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

Contents

1 Overview **3**
1.1 Introduction . . . 3
1.2 Data . . . 4
1.3 Metric . . . 4

2 Profile Approach **6**
2.1 Method . . . 6

3 PYMC Approach **10**
3.1 pymc TLP . . . 10
3.2 pymc prophet . . . 10
3.3 pymc ar . . . 10
3.4 pymc gaussian processes . . . 10

4 Statsmodels **11**
4.1 TimeSeriesAnalysis . . . 11

5 Light GBM Approach **12**

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.

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, we will use the M5 dataset, a widely recognized dataset in the forecasting community. The M5 dataset contains daily sales data from Walmart's stores and departments across various geographical locations in the United States. This 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).

We 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. For this analysis, the data before 2012 and after 2015 is discarded. The initial period is removed to eliminate potential startup effects and noise from the early data collection phase, while the final period is excluded to simplify the evaluation by using complete calendar years.

The remaining data is then split into training and testing sets based on a specified date (‘2015-01-01‘), ensuring the training data ends with a full calendar year, accommodating a "profile" approach to forecasting. This approach facilitates the models in capturing seasonal patterns and other calendar-based trends more effectively.

Figure 1.1 visualizes the aggregated weekly sales data after discarding the initial and final periods. The highlighted areas represent the discarded data, and the dashed vertical line indicates the split point between the training and test sets. This visualization helps to understand the overall sales trend and the data points to be retained for model training and evaluation.

The dataset and its visualization serve as the basis for comparing different forecasting methods, highlighting their capabilities in capturing patterns and making accurate predictions.

1.3 Metric

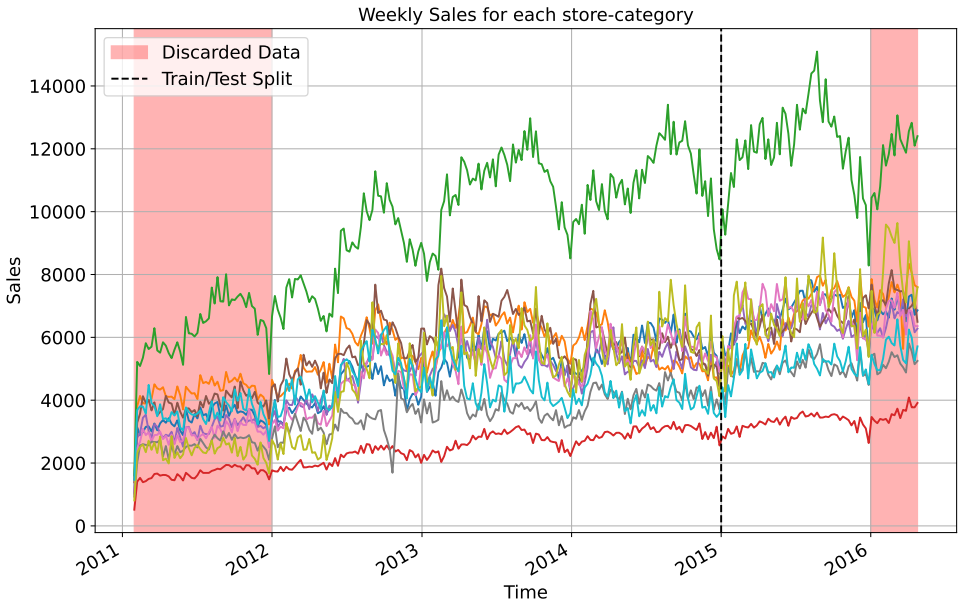

Figure 1.1: Weekly Sales for Each Store-Category

Chapter 2

Profile Approach

2.1 Method

Consider the problem of having a time series that yield the demand of some product with one entry per week and generating a representative forecast for this. The problem can be framed as a game between Nature and a Robot, where each make a decision. Nature's decision ($s_i \in \Omega_S$) is to picks the true demand and the Robot's decision ($U_i(D) \in \Omega_U$) is guided to minimize a cost function defined viz

$$C : \Omega_U \times \Omega_S \mapsto \mathbb{R} \tag{2.1}$$

where Ω_U denotes the action space of the Robot and Ω_S denotes the action space of Nature. The objective of the Robot is to minimize the expected cost across it's actions. Let

$$G \equiv \sum_{i=1}^{n} C(U_i(D), S_i), \tag{2.2}$$

then

$$\mathbb{E}[G|D, I] = \int ds_1 ds_2 \dots ds_n \sum_{i=1}^{n} C(U_i(D), s_i) p(s_1, s_2, \dots s_n|D, I) \tag{2.3}$$

where $i \in$ "weeks in forecast horizon" cover the weeks in the forecast horizon and I denotes the background information. The minimum cost across the Robot's actions is defined viz

$$\frac{d}{dU_i(D)} \mathbb{E}[G|D, I]\Big|_{U_i(D)=U_i^*(D)} \stackrel{!}{=} 0. \tag{2.4}$$

where $U_i^*(D)$ is the optimal decision rule for the Robot. In this project, the cost function is taken to be

$$C(U_i(D), s_i) = (U_i(D) - s_i)^2 \tag{2.5}$$

Combining Equation 2.3, Equation 2.5 and Equation 2.4 yields

$$U_i^*(D) = \int ds_1 ds_2 \dots ds_n s_i p(s_1, s_2, \dots s_n|D, I). \tag{2.6}$$

Equation 2.6 informs that the optimal decisions for the Robot should be the expected decisions of Nature, which makes intuitive sense considering the cost function (equation (2.5)).

To apply equation Equation 2.6, the probability distribution $p(s_1, s_2, \dots s_n|D, I)$ needs to be specified. To do this, a host of assumptions regarding the process generating the data must be made.

2.1.1 Simple Poisson Assumption

A fixed decision rule that does not require updating at a later point it preferred by the stakeholders.

Axiom 1 (Static and Independent)**.** *The demand (s_i) for a given week of the year belong to the same static, Poisson distribution with a distinct rate parameter for each week.*

Example 1.
According to axiom 1, the sales from each week for a given brand and sub category belong to the same, static distribution. E.g. data from week 5 from 2021, 2022,... belong to a static distribution, and data from week 4 from 2021, 2022,... belong to another static distribution.

Using axiom 1

$$\begin{aligned} p(s_1, s_2, \dots s_n|D, I) &= p(s_1|s_2, \dots s_n, D, I)p(s_2|\dots s_n, D, I)\dots p(s_n|D, I) \\ &= \prod_{i=1}^{n} p(s_i|D, I), \end{aligned} \tag{2.7}$$

meaning the optimal decision rule for the Robot can be written

$$\begin{aligned} U_i^*(D) &= \mathbb{E}[S_i|D, I] \\ &= \int ds_i s_i p(s_i|D, I) \end{aligned} \tag{2.8}$$

According to axiom 1

$$\begin{aligned} p(s_i|D, I) &= \int d\lambda_i p(s_i, \lambda_i|D, I) \\ &= \int d\lambda_i p(s_i|\lambda_i, D, I)p(\lambda_i|D, I) \\ &= \int d\lambda_i p(s_i|\lambda_i, D, I)\frac{p(D|\lambda_i, I)p(\lambda_i|I)}{p(D|I)} \end{aligned} \tag{2.9}$$

where

$$p(D|I) = \int d\lambda_i p(D|\lambda_i, I)p(\lambda_i|I) \tag{2.10}$$

and

$$\begin{aligned} p(s_i|\lambda_i, D, I) &= p(s_i|\lambda_i, I) \\ &= \text{Poi}(s_i|\lambda_i). \end{aligned} \tag{2.11}$$

Note that the rate parameter λ_i is associated to the demand s_i, whereas the data contains measurements of sales y_j for j representing past weeks. Thus, in order to formulate the likelihood function, a relationship between sales y and demand s must be assumed. In general

Definition 1 (Demand and Sales)**.** *Let $m_{b,c,i}$ denote the stock of a given brand b and sub category c for week i, then*

$$y_{b,c,i} = \begin{cases} s_{b,c,i} & \text{if } s_{b,c,i} \leq m_{b,c,i} \\ m_{b,c,i} & \text{else} \end{cases}. \tag{2.12}$$

Axiom 2 (Never Out of Stock)**.** *For simplicity, products are assumed to be never of of stock, meaning (referring to definition 1) $y_{b,c,i} = s_{b,c,i}$.*

Using axioms 1 and 2, the likelihood function can be written

$$p(D|\lambda_i, I) = \prod_{j \in \text{week } i} \text{Poi}(y_j|\lambda_i), \tag{2.13}$$

where only data from the corresponding week is used in accordance with axiom 1. An obvious choice of prior would be the conjugate Gamma distribution (which is also the distribution with maximum entropy) with parameters α, β, meaning

$$p(\lambda_i|I) = \text{Ga}(\lambda_i|\alpha, \beta) \tag{2.14}$$

This means

$$\frac{p(D|\lambda_i, I)p(\lambda_i|I)}{p(D|I)} = \text{Ga}(\lambda_i|\alpha + N\bar{y}, \beta + n) \tag{2.15}$$

where

$$\bar{y} \equiv \frac{1}{n} \sum_{j \in \text{week } i} y_j. \tag{2.16}$$

Then

$$\begin{aligned} \mathbb{E}[s_i|D, I] &= \sum_{s_i} s_i \int d\lambda_i \text{Poi}(s_i|\lambda_i|)\text{Ga}(\lambda_i|\alpha + N\bar{y}, \beta + n_i) \\ &= \int d\lambda_i \lambda_i \text{Ga}(\lambda_i|\alpha + N\bar{y}, \beta + n_i) \\ &= \frac{\alpha + n_i\bar{y}_i}{\beta + n_i} \end{aligned} \tag{2.17}$$

In the limit of $\alpha, \beta \to 0$

$$\begin{aligned} U_i^*(D) &= \lim_{\alpha,\beta\to 0} \mathbb{E}[S_i|D,I] \\ &= \bar{y}_i, \end{aligned} \tag{2.18}$$

Thus, using axioms 1 and 2, a fixed decision rule yielding the forecasted demand for week i in the future, represented by Equation 2.18, is be obtained.

2.1.2 Advanced Poisson Assumption

Axiom 3 (Static with coupled rate parameter)**.** *The demand (s_i) for a given week of the year belong to the same static Poisson Distribution where the rate parameter is coupled between different weeks.*

Using axiom 3

$$\begin{aligned} p(s_1, s_2, \dots s_n|D,I) &= \int d\theta p(s_1, s_2, \dots s_n, \theta|D,I) \\ &= \int d\theta p(s_1, s_2, \dots s_n|\theta, D, I) p(\theta|D,I) \end{aligned} \tag{2.19}$$

where

$$\begin{aligned} p(s_1, s_2, \dots s_n|\theta, D, I) &= p(s_1, s_2, \dots s_n|\theta, I) \\ &= \prod_{j\in\text{forecast horizon}} \text{Poi}(s_j|\lambda(\theta, j)) \end{aligned} \tag{2.20}$$

and

$$p(\theta|D,I) = \frac{p(D|\theta,I)p(\theta|I)}{p(D|I)} \tag{2.21}$$

$$p(D|\theta,I) = \prod_{i\in\text{training data}} \text{Poi}(s_i|\lambda(\theta,i)). \tag{2.22}$$

This means

$$\begin{aligned} U_j^*(D) &= \int d\theta \int ds_1 ds_2 \dots ds_n s_j \text{Poi}(s_j|\lambda(\theta,j)) p(\theta|D,I) \\ &= \int d\theta \int ds_j s_j \text{Poi}(s_j|\lambda(\theta,j)) p(\theta|D,I) \\ &= \int d\theta \lambda(\theta,j) p(\theta|D,I) \\ &= \mathbb{E}[\lambda|D,I] \end{aligned} \tag{2.23}$$

where λ is some model, in this case a Fourier series

$$\lambda(\theta, j) = a_0 + \sum_{l=1}^{M} \left(a_l \cos\frac{2\pi}{N} j + b_l \sin\frac{2\pi}{N} j \right) \tag{2.24}$$

and $\theta = \{a, b\}$.

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.