# Comparison of Forecasting Strategies in Machine Learning and Statistical Methods

**Author:** Dr. Pedram Jahangiry **Course:** DATA5630 - Fall 2025 **Module:** Machine Learning Forecasting

## Comparison of Forecasting Strategies in MLForecast (using Random Forest models)

| Forecasting Strategy | Forecasting Horizon Handling | Number of Models Trained | Error Propagation Risk | API Support in MLForecast | Feature Construction Notes | Training Complexity | Typical Use Cases |
|---|---|---|---|---|---|---|---|
| **Single-Output (Recursive)** | Predicts 1 step ahead and feeds each prediction back in to forecast the next, iterating through the horizon. | 1 (single model reused for all steps). | High – errors accumulate at each step as predictions are reused. | Yes – default strategy (no special configuration needed). | Uses standard lag features for the next-step prediction; features are recomputed after each new forecast. | Low – only one model to train (simple pipeline). | Common baseline approach; default in traditional models like ARIMA/ETS; suitable when data is limited or model simplicity is preferred. |
| **Multi-Output (Direct)** | Forecast each horizon step with its own independent model (no recursive dependence on prior predictions). | H models (one per forecast horizon step). | Minimal – no accumulation since each step is predicted independently (no feedback of previous forecast errors). | Yes – supported via `max_horizon` parameter to train one model per step. | Similar lag-based features can be used for each model; no horizon index needed as each model targets a specific lead time. | High – training multiple models is time-consuming (more training time, but can improve accuracy). | Often used when each forecast step's accuracy is critical and extra training is acceptable; ideal for short to mid-range horizons to avoid error compounding. |

| Forecasting Strategy | Forecasting Horizon Handling | Number of Models Trained | Error Propagation Risk | API Support in MLForecast | Feature Construction Notes | Training Complexity | Typical Use Cases |
|---|---|---|---|---|---|---|---|
| **Multi-output (Single model)** | Use a single model to predict all future steps in one shot (multi-output prediction of the entire horizon). | 1 (one model outputs a vector of H forecasts). | None – forecasts are generated simultaneously, so no iterative error feedback between steps. | Not directly built-in – requires a custom approach (e.g. add a "horizon" index as a feature or use a multi-target regressor) to use one model for all steps. | May include a horizon indicator feature so the model knows which step is being predicted; no need to recalculate features for each step after training (the model learns all steps together). | Moderate – one complex model to train (can capture dependencies between outputs; may train slower and needs more data to generalize). | Used to avoid error accumulation while managing a single model; useful if ample data is available to learn multi-step patterns. Often seen as a trade-off strategy to achieve direct-approach accuracy with one model. |

## Comparison of Forecasting Strategies in StatsForecast

| Forecasting Strategy | Description | Supported in StatsForecast (ARIMA/ETS)? | Implementation Details |
|---|---|---|---|
| **Single-Output (Recursive)** | Train one model for one-step-ahead, then iteratively feed forecasts back in to reach H steps. Avoids retraining, but errors compound. | Yes (default). All statsforecast statistical models use recursive multi-step forecasts by default. For example, calling `sf.predict(h=H)` on an ARIMA will internally generate 1-step, 2-step, … forecasts in sequence | Out-of-the-box: simply fit once and use the built-in `.predict(h)` or `.forecast()` method to get H steps. The library handles the recursion internally. |
| **Multi-Output (Direct)** | Train H separate models, each specialized to predict the t+k horizon directly (one model per step). Eliminates cascading errors at the cost of more models. | No (not built-in). StatsForecast has no parameter to auto-train multiple horizon-specific models – classical ARIMA/ETS are fit on one-step forecasts only. (By contrast, Nixtla's MLForecast supports this via `max_horizon` for ML models) | Manual only: If desired, you must manually fit separate StatsForecast models or rerun `.fit()` for each horizon using modified data. This is labor-intensive and not natively supported by the library's API. |
| **Multi-output (Single model)** | Train one model that outputs a vector of all H future values in one go. Often implemented via a multi-output regressor or by using horizon as an input feature to a single model. | No. Classical models in statsforecast are univariate and produce a single-step forecast. They cannot natively emit multiple-step outputs in one prediction. There is no ARIMA/ETS that directly yields, say, a 7-day forecast vector – it's achieved through recursion instead | Not supported: There is no out-of-the-box multi-output mode for ARIMA/ETS in StatsForecast. Achieving this would require using a different framework or approach (e.g. ML models with horizon features or specialized multi-output algorithms, which fall outside statsforecast's scope). |

## Key Takeaways

1. **MLForecast** provides flexible strategies for machine learning models (like Random Forest):

- Recursive forecasting is the default and simplest approach
- Direct multi-output forecasting is supported via `max_horizon` parameter
- Custom single-model multi-output approaches require additional implementation

2. **StatsForecast** focuses on traditional statistical models (ARIMA/ETS):
   - Only recursive forecasting is natively supported
   - Multi-output strategies require manual implementation
   - Best suited for classical time series approaches
3. **Error Propagation** is a critical consideration:
   - Recursive methods compound errors over longer horizons
   - Direct methods avoid error accumulation but require more computational resources
   - Single-model multi-output strikes a balance but needs sufficient training data