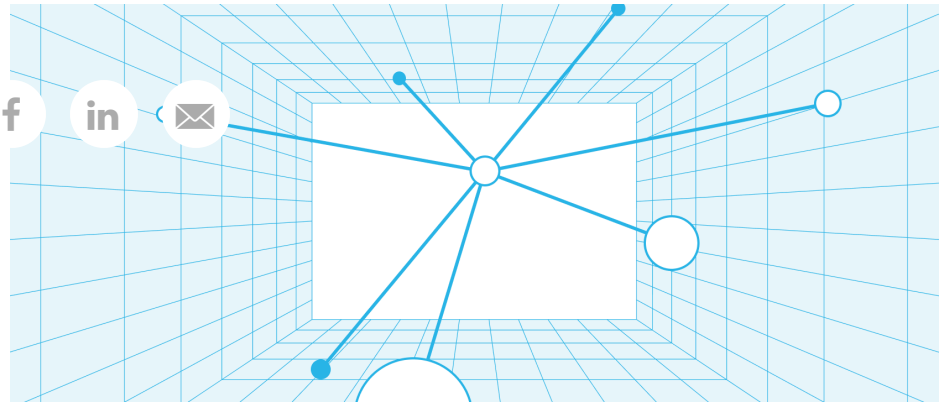
JUN 27, 2024

AUTHOR

Kyle Schmaus

Time-Series Forecasting: Comparing Transform Techniques for Tree-Based Models

Machine Learning



A key question in data analytics is: *What happens next?* At Snowflake, this time-series forecasting problem is not just academic. Our customers maintain vast and complex data sets with us — sometimes exceeding 50 trillion rows! We want to give our users (even nontechnical ones) the ability to quickly extrapolate their Snowflake data using state-of-the-art machine learning (ML) algorithms without fiddling with esoteric knobs or requiring extensive cleaning and preprocessing — all while still properly capturing overall data trends.

That's a tall order! As the engineers behind Snowflake's time-series forecasting technology, we wrote this article to share one of the simple but powerful tricks we use to accomplish these goals.

Why use tree-based models for forecasting?

First, some background. Time-series forecasting predicts future trends based on previously observed target data, as well as optionally provided feature data. We assume a fixed sampling frequency, where target data is observed at regular intervals (e.g., hourly).

AUTHOR

Kyle Schmaus

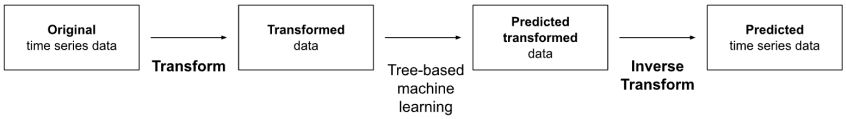
Time-series forecasting systems can rely on statistical time-series models, tree-based methods or neural network-based approaches. Classical methods, such as ARIMA and Exponential Smoothing, are fast to train, highly efficient and easy to interpret, while neural methods have increased accuracy and can incorporate exogenous inputs and non-linearities, though at the expense of training and inference efficiency.

For the launch of our **forecasting function**, we chose tree-based models as a happy middle ground. They aren't as simple and interpretable as classical statistical methods. Still, they train in a fraction of the time of neural methods, leverage exogenous features non-linearly and can have accuracy close to (if not exceeding) that of neural methods. Models, such as gradient-boosted decision trees, can be adapted to time-series forecasting, even with no feature data, by constructing transformations of the target data to serve as features. These can be simple lags of the endogenous rolling averages or other, more complicated history-based transformations of the endogenous data. We can also encode rule-based cyclic features from either the timestamp values associated with target observations or from Fourier analysis of the target signal.

How to marry tree-based models with time-series extrapolation?

Unfortunately, tree-based models have trouble extrapolating. Whereas a neural network or linear regression model can extrapolate the trend of an increasing time series, tree-based models cannot, as they rely on decision boundaries derived from the training data. This isn't a problem when a time series is stationary, and its future values are expected to look similar to its past. However, when modeling an increasing or decreasing time series, steps must be taken to adjust the target data to accommodate the tree model's lack of extrapolation capabilities.

One of the most common methods of supporting time-series extrapolation for tree-based models is adopting a preprocessing transformation of the target data, along with a corresponding, postprocessing inverse-transformation. Basically, this looks like:



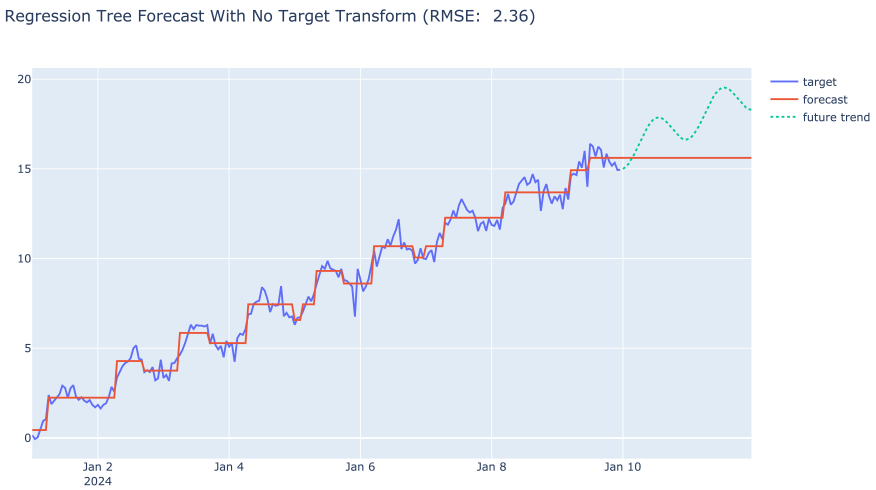
AUTHOR

Kyle Schmaus

This blog post studies how different **transforms** (and their respective **inverse transforms**) affect the quality of a time-series prediction. We illustrate why a **weighted average transform** enables even simple tree-based methods to extrapolate nonstationary time-series in a way that is robust to outliers.

Algorithm 1: Tree-based extrapolation with *no* transformation

'st le' strate the extrapolation problem, let's explore using tree-based methods without applying a target transformation at all. We'll use hourly synthetic data with a 24-hour cycle and a strongly increasing trend. To keep things simple, we're going to use a scikit-learn DecisionTreeRegressor model, with max_depth=4, for all the following examples. We will use two features, the increasing timestamp itself, and a cyclic hour-of-day feature. As one can see, the tree is incapable of continuing the extrapolation trend, as it can only interpolate the original data.



Algorithm 2: The *difference* transformation

Next, we look at a simple “difference” transformation, which is what we used at the launch of our function. This is similar to the “integration” step of an ARIMA model, where we define a new endogenous time-series **z**, derived from our “raw” series **y**, by defining the new **t**’th observation to be equal to the difference between the value of **y** at time **t** and the value of **y** **k** steps earlier.

AUTHOR

Kyle Schmaus

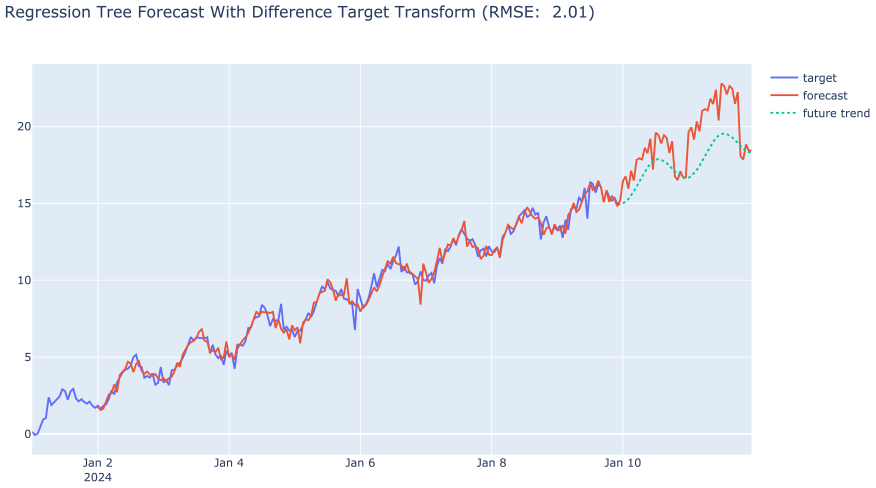
$$z_t = y_t - y_{t-k}$$

The **k** lag value can be set to 1, or to a “reasonable” value depending on the sampling frequency (e.g., 24 for the hourly data in our example). This transformation helps to induce trend-stationarity, albeit at the cost of dropping the first **k** values of our endogenous data. We train our tree-based model on the transformed series, and then when we produce forecasts, we inverse transform the data back to the original scale.

$$\hat{y}_t = \hat{z}_t + y_{t-k}$$

Note, this requires keeping track of the last **k** values of the untransformed endogenous series – beyond those, we recursively use previously generated forecasts, always offset by **k** steps.

While this transformation behaves well when there is little noise or stochasticity in a time series, one lesson we learned is that the difference transformation can memorize and repeat a noisy artifact over and over again. In our synthetic data example, a sharp downward pointing spike can be seen repeating in the next two forecasted cycles.



AUTHOR

Kyle Schmaus

SHARE

Algorithm 3: The *window* transformation

To remedy this, we generalized the difference transformation to a window-difference transform. Now instead of subtracting the last **k**'th value, we subtract the mean of **k**'th through the (**k** + **w** − 1)'th

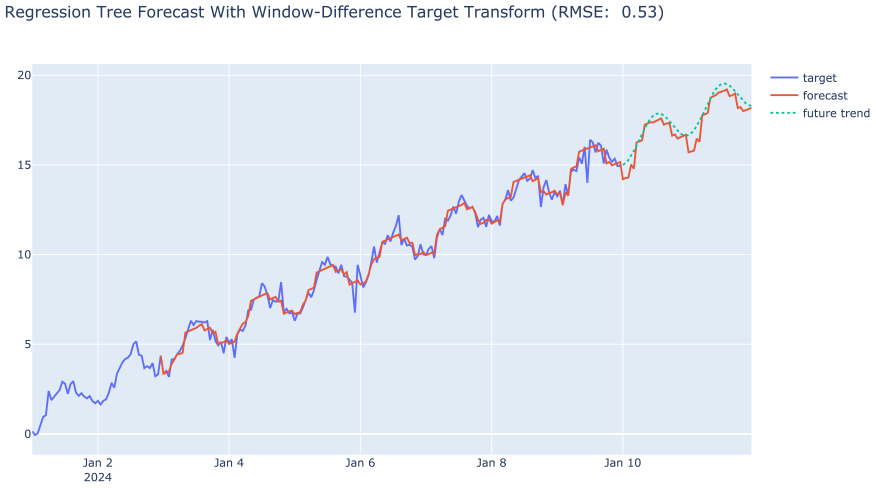
$$z_t = y_t - \frac{1}{w} \sum_{i=0}^{w-1} y_{t-k-i}$$

The inverse transform is now:

$$\hat{y}_t = \hat{z}_t + \frac{1}{w} \sum_{i=0}^{w-1} y_{t-k-i}$$

At the cost of dropping **w** more endogenous observations, we observed much smoother forecasts. Note, in practice, we usually set **k** and **w** equal to the same value.

For our synthetic data, we see the resulting forecast is much more robust to noise and outliers, and it follows the future trend much more faithfully.



AUTHOR

Kyle Schmaus

SHARE

Conclusion

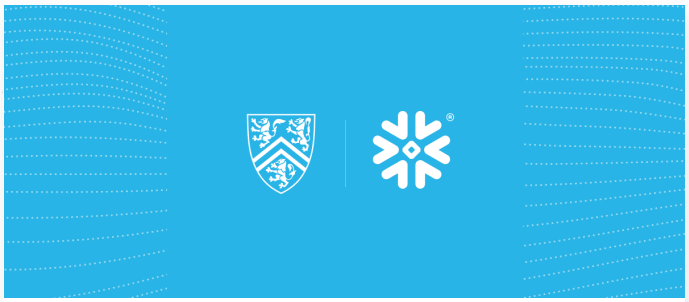
In this blog post, we shed some light on the techniques we use to create robust time-series forecasting at Snowflake. We covered how we've adapted tree-based machine learning models to enable ıe es casting, focusing on a generalization to the commonly used difference target transform. This improvement is just a single stop on our journey of making ML accessible to all. We are on a path of continuous growth and innovation, with more achievements to look forward to! Our goal is to enable our users to perform time-series forecasting (and other ML functionality, such as anomaly detection) without needing to tune parameters and adjust methodology themselves.

Define the future of Enterprise AI with us

Snowflake technology underpins some of the most consequential enterprise data workloads in the world. Snowflake AI Research is pushing the state-of-the-art in AI to understand and process the vast quantities of data that power our society. We've built Arctic, TILT, Cortex, Streamlit, Modin and many more fundamental technologies. Come join us!

SHARE

RELATED CONTENT



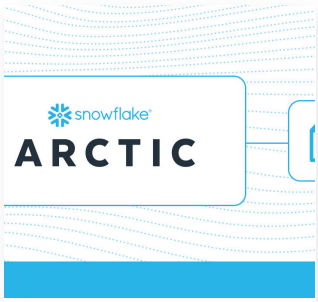
Product and Technology AI & ML

MAY 02, 2024

Moving Beyond MTEB and BEIR: Snowflake AI Research Joins Forces with the University of Waterloo to Evolve RAG and Retrieval Benchmarks

To accurately answer business questions using LLMs, companies must augment models with their data. Retrieval Augmented Generation (RAG) is a popular solution to this problem, as it integrates the organization's...

Have a look



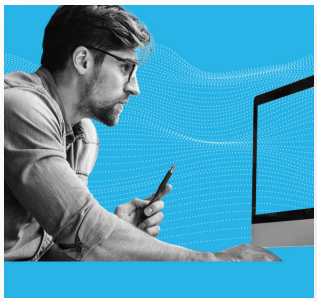
Product and Technology AI & ML

MAY 02, 2024

Snowflake's Arctic-TILT: A State-of-the-Art Document Intelligence LLM in a Single A10 GPU

The volume of unstructured data — such as PDFs, images, video and audio files —...

Expand your knowledge



Product and Technology AI & ML

APR 16, 2024

Snowflake Launches the World's Best Practical Text-Embedding Model for Retrieval Use Cases

Today Snowflake is launching and open-sourcing with an Apache 2.0 license the Snowflake Arctic embed...

More to follow

START NOW

Snowflake Inc.

AUTHOR

Kyle Schmaus

SHARE