:
  - Model never sees updated data once the test period starts.
  - May be **overly optimistic** if the data is non-stationary.



Rolling Forecast Approach

Rolling (or iterative) forecast is more realistic:

1. Fit the model on the training set.
2. Predict **one step** (e.g. next month) in the test period.
3. **Add** that *actual* observation to your training data.
4. Retrain or update the model.
5. Predict the next step, repeat until done.

Pros:

- Simulates **real-time** updating.
- May improve forecasts as new data arrives.

Cons:

- **Computationally expensive** (model fit repeated many times).
- Implementation complexity.

Pitfalls in Practice

Be mindful of:

- **Overfitting:**
  - Especially true for Decision Trees and Random Forest if `max_depth` is large.
  - KNN can also overfit if K is too small.
- **Feature Scaling/Engineering:**
  - Distance-based methods (KNN, some kernels in SVR) are sensitive to unscaled data.
  - Time series often benefit from lagged features, rolling means, and seasonal indicators.
- **Hyperparameter Choice:**
  - Blind guesses of C, γ (SVR) or `n_estimators`, `max_depth` (RF) can lead to poor results.
  - Use a **validation set** or cross-validation approach (e.g. time series split).

Best Practices to Follow

- **Experiment with simpler models first (baseline):**
  - A simple linear model or naive forecast sets a reference point.
  - Then see if ML models improve on that baseline.
- **Incorporate domain knowledge:**
  - Climate indices may have known seasonality or cyclical patterns.
  - Feature engineering can drastically improve performance.
- **Employ structured hyperparameter tuning:**
  - Grid search, random search, or Bayesian optimization.
  - Use metrics relevant to your domain (e.g. MASE if comparing across scales).
- **Consider rolling windows or expanding windows for validation**:
  - Time series cross-validation is different from random splitting.
  - Mimics real-world forecast scenarios.

Key Takeaways

- **SVR, KNN, Decision Trees, Random Forests** can each handle multivariate time series forecasting, but with different biases and complexity.
- **Rolling vs. Simple forecasts** provide different perspectives:
  - Rolling = more realistic, more computation.
  - Simple = quick, but may not reflect real-time updates.
- **Hyperparameter Tuning** is essential to unlock the models' full potential.
- Evaluate models with **multiple approaches** (baseline comparisons, domain insights, and cross-validation).

Next Steps & Lab Preview

- **In the coding session, you will:**
  - Load `combined_climate_indices_2024.csv`.
  - Implement each ML model (SVR, KNN, Decision Tree, Random Forest).
  - Compare forecast performance in **simple** and **rolling** scenarios.
  - Experiment with hyperparameters to see how results change.
- **Additional Explorations:**
  - Feature engineering (e.g. lagging ENSO, creating rolling averages).
  - Incorporating domain-specific insights (seasonality, etc.).
  - Trying advanced methods (e.g. Gradient Boosting, Neural Networks).

Get ready to code!

Thank you! Questions?