Competence Development

Time Series Forecasting Review

Jonas Petersen
Ander Runge Walther

Revision History

Revision	Date	Author(s)	Description
0.0.1	21-08-2024	JP	Created
0.0.2	24-09-2024	JP	Broad update

Contents

Chapter 1

Overview

1.1 Introduction

Forecasting time series data is a critical task in various domains, including finance, retail, healthcare, and manufacturing. It involves predicting future values based on previously observed data, and selecting the right model or algorithm for the task can greatly impact the accuracy and reliability of predictions. The goal of this project is to explore and evaluate a range of forecasting algorithms available in Python, analyzing their strengths and weaknesses through hands-on experimentation with a benchmark dataset. To limit the scope a bit, the forecasting algorithms will be evaluated relatively "out of the box" without extraordinary amounts of either hyperparameter tuning or feature engineering.

This project aims to provide a broad yet practical understanding of various forecasting techniques and the Python packages that implement them. Specifically, we will experiment with the following algorithms:

1. **Gaussian Processes (via PyMC):** A non-parametric Bayesian approach to modeling time series data that can provide uncertainty estimates along with predictions.
2. **Bayesian Two-layer Perceptron (via PyMC and TensorFlow/Keras):** A neural network model with a Bayesian framework to quantify uncertainty in the predictions.
3. **Kalman Filter (via filterpy):** A recursive algorithm for linear dynamic systems that estimates the state of a process in real-time.
4. **Decision Trees (via XGBoost, LightGBM, PyMC):** A non-linear, non-parametric approach that uses ensembles of decision trees for regression tasks.
5. **Prophet (via Prophet):** A decomposable time series model designed for forecasting at scale with an emphasis on interpretability.
6. **LSTM (via TensorFlow/Keras):** A type of recurrent neural network that is well-suited for modeling sequential data with long-term dependencies.

7. **SARIMAX (via statsmodels):** An extension of the ARIMA model that includes seasonality, exogenous regressors, and error correction mechanisms.

8. **Exponential Smoothing (via statsmodels):** A method for forecasting univariate time series by weighting past observations exponentially.

9. **Sorcerer (via PyMC):** A Prophet-inspired Generalized Additive Model (GAM) implemented in PyMC, which combines the flexibility of GAMs with Bayesian modeling techniques to provide robust time series forecasts.

For each method, we will utilize appropriate Python packages, such as PyMC for Bayesian methods, TensorFlow/Keras for deep learning models, and statsmodels for traditional statistical models. The project will be managed using Visual Studio Code with Anaconda, and version control will be handled with Git, creating a repository for each competency development project.

1.2 Data

To benchmark the forecasting algorithms, an aggregated version of the M5 dataset, a widely recognized dataset in the forecasting community, will be used. The M5 dataset contains daily sales data from Walmart's stores and departments across various geographical locations in the United States. The dataset is available from the M5 Forecasting Competition on Kaggle and provides a comprehensive and challenging setting to test a broad range of forecasting algorithms due to its scale, granularity, and multiple hierarchies (e.g., store, department, and product categories).

This study will focus on the weekly aggregated sales data for a particular store-category combination, specifically household sales. The time range of interest spans from 2011 to 2016 (see figure 1.1) with sales for each week.

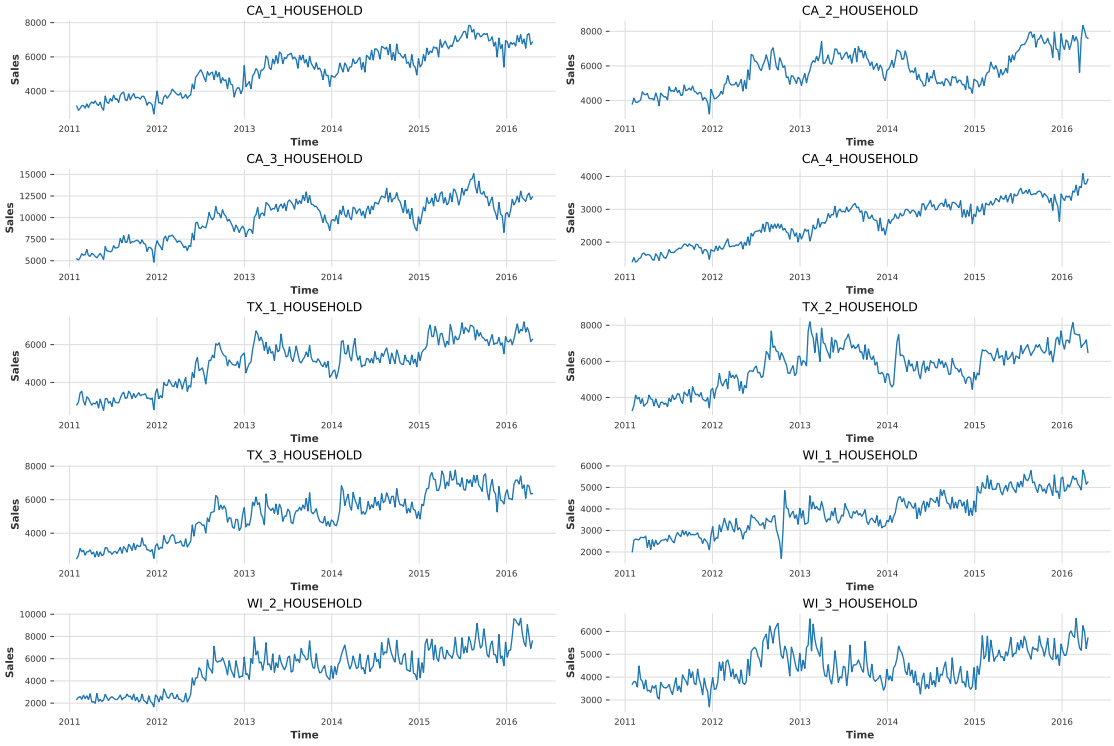

Figure 1.1: Weekly Sales for Each Store-Category

1.3 Metric

In order to quantify and compare the accuracy of different forecasting algorithms, the Mean Absolute Scaled Error (MASE)

$$\text{MASE} = \frac{\text{mean absolute forecasting error}}{\text{mean gradient in training data}} \tag{1.1}$$

is used. The mean gradient in the training data represents the average forecasting error when using the Naive forecast (predict the next timestep to be equal to the current), and so the MASE represent how good a model is relative to this. This study will consider simultaneous forecasting of ten time series ($N_{\text{time series}} = 10$) (see figure 1.1) over a forecast horizon of 26 weeks ($N_{\text{forecast horizon}} = 26$) and so a bit of notation must be introduced to handle this. In order to produce a robust estimate of the MASE, an ensemble of N_{ensemble} forecasts are consider. The ensemble is produced by producing forecasts for each element in on the interval

$$[N - (N_{\text{ensemble}} + N_{\text{forecast horizon}}), N - N_{\text{forecast horizon}}] \tag{1.2}$$

where N is the number of datapoints in the time series (all have the same number of data points for simplicity). For each time step in the forecast horizon, there will therefore be

N_{ensemble} values from which to determine a MASE for each time series and forecast time step. Let $f_{i,j,q}$ and $y_{i,j,q}$ denote the model forecast and data, respectively, for time step i, time series j and ensemble number q then

$$\text{MASE}_{i,j} = \frac{\frac{1}{N_{\text{ensemble}}} \sum_{q=1}^{N_{\text{ensemble}}} |f_{i,j,q} - y_{i,j,q}|}{\frac{1}{N_{\text{training}}-1} \sum_{i=1}^{N_{\text{training}}} |y_{i-2,j} - y_{i,j}|} \tag{1.3}$$

where $N_{\text{training}} \equiv N - (N_{\text{ensemble}} + N_{\text{forecast horizon}}) - 1$ represent the part of the data that is common to all ensembles. To condense the information, the MASE can be averaged over time series

$$\text{MASE}_i \equiv \frac{1}{N_{\text{time series}}} \sum_{j=1}^{N_{\text{time series}}} \text{MASE}_{i,j} \tag{1.4}$$

where

$$\text{MASE}_i \equiv \text{Average MASE over time series} \tag{1.5}$$

and again over the forecast horizon

$$\langle \text{MASE} \rangle \equiv \frac{1}{N_{\text{forecast horizon}} N_{\text{time series}}} \sum_{i=1}^{N_{\text{forecast horizon}}} \sum_{j=1}^{N_{\text{time series}}} \text{MASE}_{i,j} \tag{1.6}$$

where for shorthand

$$\langle \text{MASE} \rangle \equiv \text{Average MASE over time series and forecast horizon.} \tag{1.7}$$

As a *rough* estimate of uncertainty for equation (1.7), the standard deviation of the average over time series is considered

$$\delta \langle \text{MASE} \rangle = \sqrt{\frac{1}{N_{\text{forecast horizon}} - 1} \sum_{i=1}^{N_{\text{forecast horizon}}} (\text{MASE}_i - \langle \text{MASE} \rangle)^2} \tag{1.8}$$

which assumes the variation between time series can be neglected relative to the variation due between forecast horizon steps. The uncertainty is intended to be a conservative estimate of the variance in the sample rather than the uncertainty of the mean estimate (if the latter is intended, the variance in equation (1.4) and error propagation can be used). Equation (1.4) and (1.7) are shown for the models considered in figure 6.1 and 6.2, respectively.

Chapter 2

Profile Approach

$$content... \tag{2.1}$$

Chapter 3

PYMC Approach

The advantage is that there is complete control over the architecture and the formalism is Bayesian, meaning there is uncertainty and a statistical model behind. The downside is the computation time and the expertise required to get a good model running.

3.1 pymc TLP

The advantage is this is a flexible one-size-fits-all approach, which can accommodate multidimensional time series. The downside is that the computation time and lack of interpretability of the coefficients. Feature engineering is also required to get a good result.

3.2 pymc prophet

The advantage is a flexible model that can accommodate multidimensional time series and no feature engineering is required. The downside is the computation time and difficulty in getting the result to converge. The coefficients are highly interpretable, which is a strength, but this leans into the difficulty of obtaining convergence of a good result.

3.3 pymc ar

3.4 pymc gaussian processes

Chapter 4

Statsmodels

4.1 TimeSeriesAnalysis

The advantage is it does not require feature engineering, is fairly accurate and fast. It also includes uncertainty.

Chapter 5

Light GBM Approach

This is very fast and accurate, however, with not apparent uncertainty quantification and it does not generalize to multiple time series.

Chapter 6

Results

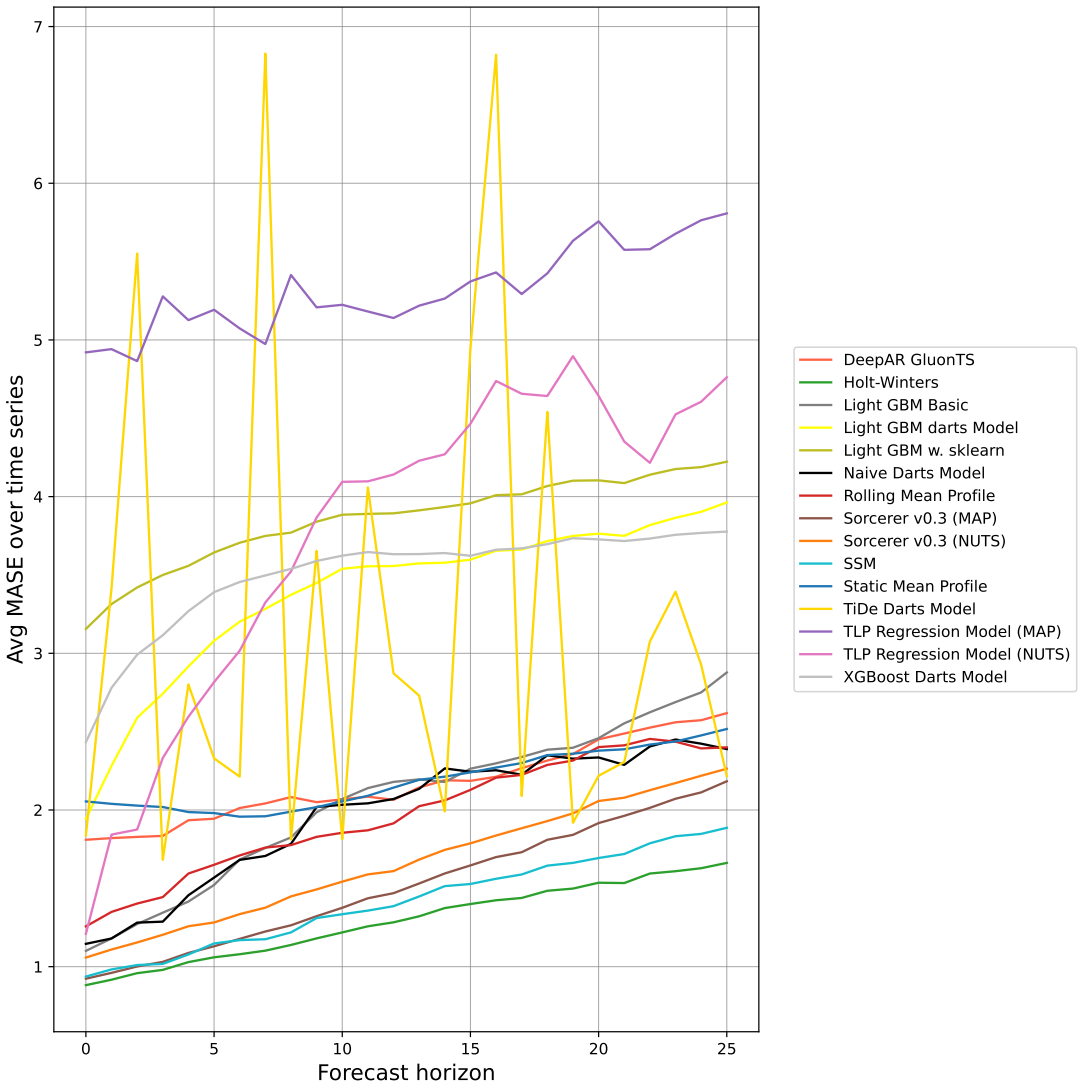

Figure 6.1: Average MASE over time series.

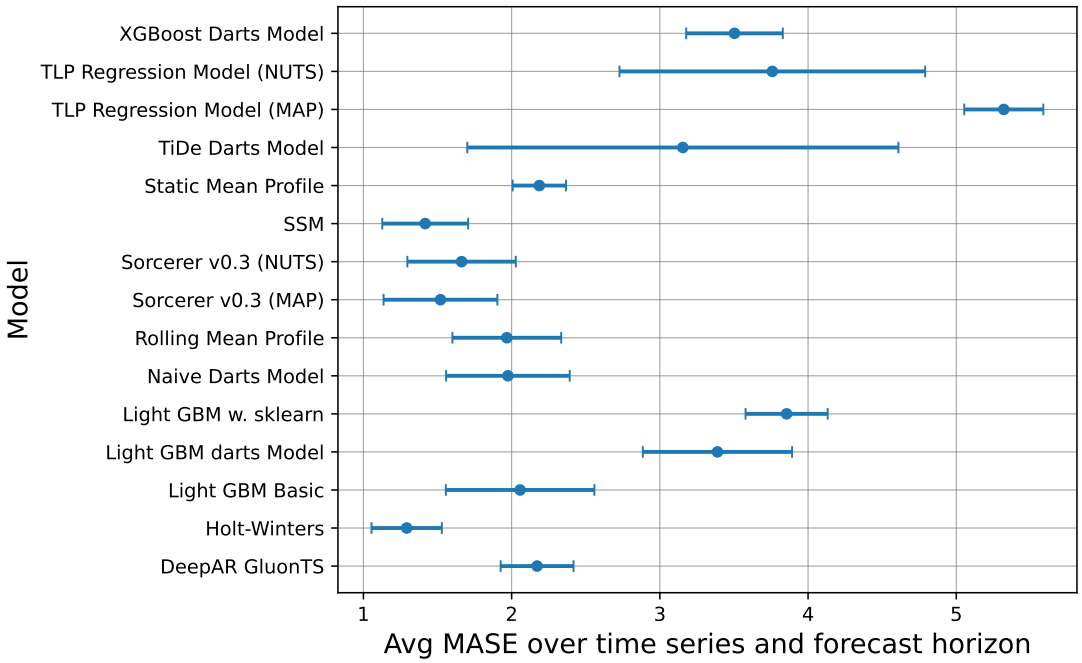

Figure 6.2: Average MASE over time series and forecast horizon.

Chapter 7

Acknowledgements

Simon: lgbm_feature_darts, SSM, lgbm_darts, tide_darts, tft_darts