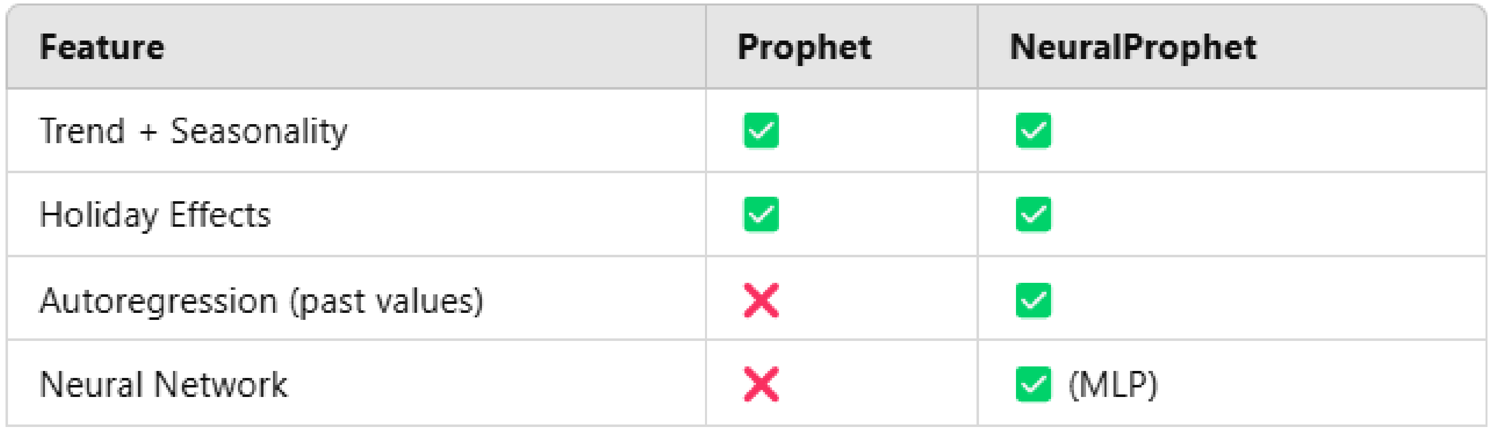
Road map!

- Module 1- Introduction to Deep Forecasting
- Module 2- Setting up Deep Forecasting Environment
- Module 3- Exponential Smoothing
- Module 4- ARIMA models
- Module 5- Machine Learning for Time series Forecasting
- Module 6- Deep Neural Networks
- Module 7- Deep Sequence Modeling (RNN, LSTM)
- Module 8- Prophet and Neural Prophet

Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Module 8- Prophet and Neural Prophet



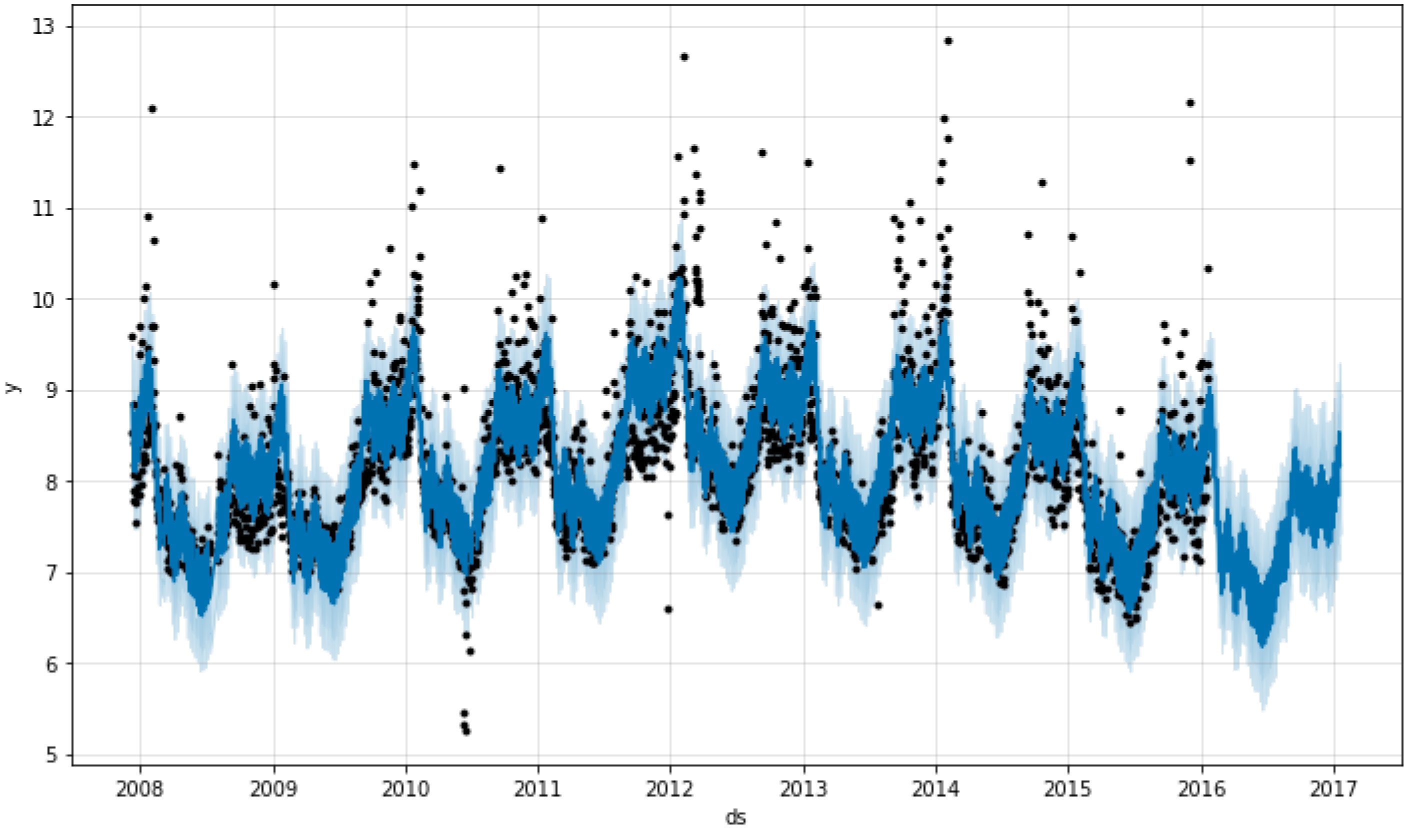
Feature	Prophet	NeuralProphet
Trend + Seasonality	✓	✓
Holiday Effects	✓	✓
Autoregression (past values)	✗	✓
Neural Network	✗	✓ (MLP)

Forecasting at Scale” (Taylor & Letham, 2017)

NeuralProphet: Explainable Forecasting at Scale” (Triebe et al., 2021)

Module 8- Part 1

PROPHET



Jon M. Huntsman School of Business
UtahStateUniversity

What is Facebook Prophet?

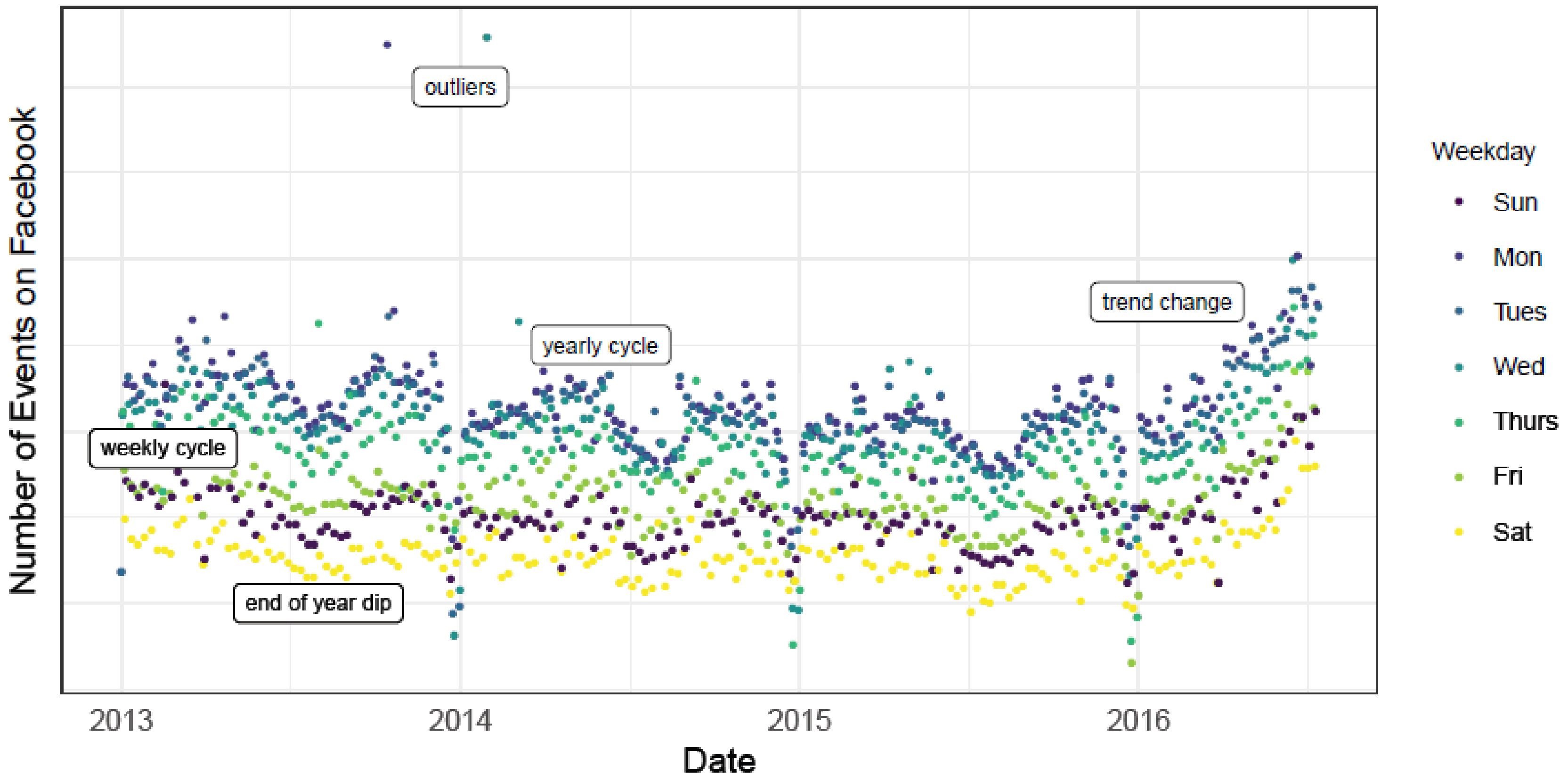
- Prophet is a robust open-source timeseries forecasting tool, available in Python and R.
- Developed by Sean J. Taylor and Benjamin Letham from Facebook (2017)
- Designed to produce forecasts **quickly** and reliably across a wide range of business applications.
- Forecasting at Scale:
    1. Large number of people making forecasts (possibly without training in ts methods)
    2. Large variety of forecasting problems
    3. Large number of forecasts (model evaluation and comparison while using human feedback to fix performance problems)
- Ideal for organizations that need to manage numerous and varied forecasting tasks simultaneously.

Key Features of Prophet

- Easy to Use with Analyst in the Loop: Prophet facilitates **active engagement** from analysts, requiring only basic familiarity with time series models while still benefiting from their domain expertise.
- Additive Model Components: Combines trends, seasonality, and holidays into an additive model that adjusts to changes in the time series.
- Automatic Trend and Seasonality Adjustments: Robust to dramatic shifts in trend and seasonality
- Robust to Data Anomalies: Prophet's non-recursive nature means it does not rely on lagged observations, which enhances its ability to handle datasets with missing entries. Additionally, its model formulation helps manage and mitigate the impact of outliers.

Business Time Series

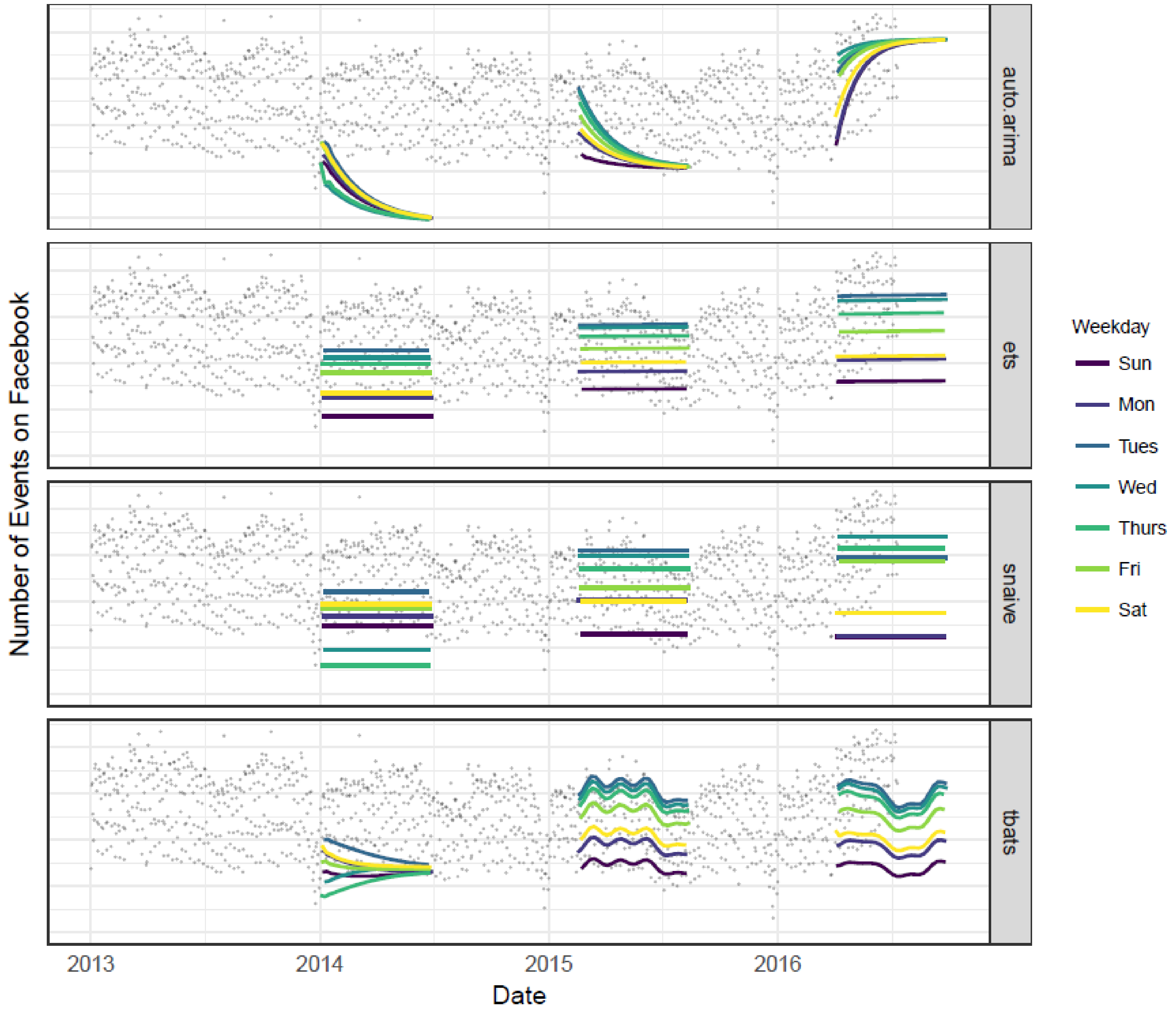
- The features of this time series are representative of many business time series: multiple strong seasonalities, trend changes, outliers, and holiday.



Benchmarks?!

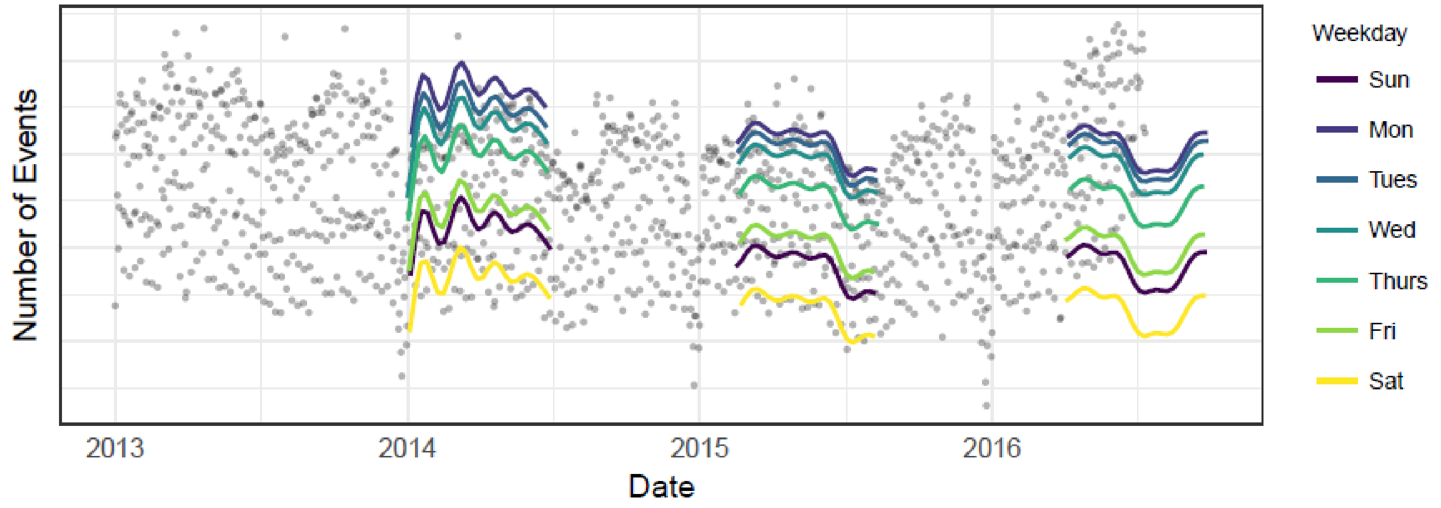
- Forecasts made at 3 points in the history, each using only the portion of the time series up to that point.
- (It seems that) other models have failed to:

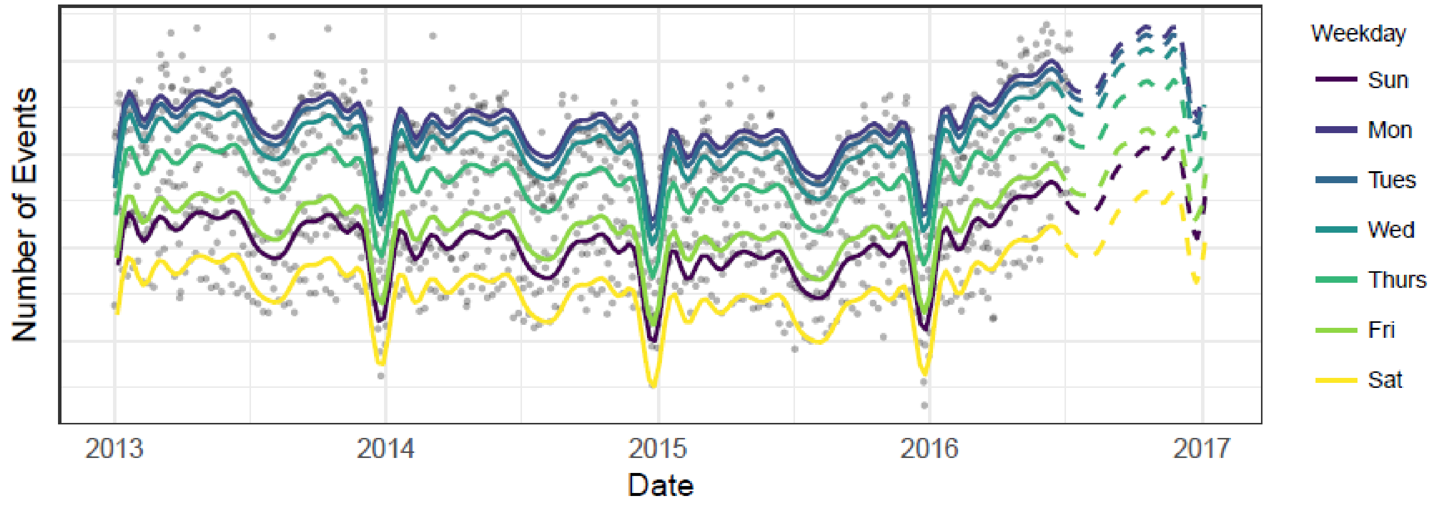
- Capture trend change near cut off points
- Capture mixed seasonalities
- Handle end-of-year dip



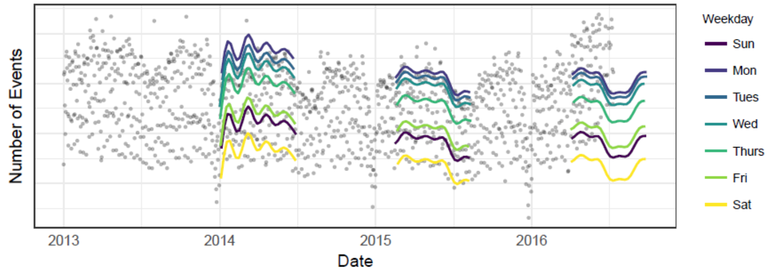
JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Business Time Series with Prophet





Business Time Series with Prophet



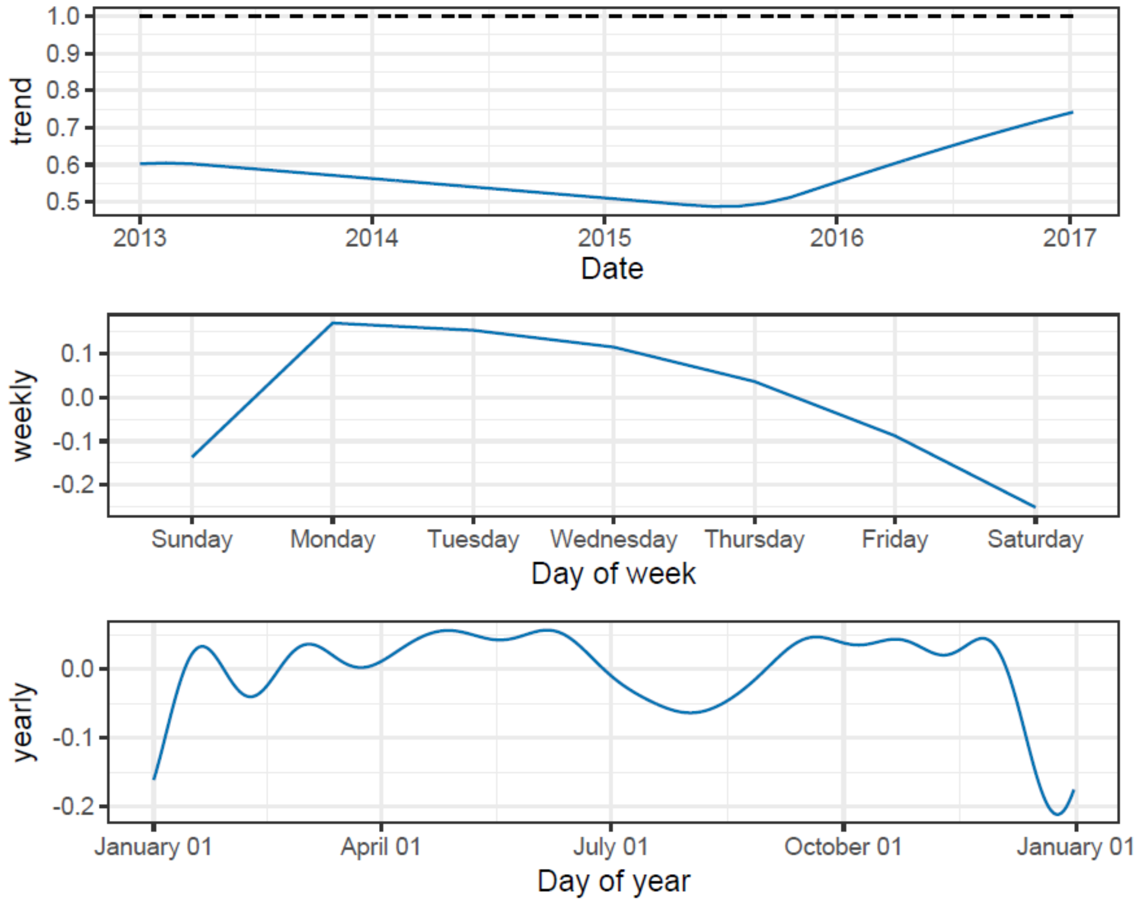


Figure 6: Components of the Prophet forecast in Fig. 5.

Core Components of Prophet

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- Trend Component: Captures non-periodic changes and identifies points where the time series have abrupt changes in trajectory.
- Seasonality Component: Models periodic changes using Fourier series to provide a flexible representation of periodic effects.
- Holidays Component: Allows for the specification of irregular events that can affect the time series.
- Error term: Represents any idiosyncratic changes not captured by the model.
- Similar to Generalized Additive Models (GAM) with Non-Recursive nature.

Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Core Components of Prophet

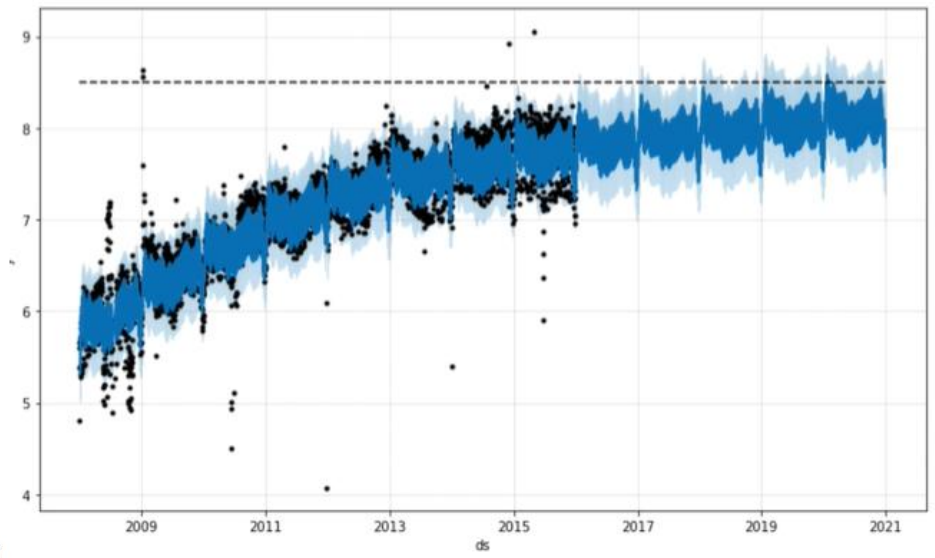
$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

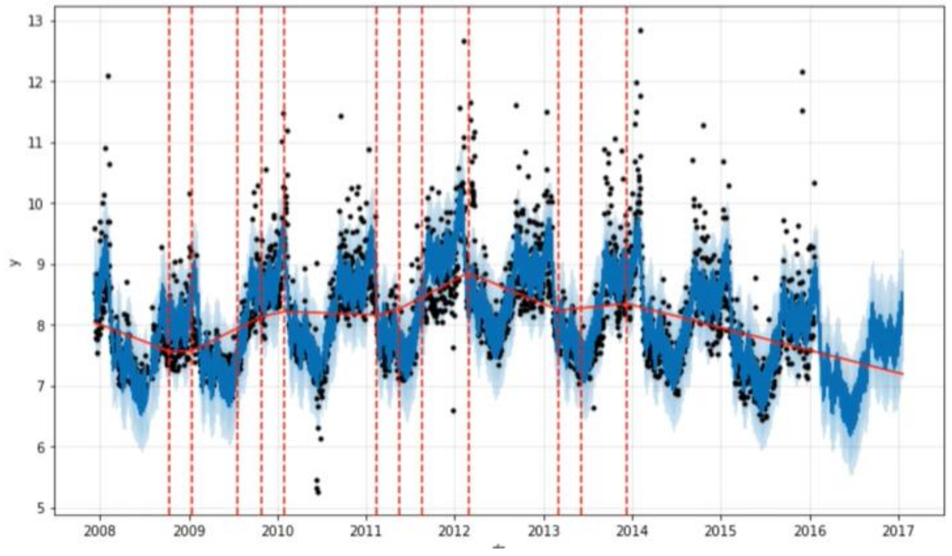
- In essence, framing the forecasting problem as a curve-fitting exercise.
- Inherently different from models that explicitly account for the temporal dependence structure in the data.
- Practical advantages are:
  - Flexibility
  - Doesn’t require regular time series → no need to interpolate missing values or outliers
  - Fast fitting → allows for exploring many model specifications
  - Interpretable (also easily extendable to include more components)

The Trend Model

- Prophet uses a piecewise linear or logistic growth curve to model non-periodic changes, making it adaptable to different growth scenarios.
- It implements two trend models
  1. Nonlinear, Saturating Growth
  2. Linear Trend with Changepoints

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$





1. Nonlinear, Saturating Growth

- Saturation: Accounts for carrying capacity in time series data.
- The most basic form is using the logistic growth model:

$$g(t) = \frac{C}{1 + \exp(-k(t - m))}$$

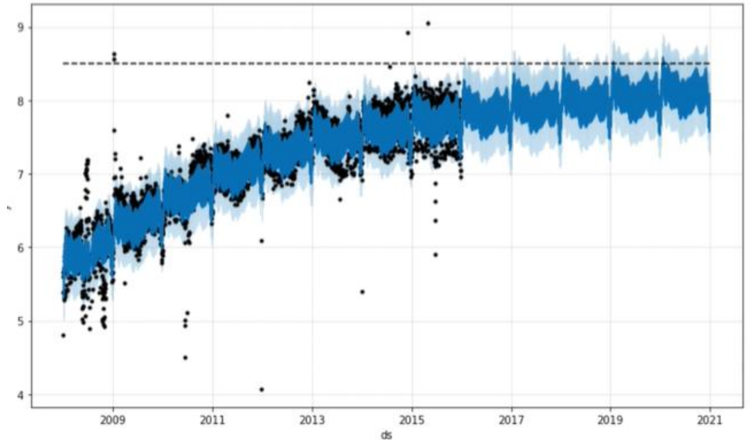
- C is the carrying capacity, k the growth rate, and m an offset parameter.
- Updates: Time varying Carrying capacity + Time varying Growth rate

$$g(t) = \frac{C(t)}{1 + \exp(-(k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})(t - (m + \mathbf{a}(t)^\intercal \boldsymbol{\gamma})))}$$

- This feature is particularly useful when forecasting business metrics that have natural saturation points, such as market size or total population.

Jon M. Huntsman School of Business UtahStateUniversity

Pedram Jahangiry



$$g(t) = \frac{C(t)}{1 + \exp(-(k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})(t - (m + \mathbf{a}(t)^\intercal \boldsymbol{\gamma})))}$$

JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

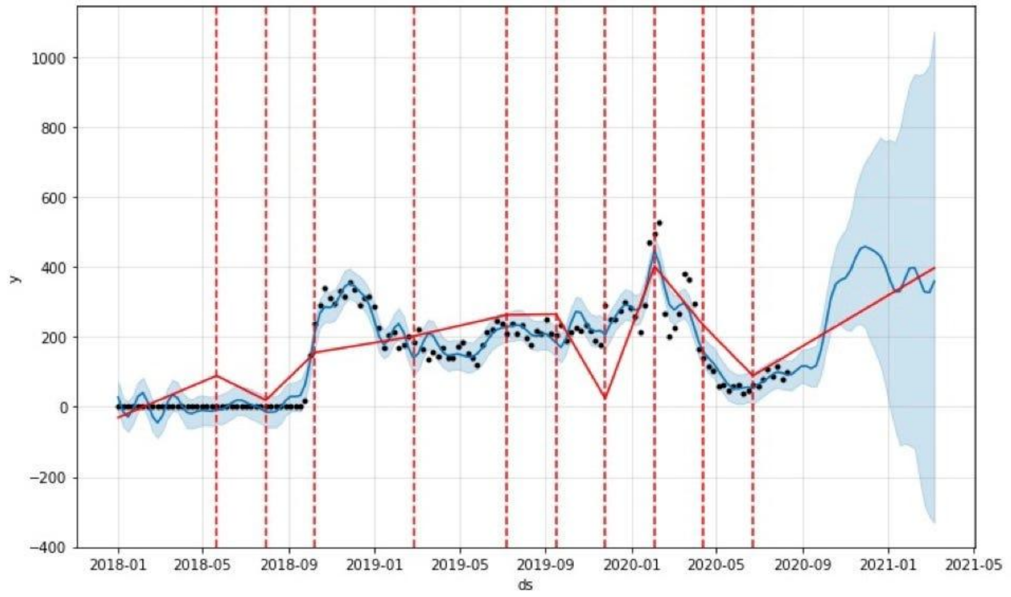
2. Linear Trend with Changepoints

- For forecasting problems that do not exhibit saturating growth, a piece-wise constant rate of growth provides a parsimonious and often useful model

$$g(t) = (k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})t + (m + \mathbf{a}(t)^\intercal \boldsymbol{\gamma})$$

- Changepoints could be specified by the analyst using known dates of product launches and other growth-altering events.

- Automatic changepoint selection could be done by selecting a set of prior candidates.



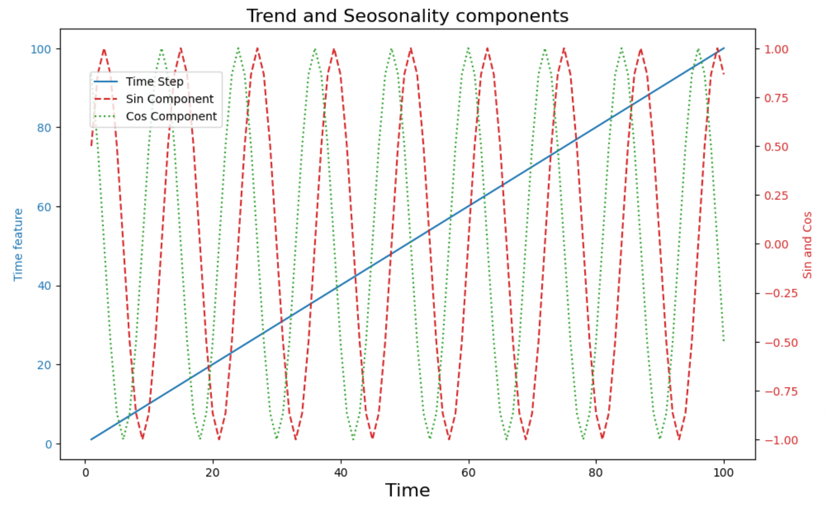
Seasonality Component

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- Business time series often have multi-period seasonality because of the human behaviors they represent (5-day work week, vacations, school breaks, etc.)
- Prophet handles daily, weekly, and yearly seasonality, as well as custom seasonalities, allowing accurate forecasting for various temporal granularities.
- This is done by specifying seasonality models that are periodic functions of t → Fourier Series approximating arbitrary smooth seasonal effects:

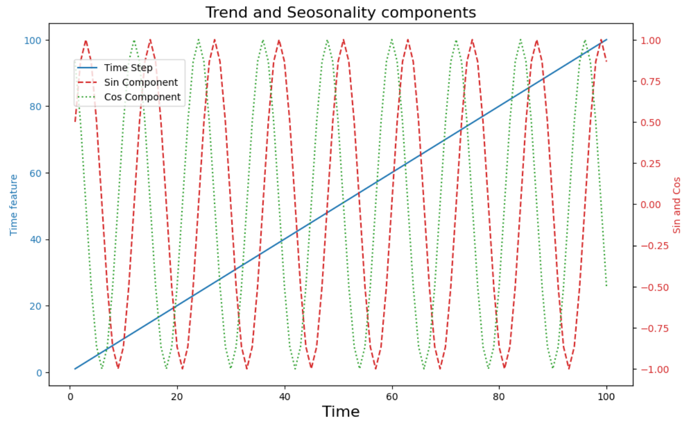
$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right)$$

- P is the regular period we expect the time series to have.



Seasonality Component

$$s(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right)$$



- P is the regular period we expect the time series to have.
- Fitting seasonality requires estimating 2N parameters $(a_1, \ldots a_n, b_1, \ldots, b_n)$
- This is done by constructing a matrix of seasonality vectors i.e. features!
- For example, with yearly seasonality and $N = 10$, the features are:

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{365.25}\right), \ldots, \sin\left(\frac{2\pi(10)t}{365.25}\right) \right]$$

- And the seasonal component is:

$$s(t) = X(t)\beta$$

Holiday Component

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- Holidays and events provide large, somewhat predictable shocks to many business time series and often do not follow a periodic pattern.
  - Thanksgiving in US on 4th Thursday in November
  - Super Bowl, Sunday in Jan or Feb.
  - Lunar calendar, etc.
- So, Holiday effects are not well modeled by a smooth cycle.

Holiday	Country	Year	Date
Thanksgiving	US	2015	26 Nov 2015
Thanksgiving	US	2016	24 Nov 2016
Thanksgiving	US	2017	23 Nov 2017
Thanksgiving	US	2018	22 Nov 2018
Christmas	*	2015	25 Dec 2015
Christmas	*	2016	25 Dec 2016
Christmas	*	2017	25 Dec 2017
Christmas	*	2018	25 Dec 2018

Holiday Component

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- Prophet allows the analyst to provide a custom list of past and future events.
- Assumption: the effects of holidays are independent.
- One-hot encoding every holiday:
  - For each holiday i, D_i is the set of past and future dates for that holiday.
  - $Z(t)$ is a matrix of regressors, including indicator functions representing if time t is during holiday i, and assign each holiday a parameter κ_i which is the corresponding change in the forecast.

$$Z(t) = [\mathbf{1}(t \in D_1), \ldots, \mathbf{1}(t \in D_L)]$$

- And the holiday component is: $h(t) = Z(t)\boldsymbol{\kappa}$

Jon M. Huntsman School of Business Utah State University

Pedram Jahangiry

Model Fitting

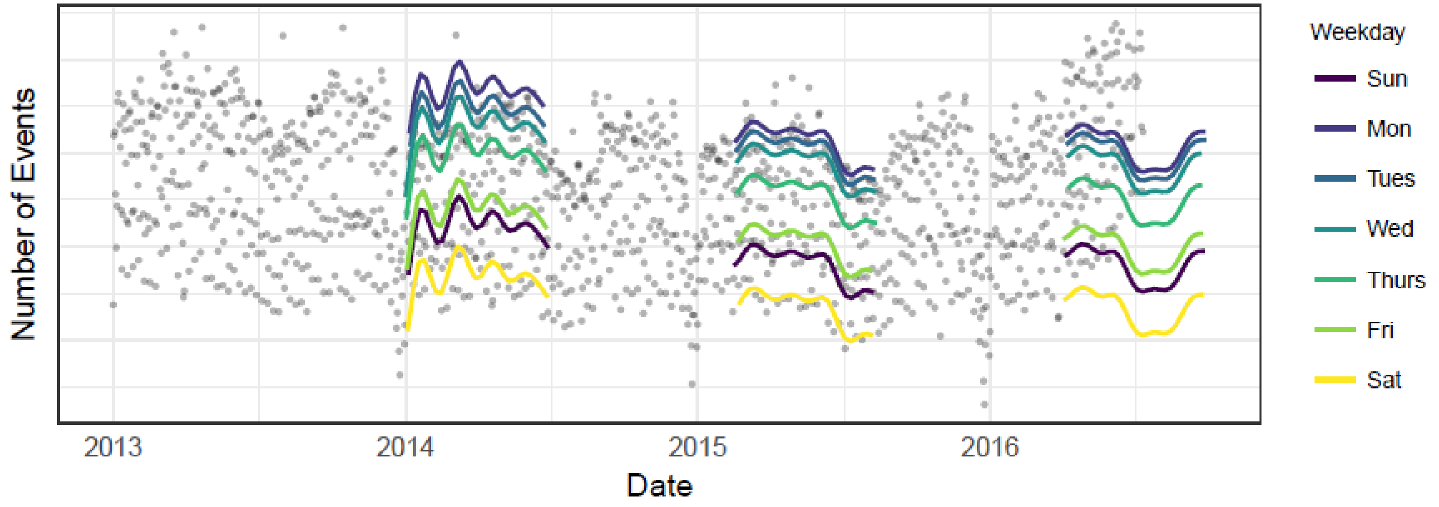
$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

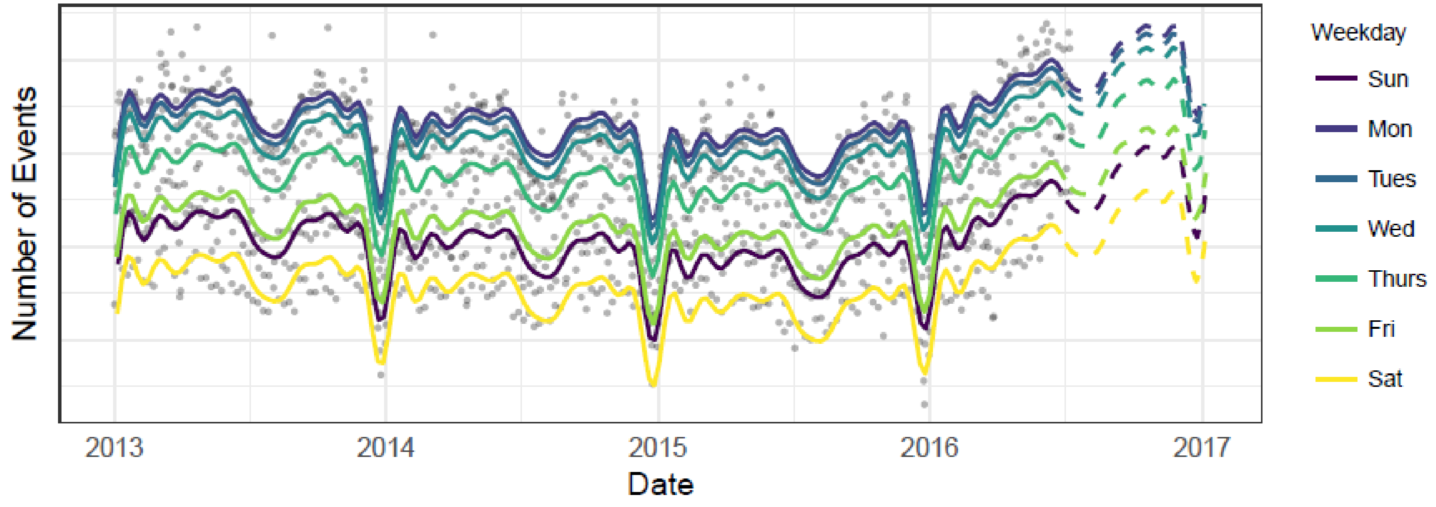
- **Matrix Formulation**: Seasonality, holidays, and change point effects are consolidated into matrix forms X and A, allowing the model to be efficiently expressed and computed.
- **Stan Code**: The entire forecasting model is implemented in Stan, a Bayesian library for statistical modeling and high-performance statistical computation.
- **Prior Selection**: Priors for the model parameters are chosen to reflect reasonable beliefs about the data without being overly restrictive. (including client on the modeling process)
- **Likelihood Functions**: Two forms of likelihood—logistic and linear—are used to capture different types of data behavior, with normal distributions assumed for the noise.
- **Full Posterior Inference**: Stan is also capable of conducting full posterior inference to estimate the uncertainty in the model parameters.

Listing 1: Example Stan code for our complete model.

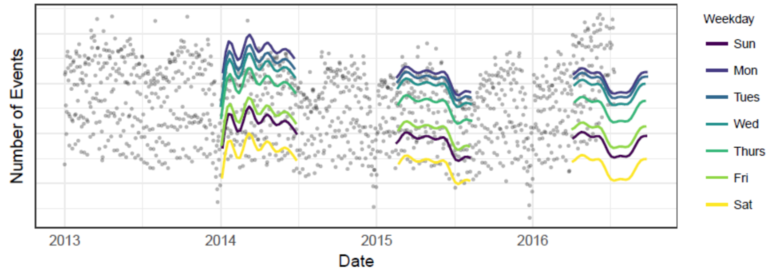
```
model {
  // Priors
  k ~ normal(0, 5);
  m ~ normal(0, 5);
  epsilon ~ normal(0, 0.5);
  delta ~ double_exponential(0, tau);
  beta ~ normal(0, sigma);

  // Logistic likelihood
  y ~ normal(C ./ (1 + exp(-(k + A * delta) .* (t - (m + A * gamma)))) +
             X * beta, epsilon);
  // Linear likelihood
  y ~ normal((k + A * delta) .* t + (m + A * gamma) + X * beta, sigma);
}
```

Business Time Series with Prophet





Business Time Series with Prophet



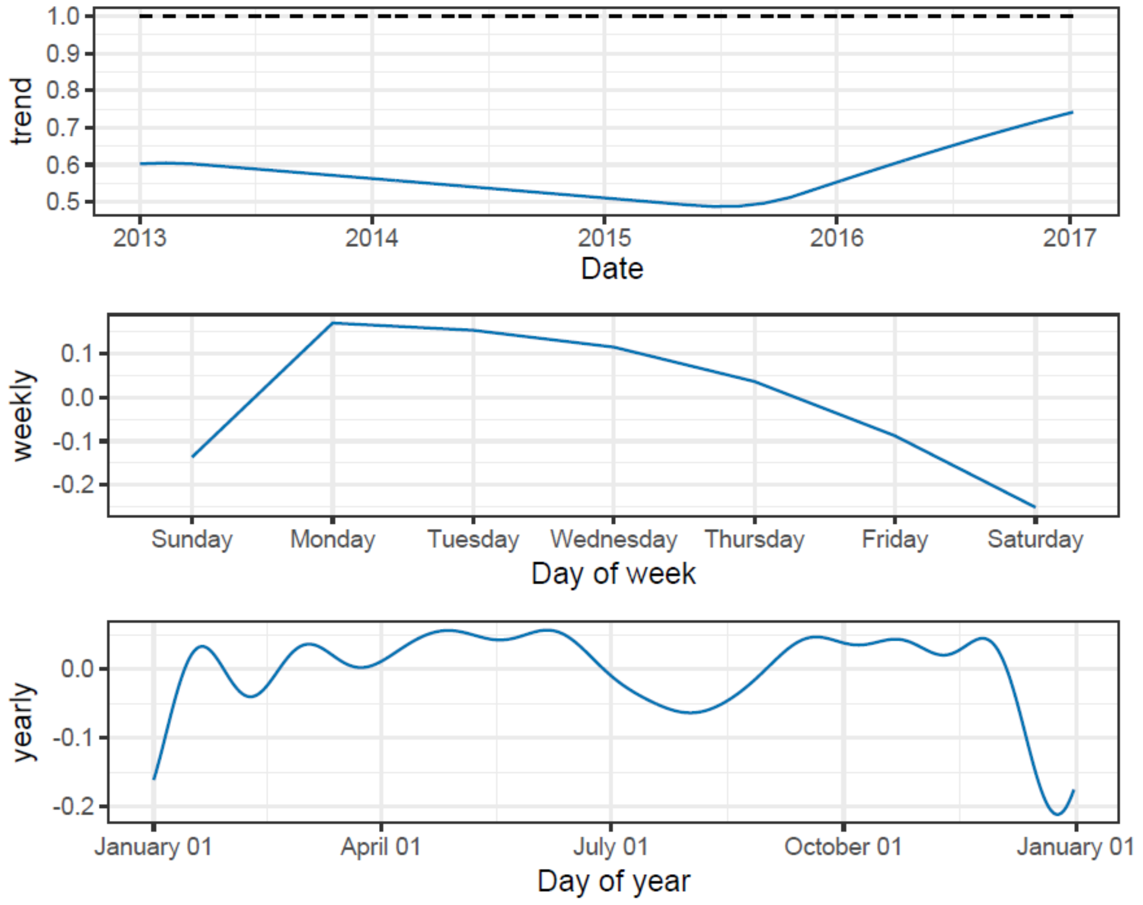


Figure 6: Components of the Prophet forecast in Fig. 5.

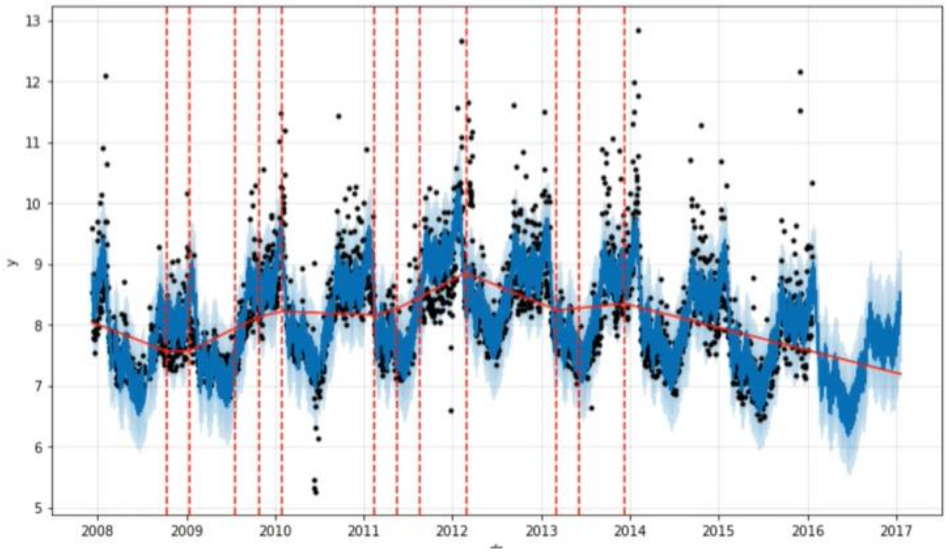
The Role of the Analyst in Prophet's Forecasting

- ✓ Domain Knowledge Utilization: Analysts can input their extensive domain knowledge into the Prophet model to refine forecasts without needing statistical expertise.
  - ✓ Capacities: Integration of market size data to set the carrying capacity for forecasts.
  - ✓ Changepoints: Direct specification of known events that impact the time series.
  - ✓ Holidays and Seasonality: Input of regional growth impacts from holidays based on experience.

- ✓ Model Adjustment via Parameters

The Role of the Analyst in Prophet's Forecasting

- Domain Knowledge Utilization
- Model Adjustment via Parameters: Parameters in Prophet allow analysts to apply their insights to adjust the model's sensitivity to trends and seasonality.
  - Smoothing Parameters: Adjusting priors to change trend flexibility and influence the strength of seasonality.
  - Visualization Tools: Essential for validating model fit and identifying areas for refinement.



Conclusion

- Prophet's Innovation: Analyst-in-the-Loop Modeling Approach, combines the precision of statistical forecasts with the nuanced insights of judgmental forecasts.
- Versatile Model Adjustments: Analysts can apply their domain knowledge through intuitive parameters affecting capacities, changepoints, and seasonalities.
- Automated vs. Judgmental Forecasting: Prophet marries the scalability of automated forecasting with the depth of judgmental forecasting.
- Automated Forecast Evaluation: Critical for scaling analyst-in-the-loop, automated tools assess forecast quality, guiding where human input is most beneficial.

PROPHET

Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Limitations of Prophet

- Lacks Local Context: Cannot learn short-term autoregressive patterns (no use of recent lagged values).
- Challenging to Extend: Built on Stan (probabilistic programming language), making customization and scaling harder.
- Assumes Additivity (or simple multiplicativity): Cannot easily handle complex interactions between components.
- Error Accumulation in Multi-step Forecasts: Must use recursive prediction, which can lead to compounding errors.

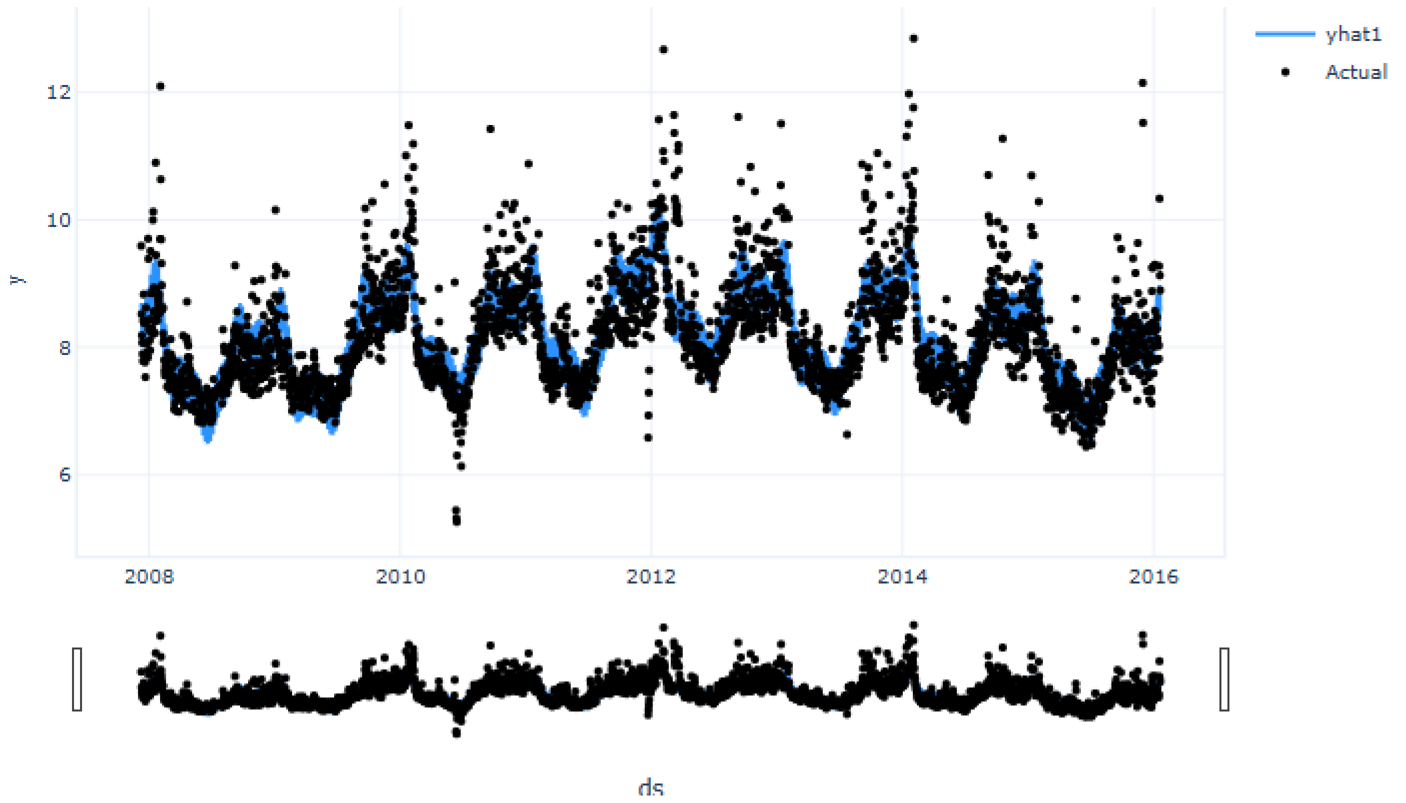
PROPHET

JON M. HUNTSMAN SCHOOL OF BUSINESS UtahStateUniversity

Pedram Jahangiry

Module 8- Part 2

Neural Prophet



NeuralProphet - Model Intuition

What is NeuralProphet ? Explainable forecasting at scale (2020)

- Designed for industrial timeseries forecasting, just like Facebook Prophet
  - Explainable, scalable, minimal tuning, reasonably accurate
- A hybrid forecasting framework
- Based on PyTorch, open-source and customizable
- Adds power and flexibility:
  - Local context is introduced with AR and covariate modules
  - Extra regressors can be modeled as linear regression or as Neural Networks.

NeuralProphet - Model Components

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t)$$

$T(t)$ = Trend at time t

$S(t)$ = Seasonal effects at time t

$E(t)$ = Event and holiday effects at time t

$F(t)$ = Regression effects at time t for future-known exogenous variables

$A(t)$ = Auto-regression effects at time t based on past observations

$L(t)$ = Regression effects at time t for lagged observations of exogenous variables

- All modules must produce h outputs! This allows multi-output forecasting

Trend $T(t)$

- Trend is modeled as a continuous piece-wise linear series.
- A set of n_c changepoints are defined at different times
- Time-dependent growth rate $\rho(t)$ and offset parameter $\delta(t)$
- Between changepoints, growth rate is kept constant
- Unlike Prophet, NP doesn't allow for non-linear, saturating trend.

$$\delta = (\delta_1, \delta_2, \dots, \delta_{n_C})$$

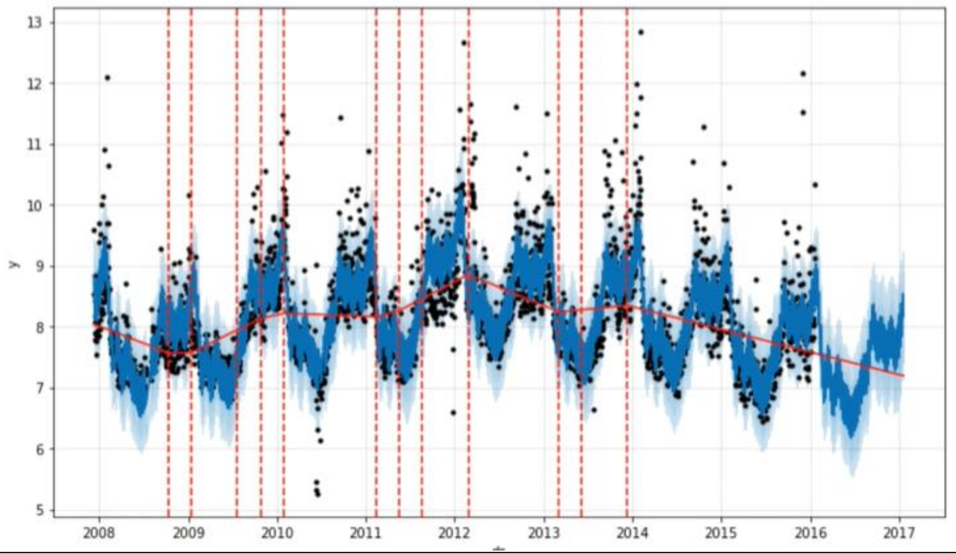
$$\rho = (\rho_1, \rho_2, \dots, \rho_{n_C})$$

$$\Gamma(t) = (\Gamma_1(t), \Gamma_2(t), \dots, \Gamma_{n_c}(t))$$

$$\Gamma_j(t) = \begin{cases} 1, & \text{if } t \geq c_j \\ 0, & \text{otherwise} \end{cases}$$

$$\hat{y}_t = \boxed{T(t)} + S(t) + E(t) + F(t) + A(t) + L(t)$$

$$\boxed{T(t)} = (\delta_0 + \Gamma(t)^T \delta) \cdot t + (\rho_0 + \Gamma(t)^T \rho)$$



Jon M. Huntsman School of Business Utah State University

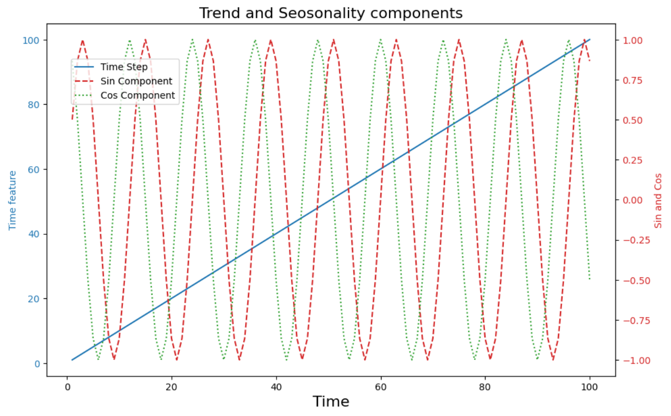
Explainable forecasting at scale (2020)

Seasonality $S(t)$

- Just like Prophet, Fourier terms are defined for each seasonality

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t)$$

$$S_p(t) = \sum_{j=1}^{k} \left(a_j \cdot cos\left(\frac{2\pi jt}{p}\right) + b_j \cdot sin\left(\frac{2\pi jt}{p}\right)\right)$$



- A higher number of Fourier terms, allows to fit more complex seasonal pattern
- Both additive and multiplicative seasonal patterns are supported.

$$S(t) = \sum_{p \in \mathbb{P}} S_p^{\star}(t) \qquad S_p^{\star}(t) = \begin{cases} S_p^{\dagger}(t) = T(t) \cdot S_p(t), & \text{if } S_p \text{ is multiplicative} \\ S_p(t), & \text{otherwise} \end{cases}$$

- The framework automatically activate daily, weekly and or yearly depending on data frequency.

Jon M. Huntsman School of Business
UtahStateUniversity

Pedram Jahangiry

Auto-Regressive $A(t)$

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t)$$

$$y_t = c + \sum_{i=1}^{i=p} \theta_i \cdot y_{t-i} + e_t$$

- Classic AR(p) process!
- Modified version of AR-Net model (2019)
- Able to produce multi-output (*h* forecasts) with one model.
- Linear AR vs Deep AR (Regularization can be used to sparsify the model weights)

Lagged Regressors $L(t)$

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t)$$

- Lagged regressors (covariates) are used to correlate other variables to our target variable
- At time t of forecasting, the future of these regressors is unknown
- Each lagged regressor module is like the AR(p) module.

$$L(t) = \sum_{x \in \mathbb{X}} L_x(x_{t-1}, x_{t-2}, ..., x_{t-p})$$

- Each module producing h additive components to the overall forecasts (at the same time)

$$L_x^t(t), L_x^t(t+1), ..., L_x^t(t+h-1) = \text{AR-Net}(x_{t-1}, x_{t-2}, ..., x_{t-p})$$

Future Regressors F(t)

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t)$$

- Both past and future values of these regressors are known.
- The effect of all future regressors at time t can be denoted by $F(t)$

$$F_f(t) = d_f f(t)$$

$$F(t) = \sum_{f \in \mathbb{F}} F_f^\star(t)$$

- Future regressors can be modeled as both additive and multiplicative

$$F_f^\star(t) = \begin{cases} F_f^\dagger(t) = T(t) \cdot F_f(t), & \text{if } f \text{ is multiplicative} \\ F_f(t), & \text{otherwise} \end{cases}$$

Events and Holidays E(t)

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t)$$

- Effects from special events or holidays may occur sporadically.
- Events are models analogous to the future regressors, with each event e, as a binary variable. The effect of all events at time t can be denoted by $E(t)$

$$E_e(t) = z_e e(t) \qquad\qquad E(t) = \sum_{e \in \mathbb{E}} E_e^{\star}(t)$$

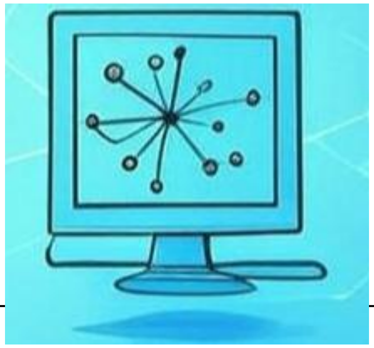
- Events can be modeled as both additive and multiplicative

$$E_e^{\star}(t) = \begin{cases} E_e^{\dagger}(t) = T(t) \cdot E_e(t), & \text{if } e \text{ is multiplicative} \\ E_e(t), & \text{otherwise} \end{cases}$$

Preprocessing, Training, Postprocessing

Preprocessing

- Missing Data (imputation)
- Data Normalization
- Data Tabularization



Training

- Loss Function
- Regularization
- Optimizer
- Learning Rate
- Batch size
- Training Epochs

Postprocessing

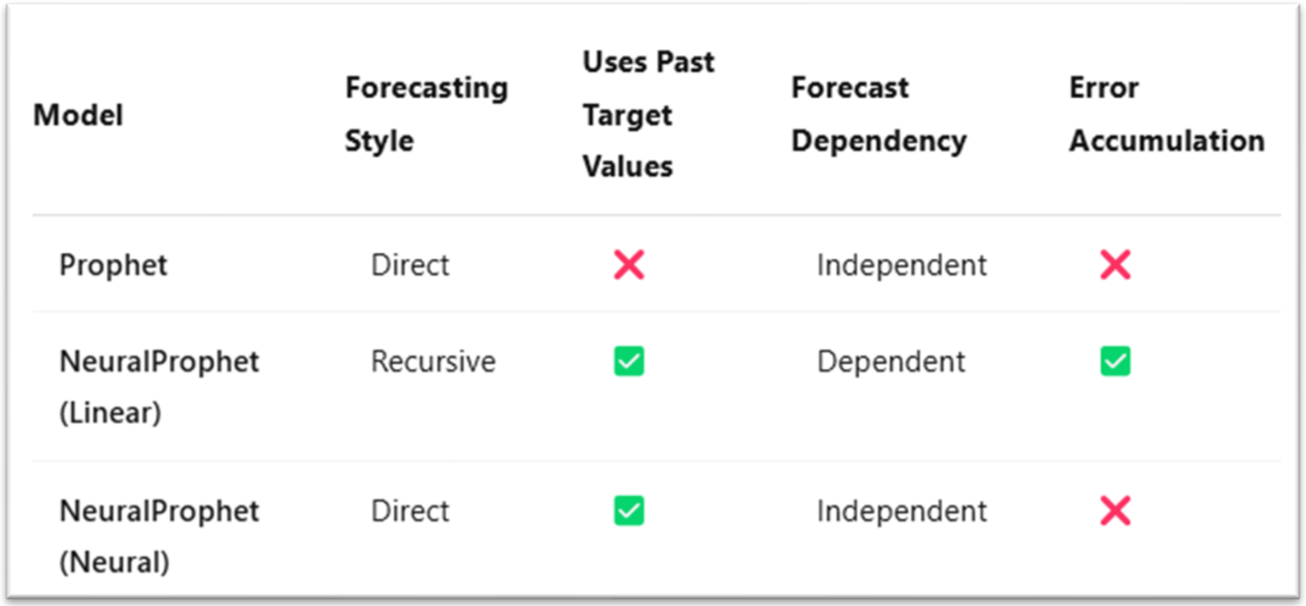
- Transforming back
- Metrics
- Forecast presentation

Jon M. Huntsman School of Business
Utah State University

Pedram Jahangiry

Multi-step Forecasting – Prophet vs NeuralProphet

- Prophet predicts multiple future points in one go, using calendar-based components only (trend, seasonality, holiday).
- NeuralProphet uses past lagged values. If configured with a neural AR-Net, it can predict multiple future steps directly.



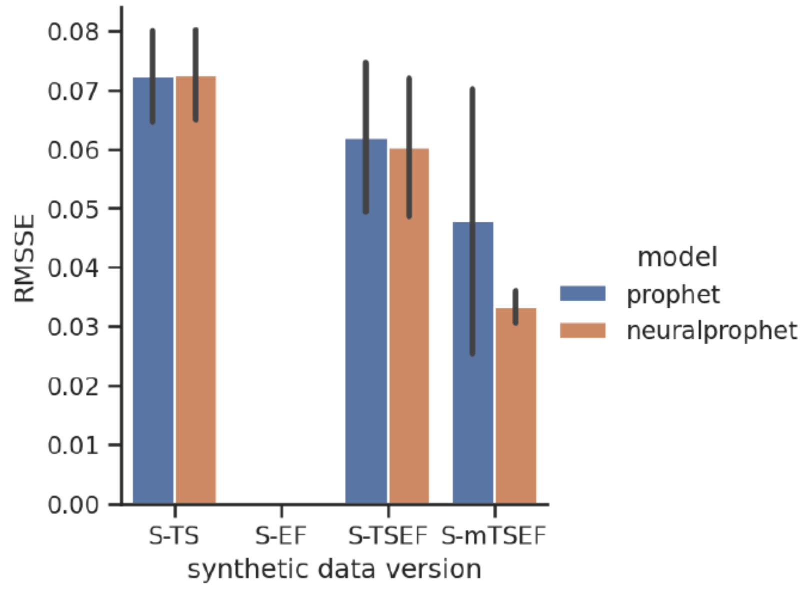
Model	Forecasting Style	Uses Past Target Values	Forecast Dependency	Error Accumulation
Prophet	Direct	❌	Independent	❌
NeuralProphet (Linear)	Recursive	✅	Dependent	✅
NeuralProphet (Neural)	Direct	✅	Independent	❌

- Linear AR configuration in NeuralProphet predicts recursively — each step depends on the previous prediction, which can lead to error accumulation.

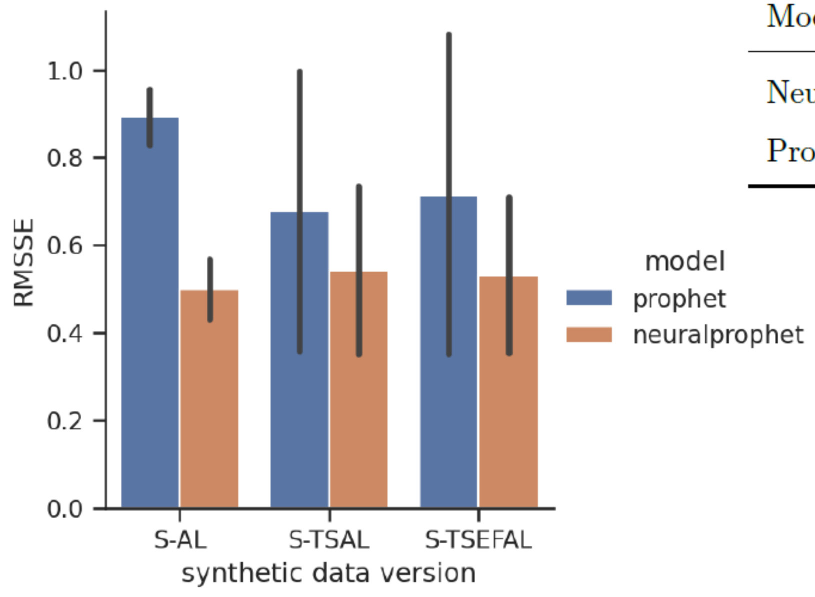
Prophet vs NeuralProphet performance (synthetic data)

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t)$$

Model	Training Time (seconds)	Prediction Time (seconds)
NeuralProphet	20.50 ($\pm$4.70)	**0.16** ($\pm$0.05)
Prophet	**5.07** ($\pm$2.30)	2.16 ($\pm$1.08)



(a) Experiments without lagged components.



(b) Experiments including lagged components.

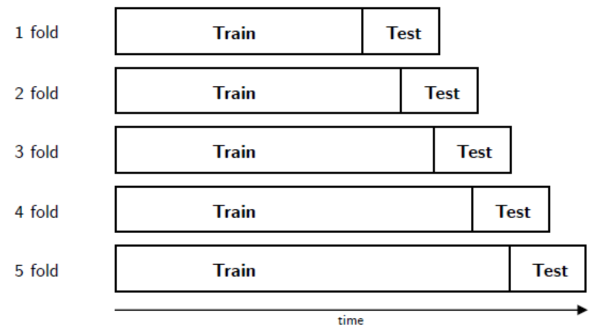


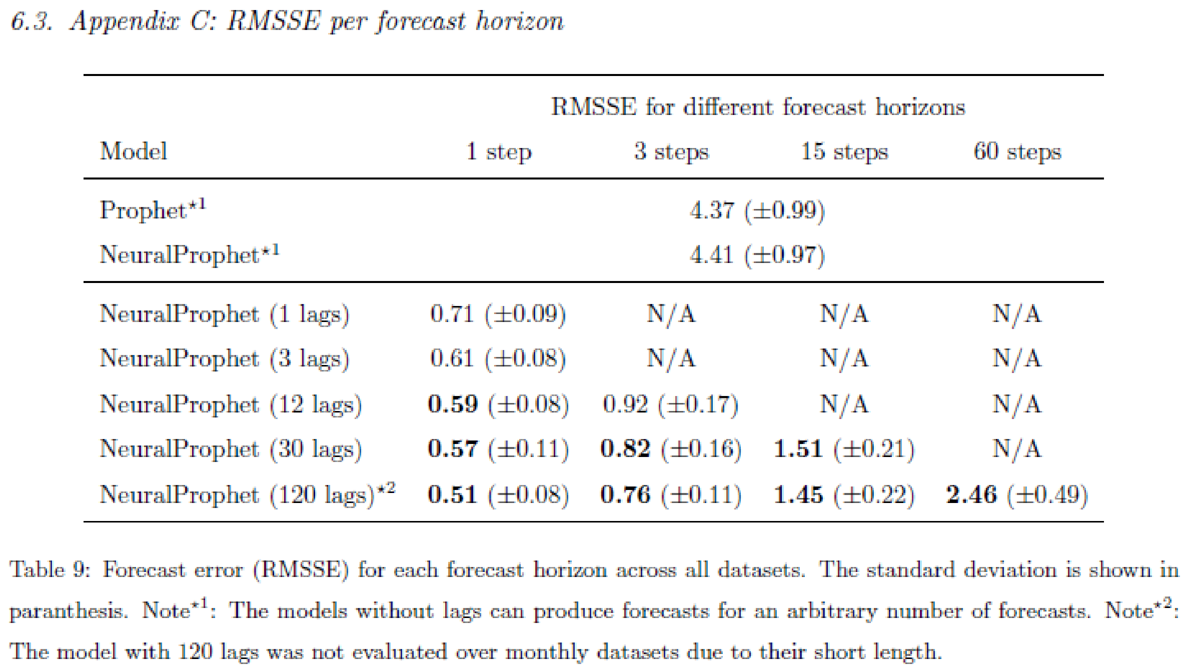
Figure 4: RMSSE error of the model fit on the synthetic time-series y itself. The plotted bar displays mean value and the line displays standard deviation over 5 runs.

JON M. HUNTSMAN SCHOOL OF BUSINESS
UtahStateUniversity

Pedram Jahangiry

Explainable forecasting at scale (2020)

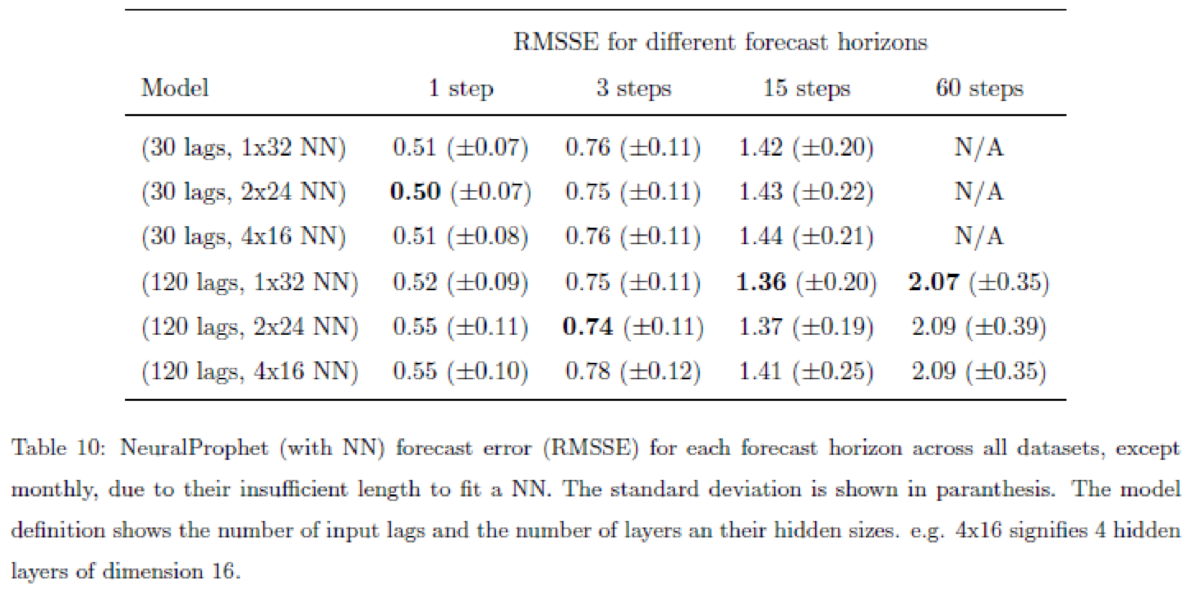
Prophet vs NeuralProphet performance (synthetic data)



6.3. Appendix C: RMSSE per forecast horizon

	RMSSE for different forecast horizons			
Model	1 step	3 steps	15 steps	60 steps
Prophet*¹	4.37 (±0.99)			
NeuralProphet*¹	4.41 (±0.97)			
NeuralProphet (1 lags)	0.71 (±0.09)	N/A	N/A	N/A
NeuralProphet (3 lags)	0.61 (±0.08)	N/A	N/A	N/A
NeuralProphet (12 lags)	**0.59** (±0.08)	0.92 (±0.17)	N/A	N/A
NeuralProphet (30 lags)	**0.57** (±0.11)	**0.82** (±0.16)	**1.51** (±0.21)	N/A
NeuralProphet (120 lags)*²	**0.51** (±0.08)	**0.76** (±0.11)	**1.45** (±0.22)	**2.46** (±0.49)

Table 9: Forecast error (RMSSE) for each forecast horizon across all datasets. The standard deviation is shown in paranthesis. Note*¹: The models without lags can produce forecasts for an arbitrary number of forecasts. Note*²: The model with 120 lags was not evaluated over monthly datasets due to their short length.



	RMSSE for different forecast horizons			
Model	1 step	3 steps	15 steps	60 steps
(30 lags, 1x32 NN)	0.51 (±0.07)	0.76 (±0.11)	1.42 (±0.20)	N/A
(30 lags, 2x24 NN)	**0.50** (±0.07)	0.75 (±0.11)	1.43 (±0.22)	N/A
(30 lags, 4x16 NN)	0.51 (±0.08)	0.76 (±0.11)	1.44 (±0.21)	N/A
(120 lags, 1x32 NN)	0.52 (±0.09)	0.75 (±0.11)	**1.36** (±0.20)	**2.07** (±0.35)
(120 lags, 2x24 NN)	0.55 (±0.11)	**0.74** (±0.11)	1.37 (±0.19)	2.09 (±0.39)
(120 lags, 4x16 NN)	0.55 (±0.10)	0.78 (±0.12)	1.41 (±0.25)	2.09 (±0.35)

Table 10: NeuralProphet (with NN) forecast error (RMSSE) for each forecast horizon across all datasets, except monthly, due to their insufficient length to fit a NN. The standard deviation is shown in paranthesis. The model definition shows the number of input lags and the number of layers an their hidden sizes. e.g. 4x16 signifies 4 hidden layers of dimension 16.

1. Prophet and NeuralProphet perform near identical in their default mode.
2. NeuralProphet forecasts improve substantially when configuring any amount of lags
3. More lags generally lead to a better performance
4. all NN configuration Significantly outperform their linear counterpart.

$$MASE = \frac{\frac{1}{J}\sum_{j=T+1}^{T+J}|y_i - \hat{y}_i|}{\frac{1}{T-1}\sum_{i=2}^{T}|y_i - y_{i-1}|}$$

$$RMSSE = \frac{\sqrt{\frac{1}{J}\sum_{j=T+1}^{T+J}(y_i - \hat{y}_i)^2}}{\sqrt{\frac{1}{T-1}\sum_{i=2}^{T}(y_i - y_{i-1})^2}}$$

Explainable forecasting at scale (2020)

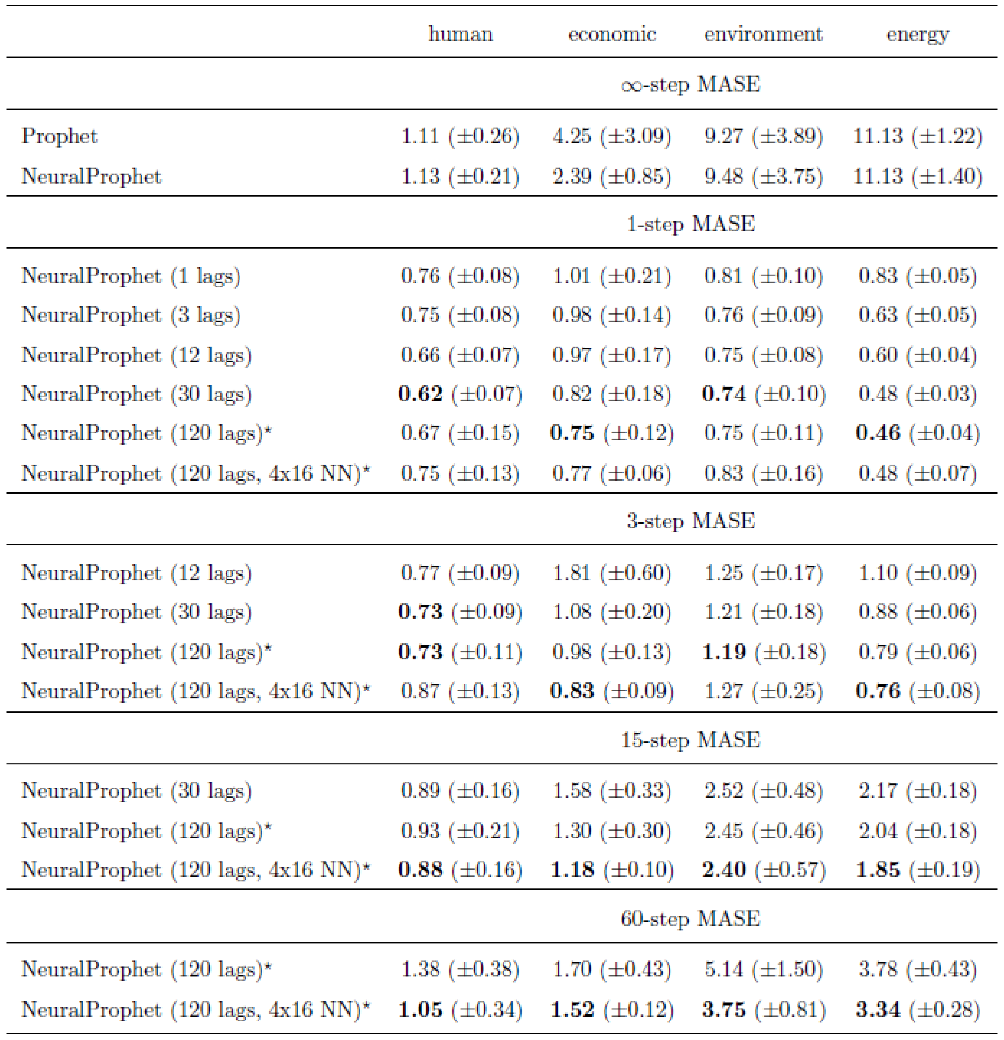
Prophet vs NeuralProphet performance (Real-world data)

- Human Data: Tracks behavior or activity of individuals, such as tourism counts (e.g., air_passengers), online interactions (e.g., peyton_manning), or population trends (e.g., birth records).
- Economic Data: Involves business or financial metrics, such as retail sales volume or energy market prices (e.g., retail_sales, price_ercot).
- Energy Data: Represents electricity or power-related time series, including demand loads (e.g., load_ercot), solar and wind generation (e.g., power_solar, power_wind).
- Environmental Data: Captures natural or atmospheric phenomena like solar activity (e.g., sunspot, solar_sf), weather (e.g., yosemite.temps), water levels, or air quality.

Name	Type	Subtype	Freq	T
air_passengers	human	tourism	M	144
retail_sales	economic	sales	M	293
peyton_manning	human	clicks	D	2905
birth	human	population	D	7305
load_hospital	energy	load	H	8760
solar_sf	environment	sun	H	8,760
load_rte	energy	load	H	17518
load_victoria	energy	load	30min	17520
yosemite_temps	environment	weather	5min	18721
river	environment	water	D	23741
price_ercot	economic	energy price	H	25537
air_beijing	environment	air quality	H	43800
sunspot	environment	sun	D	73924
load_ercot	energy	load	H	154872
power_wind	energy	wind	1min	493144
power_solar	energy	solar	1min	493149

Prophet vs NeuralProphet performance (Real-world data)

- 🔺 MASE increases with forecasting horizon. longer-term forecasts are naturally harder.
- 🧱 More lags generally improve performance,
- 🤖 Deep models outperform linear ones at longer horizons due to their ability to learn complex patterns.
- 🔁 For one-step ahead (or short horizon) forecasts, a linear AR model with sufficient lags often works best.



	human	economic	environment	energy
∞-step MASE				
Prophet	1.11 (±0.26)	4.25 (±3.09)	9.27 (±3.89)	11.13 (±1.22)
NeuralProphet	1.13 (±0.21)	2.39 (±0.85)	9.48 (±3.75)	11.13 (±1.40)
1-step MASE				
NeuralProphet (1 lags)	0.76 (±0.08)	1.01 (±0.21)	0.81 (±0.10)	0.83 (±0.05)
NeuralProphet (3 lags)	0.75 (±0.08)	0.98 (±0.14)	0.76 (±0.09)	0.63 (±0.05)
NeuralProphet (12 lags)	0.66 (±0.07)	0.97 (±0.17)	0.75 (±0.08)	0.60 (±0.04)
NeuralProphet (30 lags)	**0.62** (±0.07)	0.82 (±0.18)	**0.74** (±0.10)	0.48 (±0.03)
NeuralProphet (120 lags)*	0.67 (±0.15)	**0.75** (±0.12)	0.75 (±0.11)	**0.46** (±0.04)
NeuralProphet (120 lags, 4x16 NN)*	0.75 (±0.13)	0.77 (±0.06)	0.83 (±0.16)	0.48 (±0.07)
3-step MASE				
NeuralProphet (12 lags)	0.77 (±0.09)	1.81 (±0.60)	1.25 (±0.17)	1.10 (±0.09)
NeuralProphet (30 lags)	**0.73** (±0.09)	1.08 (±0.20)	1.21 (±0.18)	0.88 (±0.06)
NeuralProphet (120 lags)*	**0.73** (±0.11)	0.98 (±0.13)	**1.19** (±0.18)	0.79 (±0.06)
NeuralProphet (120 lags, 4x16 NN)*	0.87 (±0.13)	**0.83** (±0.09)	1.27 (±0.25)	**0.76** (±0.08)
15-step MASE				
NeuralProphet (30 lags)	0.89 (±0.16)	1.58 (±0.33)	2.52 (±0.48)	2.17 (±0.18)
NeuralProphet (120 lags)*	0.93 (±0.21)	1.30 (±0.30)	2.45 (±0.46)	2.04 (±0.18)
NeuralProphet (120 lags, 4x16 NN)*	**0.88** (±0.16)	**1.18** (±0.10)	**2.40** (±0.57)	**1.85** (±0.19)
60-step MASE				
NeuralProphet (120 lags)*	1.38 (±0.38)	1.70 (±0.43)	5.14 (±1.50)	3.78 (±0.43)
NeuralProphet (120 lags, 4x16 NN)*	**1.05** (±0.34)	**1.52** (±0.12)	**3.75** (±0.81)	**3.34** (±0.28)

Table 6: Forecast error (MASE) for each dataset subject type for different forecast horizons. The standard deviation is shown in paranthesis. Note*: Models with 120 lags were not evaluated over monthly datasets due to their short length.

Minimal example

```
pip install neuralprophet
```

```
from neuralprophet import NeuralProphet

# Define and fit the model
m = NeuralProphet()
metrics = m.fit(df)

# Create future dataframe and forecast
future = m.make_future_dataframe(df, periods=12)
forecast = m.predict(future)

# Plotting
fig_forecast = m.plot(forecast)
fig_components = m.plot_components(forecast)
fig_model = m.plot_parameters()
```

https://neuralprophet.com/how-to-guides/feature-guides/Migration_from_Prophet.html

Takeaway

Task	Prophet	NeuralProphet
Small dataset ($T \ll 100$)	×	
Medium to large dataset ($T \gg 100$)		×
Long range forecast ($h \gg 100$)	×	×
Short to medium range forecast ($1 \leq h \ll 1000$)		×
Specific forecast horizon (e.g. $h = 24$)		×
Auto-correlation (dependence on previous observations)		×
Lagged regressors (dependence on observed covariates)		×
Non-linear dynamics		×
Global modelling of panel dataset		×
Frequent retraining on small datasets with constrained computational resources	×	
Fast prediction (computational inference time)		×

Table 7: Comparison of Prophet's and NeuralProphet's task-dependent strength. × marks which of the two models we suggest to use for the given task.

Road map!

- ✓ Module 1- Introduction to Deep Forecasting
- ✓ Module 2- Setting up Deep Forecasting Environment
- ✓ Module 3- Exponential Smoothing
- ✓ Module 4- ARIMA models
- ✓ Module 5- Machine Learning for Time series Forecasting
- ✓ Module 6- Deep Neural Networks
- ✓ Module 7- Deep Sequence Modeling (RNN, LSTM)
- ✓ Module 8- Prophet and Neural Prophet