MATH 7339 - Machine Learning and Statistical Learning Theory 2

Section- Automating Forecasting at Scale

1. Over view of time series forecasting models
2. Prophet-an automated model
3. Forecasting Metrics

Times Series Forecasting:

1. **Time domain statistics models:** AR, MA, ARMA, ARIMA, SARIMA

2. **Spectral analysis** (Fourier Transform, Wavelet Transform)

3. **Hybrid automated models** (e.g., Prophet)

4. **Deep Learning:** (RNN, CNN, LSTM, Transformers, etc.)

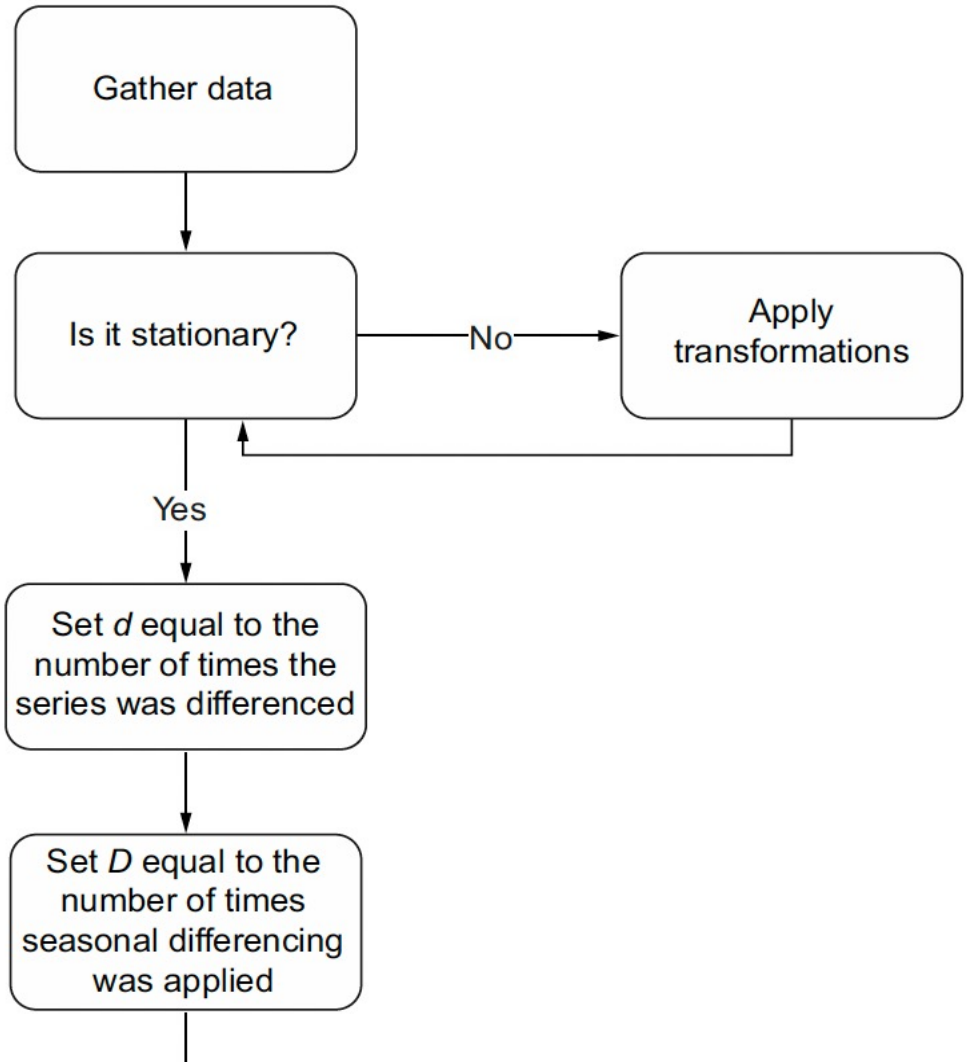
1. Time domain models

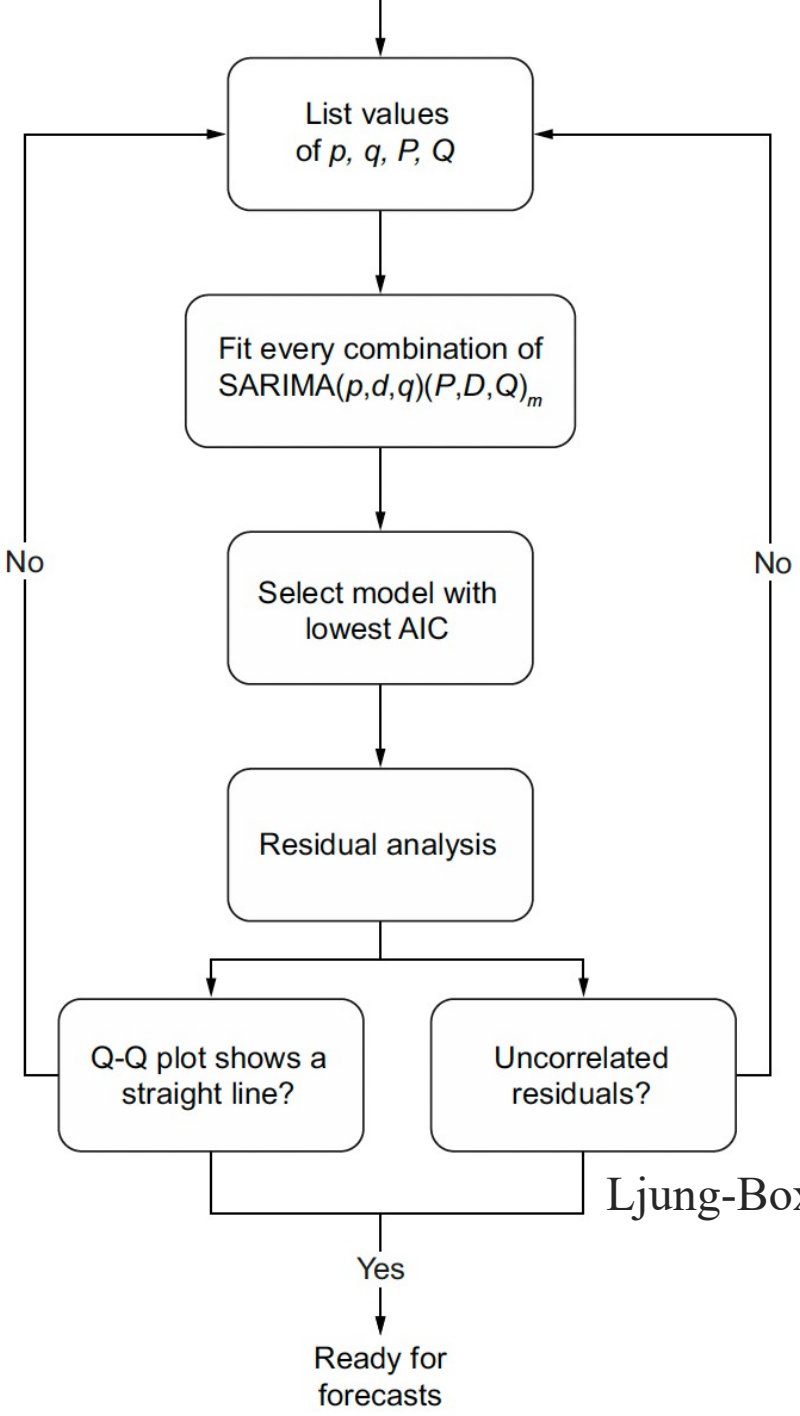
Time domain models like AutoRegressive Integrated Moving Average (ARIMA) process focus on modeling the temporal dependencies and dynamics of the time series itself, without explicitly considering frequency components.

The models give an explicit formula for the current observation in terms of past observations and past white noise terms.

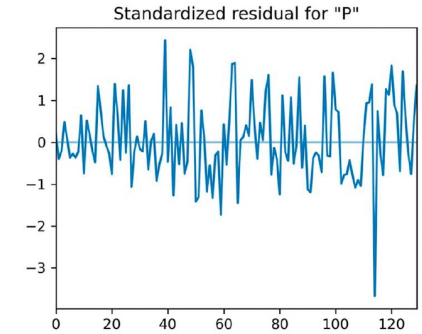
Applying the modeling procedure with a SARIMA model

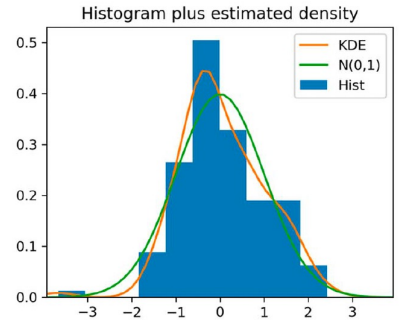
General modeling procedure for the $\mathrm{SARIMA}(p,d,q)(P,D,Q)_s$ model. Note that we can set P, D, and Q to 0 to obtain an ARIMA(p,d,q) model.

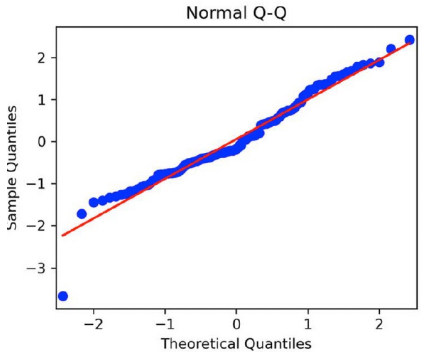


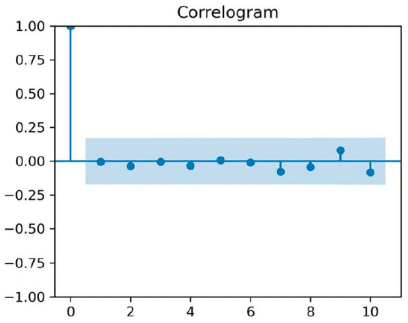


Residual Analysis

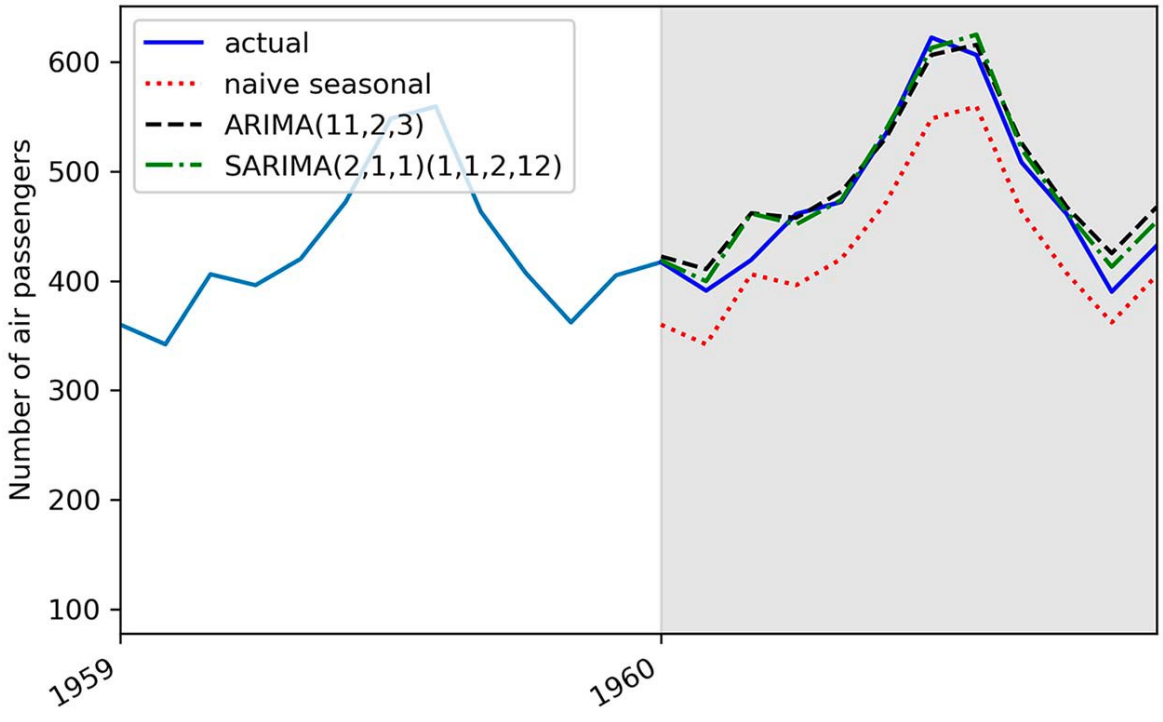


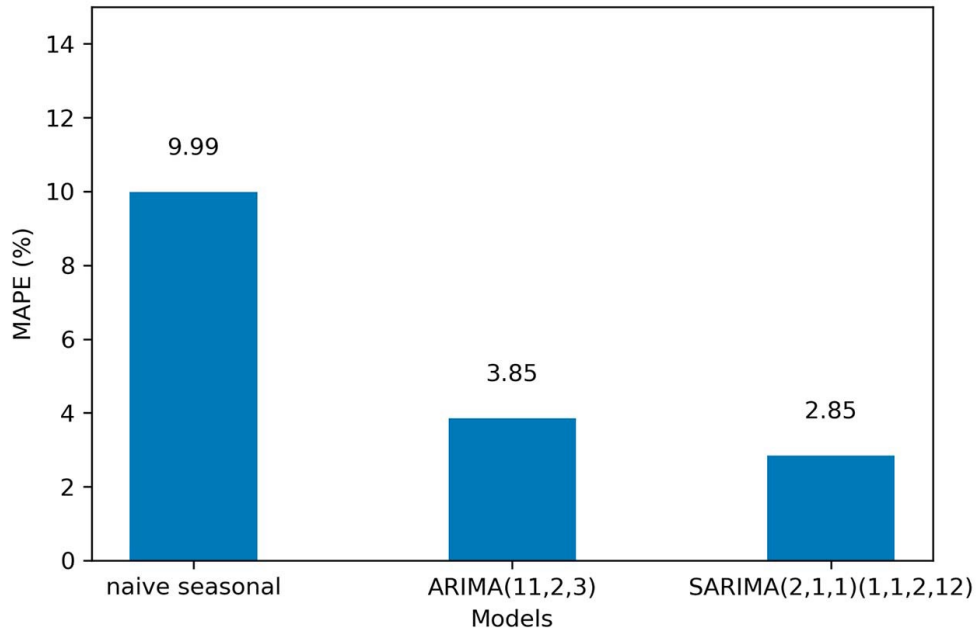






Ljung-Box Test


actual
naive seasonal
ARIMA(11,2,3)
SARIMA(2,1,1)(1,1,2,12)
Number of air passengers
600
500
400
300
200
100
1959
1960
Date

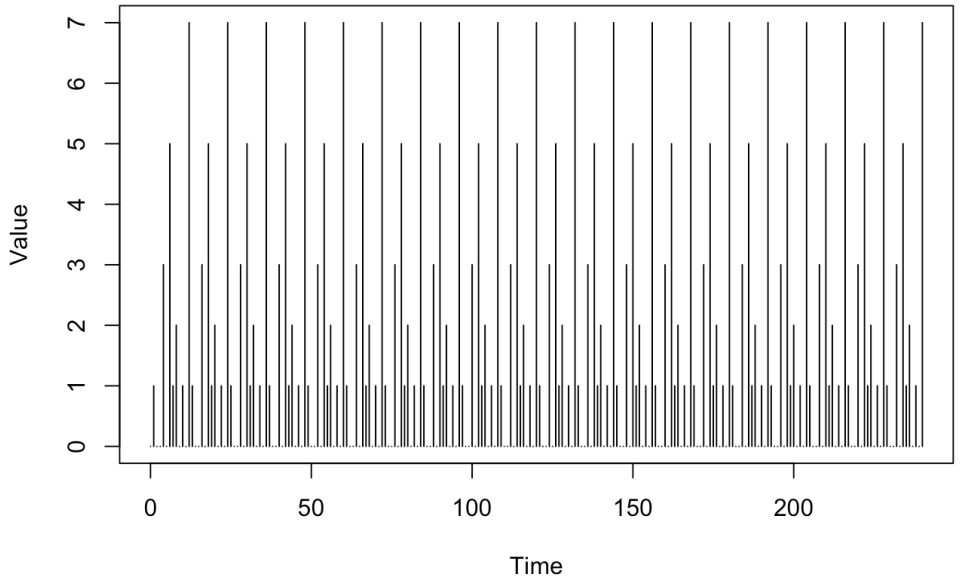

MAPE (%)
14
12
10
8
6
4
2
0
9.99
3.85
2.85
naive seasonal
ARIMA(11,2,3)
SARIMA(2,1,1)(1,1,2,12)
Models

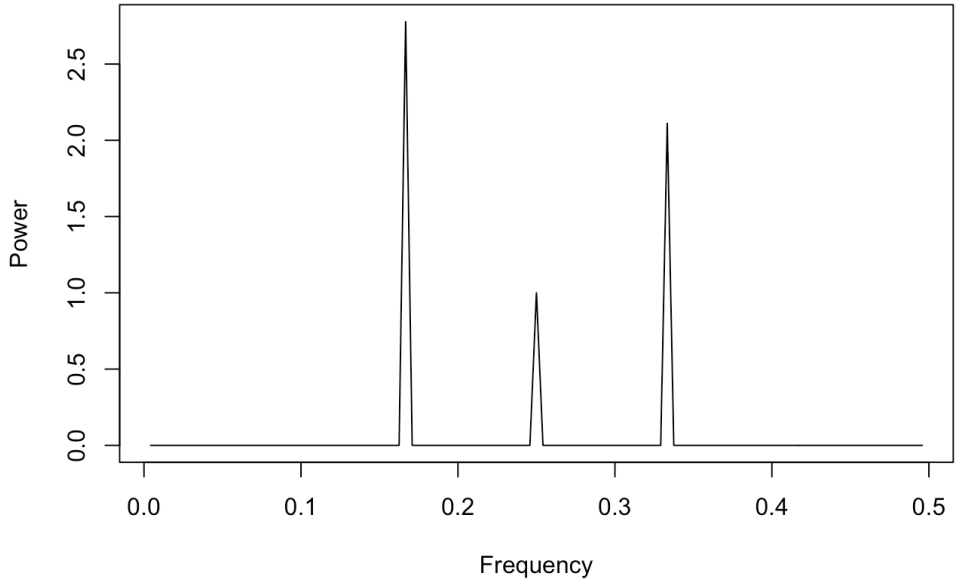
2. Spectral Analysis (frequency domain analysis)

Many time series show periodic behavior. This periodic behavior can be very complex. Spectral analysis is a technique that allows us to discover underlying periodicities:

1. identifying the underlying periodicity of a time series,
2. detecting dominant frequencies or harmonics,
3. detecting changes in frequency over time.

It is particularly useful for analyzing signals that have periodic components, such as audio signals or electromagnetic signals.


Value
0
1
2
3
4
5
6
7
0
50
100
150
200
Time


Power
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.1
0.2
0.3
0.4
0.5
Frequency

3. Automated forecasting libraries

We have so far been building our models by hand. This has given us granular control over what is happening, but it can also be a lengthy process.

The data science community and companies have developed many libraries to automate the forecasting process and make it easier.

Some of the most popular libraries:

- Pmdarima—http://alkaline-ml.com/pmdarima/modules/classes.html
- Prophet—https://facebook.github.io/prophet
- NeuralProphet—https://neuralprophet.com/html/index.html
- PyTorch Forecasting—https://pytorch-forecasting.readthedocs.io/en/stable

❑ Prophet

Prophet is an open source library (for Python and R) for time series forecasting created by Facebook.

It allows people forecast rapidly with minimal manual work. Advanced users, such as ourselves, can fine tune the model to ensure that we get the best results possible.

"Prophet is a procedure for forecasting time series data based on an **additive** model where **non-linear trends** are fit with yearly, weekly, and daily **seasonality**, plus **holiday** effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well."

https://facebook.github.io/prophet/

Prophet Model:

Prophet implements a **general additive model**

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

where each time series $y(t)$ is modeled as the linear combination of

- **trend** g(t), which models the non-periodic long-term changes in the time series,
- **seasonal component** $s(t)$, which models the periodic change,
- **holiday effect** $h(t)$, occurs irregularly and potentially on more than one day
- an **error term** ϵ_t, which is normally distributed and represents any change in value that cannot be explained by the previous three components.

The GAM formulation has the advantage that it decomposes easily and accommodates new components as necessary, for instance when a new source of seasonality is identified. (See Splines section)

Remarks:

Prophet model does not take into account the time dependence of the data, unlike the ARIMA(p, d, q) model, where future values are dependent on past values. Thus, this process is closer to fitting a curve to the data, rather than finding the underlying process.

Although there is some loss of predictive information using this method, it comes with a number of practical advantages:

- **Flexibility**: We can easily accommodate seasonality with multiple periods and let the analyst make different assumptions about trends.
- Unlike with ARIMA models, the measurements do not need to be regularly spaced, and we do not need to interpolate missing values, for example, from removing outliers.
- Fitting is very **fast**, allowing the analyst to interactively explore many model specifications.
- The forecasting model has easily **interpretable** parameters that can be changed by the analyst to impose assumptions on the forecast. Moreover, analysts typically do have experience with regression and are easily able to extend the model to include new components.

1. Trend

Two trend models used in Prophet:

- a saturating growth model,
- a piecewise linear model.

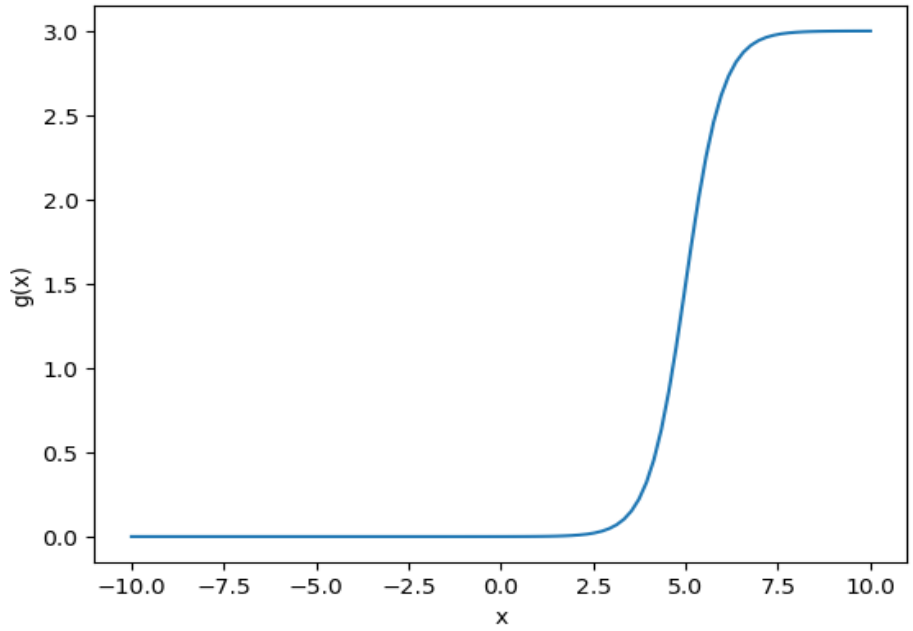
1) Nonlinear, Saturating Growth

This sort of growth is typically modeled using the logistic growth model, which in its most basic form is

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

with C the carrying capacity, k the growth rate, and m an offset parameter.

It is similar to population growth in natural ecosystems (e.g., Hutchinson 1978), where there is nonlinear growth that saturates at a carrying capacity.



$$g(t) = \frac{3}{1 + \exp\big(-2(t-5)\big)}$$

```
def sigmoid(x):
    return 3 / (1 + np.exp(-2*(x-5)))

x = np.linspace(-10, 10, 100)
y = sigmoid(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('f(x)')
plt.show()
```

There are two important modifications for the trend model:

First, the **carrying capacity C** is not constant—as the number of people in the world who have access to the Internet increases, so does the growth ceiling. Replace the fixed capacity C with a time-varying capacity $C(t)$. The set of parameters in $C(t)$ is important. Analysts often have insight into market sizes and can set these accordingly.

Second, the **growth rate k** is not constant. New products can profoundly alter the rate of growth in a region, so the model must be able to incorporate a varying rate to fit historical data.

The piecewise logistic growth model is then

$$g(t) = \frac{C(t)}{1 + \exp\left(-\left(k + \vec{a}(t)^T \vec{\delta}\right)\left(t - \left(m + \vec{a}(t)^T \vec{\gamma}\right)\right)\right)}$$

We incorporate trend changes in the growth model by explicitly defining changepoints where the growth rate is allowed to change. Suppose there are S changepoints at times s_j, $j = 1, \ldots, S$. We define a vector of rate adjustments $\boldsymbol{\delta} \in \mathbb{R}^S$, where δ_j is the change in rate that occurs at time s_j. The rate at any time t is then the base rate k, plus all of the adjustments up to that point: $k + \sum_{j:t>s_j} \delta_j$. This is represented more cleanly by defining a vector $\mathbf{a}(t) \in \{0, 1\}^S$ such that

$$a_j(t) = \begin{cases} 1, & \text{if } t \geq s_j, \\ 0, & \text{otherwise.} \end{cases}$$

The rate at time t is then $k + \mathbf{a}(t)^\intercal \boldsymbol{\delta}$. When the rate k is adjusted, the offset parameter m must also be adjusted to connect the endpoints of the segments. The correct adjustment at changepoint j is easily computed as

$$\gamma_j = \left(s_j - m - \sum_{l<j} \gamma_l \right) \left(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l \leq j} \delta_l} \right).$$

2.) Linear Trend With Changepoints

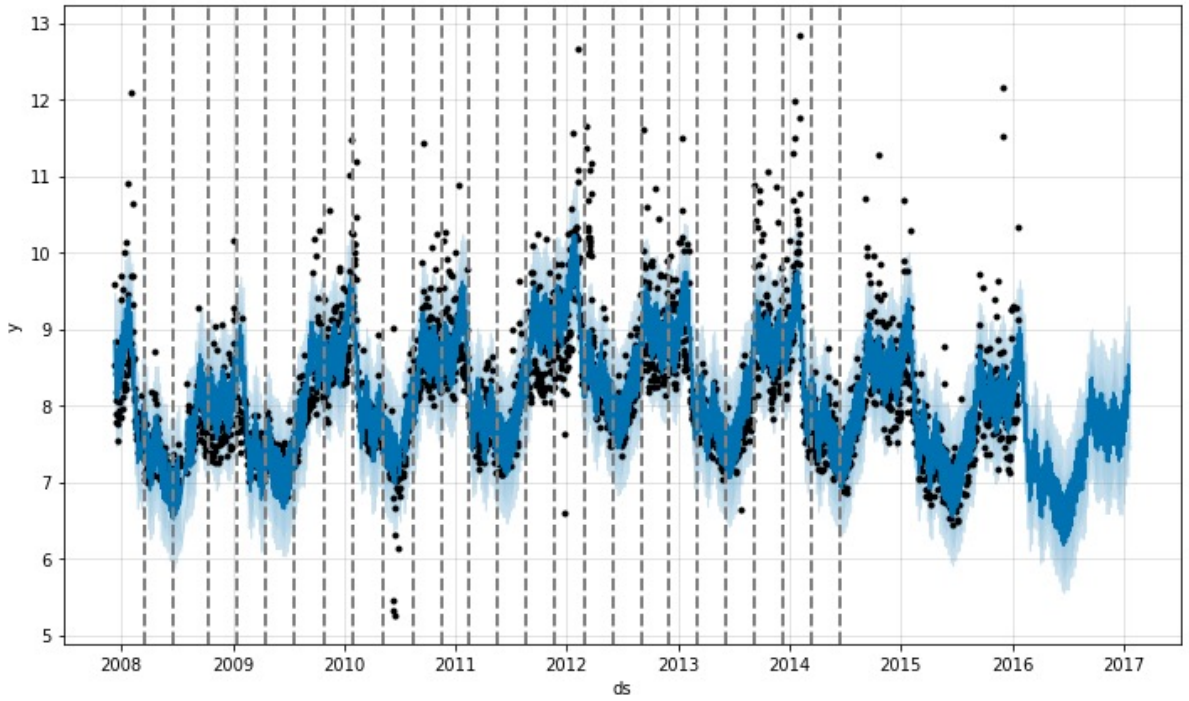
For forecasting problems that do not exhibit saturating growth, a **piece-wise constant rate** of growth provides a parsimonious and often useful model. Here the trend model is

$$g(t) = \left(k + \vec{a}(t)^T \vec{\delta}\right)t + (m + \vec{a}(t)^T \vec{\gamma})$$

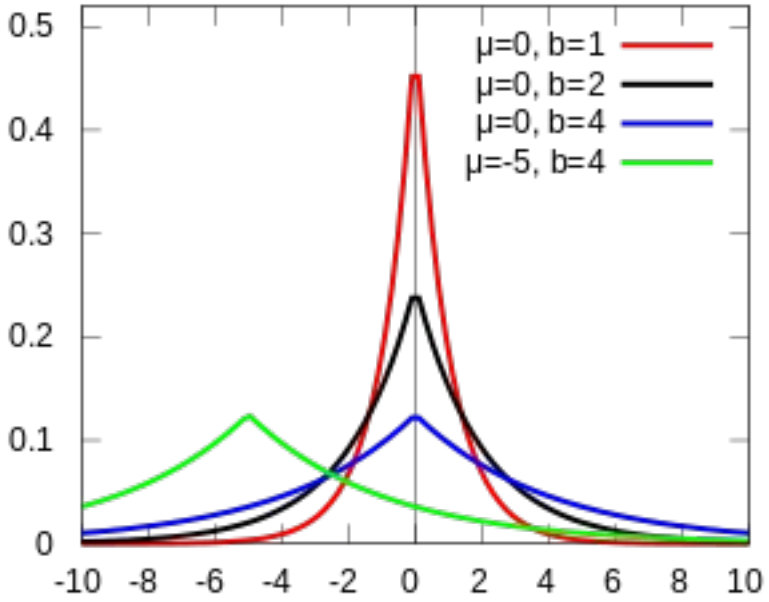
where k is the growth rate, δ has the rate adjustments, m is the offset parameter, and γ_j is set to $-s_j \delta_j$ to make the function continuous.

Automatic Changepoint Selection

The changepoints s_j could be specified by the analyst using known dates of product launches and other growth-altering events, or may be automatically selected given a set of candidates.



Specify a large number of changepoints (e.g., one per month for a several year history) and use the prior $\delta_j \sim Laplace(0, \tau)$. The parameter τ directly controls the flexibility of the model in altering its rate.



$Laplace(\mu, b)$.

2. **Seasonality**

Prophet uses the Fourier series to model multiple periodic effects.

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

where P is the length of the seasonal period in days, and N is the number of terms in the Fourier series.

If we have a yearly seasonality, P = 365.25, as there are 365.25 days in a year. For a weekly seasonality, P = 7.

Truncating the series at N applies a low-pass filter to the seasonality, so increasing N allows for fitting seasonal patterns that change more quickly, albeit with increased risk of overfitting.

By default, Prophet uses 10 terms to model the yearly seasonality and 3 terms to model the weekly seasonality. The choice of these parameters could be automated using a model selection procedure such as AIC.

3. Holiday

Holidays are irregular events that can have a clear impact on a time series. For example, events such as Black Friday in the United States can dramatically increase the attendance in stores or the sales on an ecommerce website. Similarly, Valentine's Day is probably a strong indicator of an increase in sales of chocolates and flowers.

For each holiday i, let D_i be the set of past and future dates for that holiday. We add an indicator function representing whether time t is during holiday i, and assign each holiday a parameter κ_i which is the corresponding change in the forecast.

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

and taking

$$h(t) = Z(t)\vec{\kappa}$$

As with seasonality, we use a prior $\kappa \sim Normal(0, \nu^2)$.

- **Model Fitting**

  Maximum a posteriori estimate

- **Analyst-in-the-Loop Modeling**

  Analysts can alter the model to apply their expertise and external knowledge with

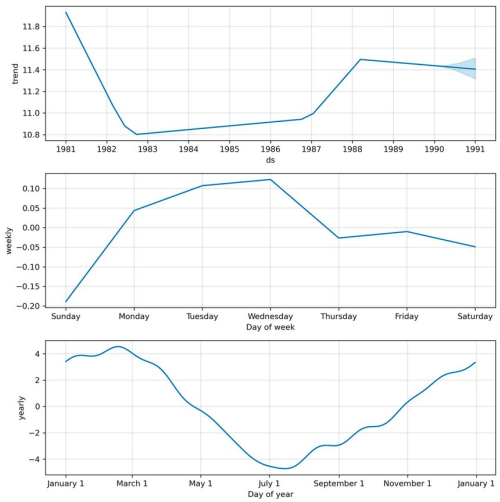
  1. Capacities
  2. Changepoints
  3. Holidays and seasonality
  4. Smoothing parameters

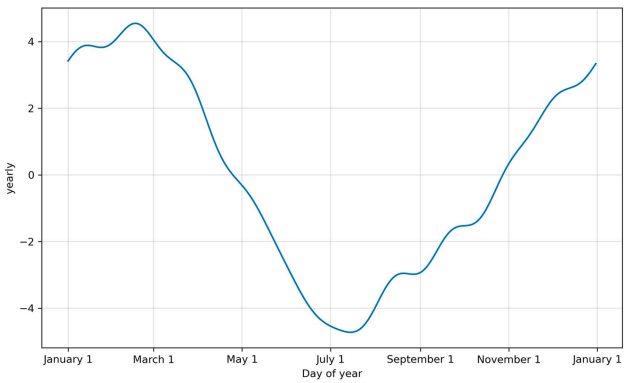
- **Automating Evaluation of Forecasts**

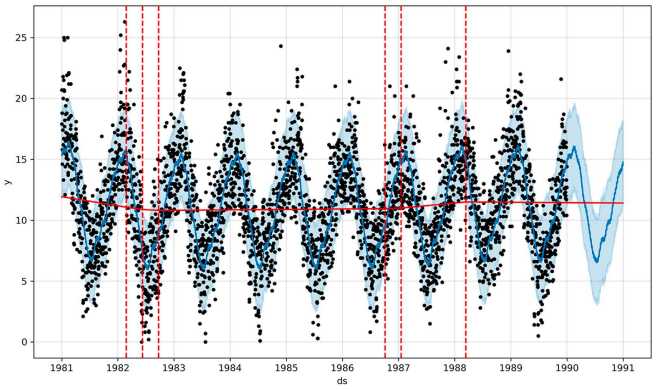
❑ Prophet's advanced functionality

1. Visualization capabilities

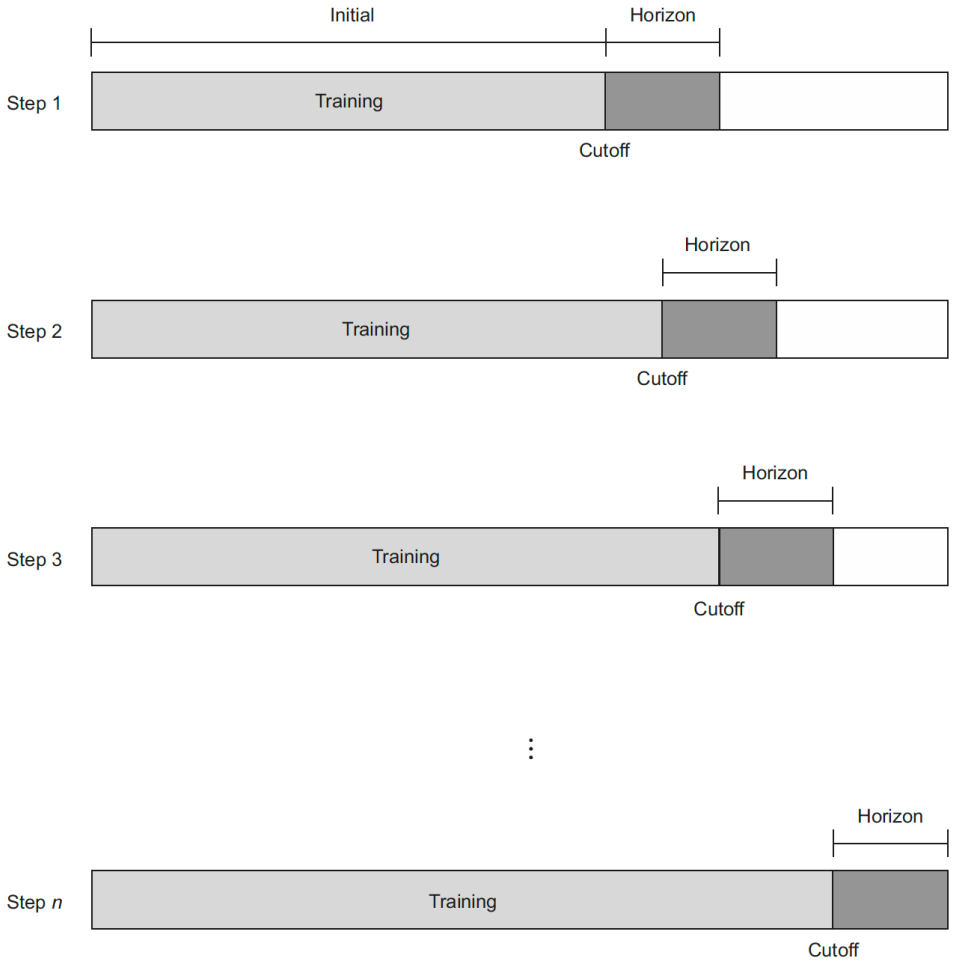
```
from fbprophet.plot import plot_yearly, plot_weekly
from fbprophet.plot import add_changepoints_to_plot
fig1 = m.plot(forecast)
fig2 = m.plot_components(forecast)
fig3 = plot_yearly(m)
fig4 = m.plot(forecast)
a = add_changepoints_to_plot(fig4.gca(), m, forecast)
```







2. Cross-validation




```
from fbprophet.diagnostics import cross_validation
df_cv = cross_validation(m, initial='730 days',
period='180 days', horizon='365 days')
```

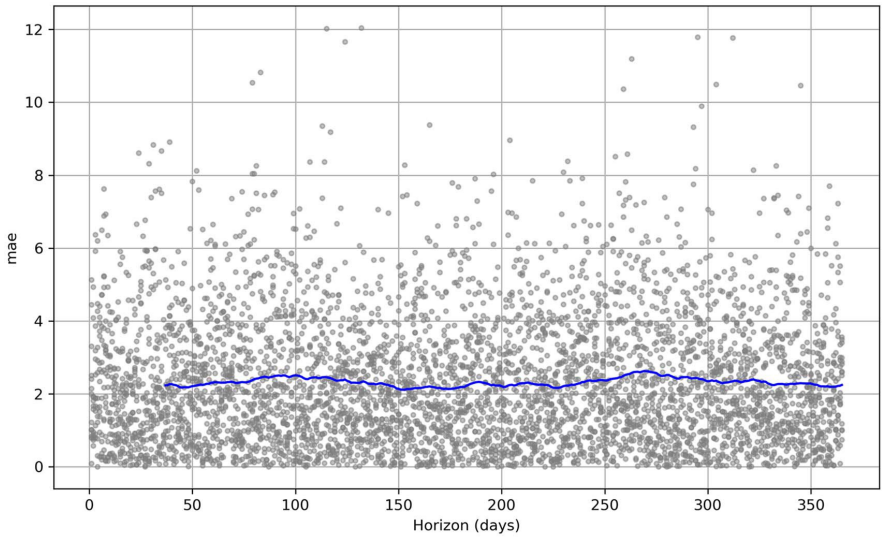
The **initial training set** has 2 years of data. Each **cutoff** date (period) is separated by 180 days, or half a year. The forecast **horizon** is 365 days, which is a year.

3. Performance metrics

```
from fbprophet.diagnostics import performance_metrics
df_perf = performance_metrics(df_cv, rolling_window=0)
df_perf.head()
```

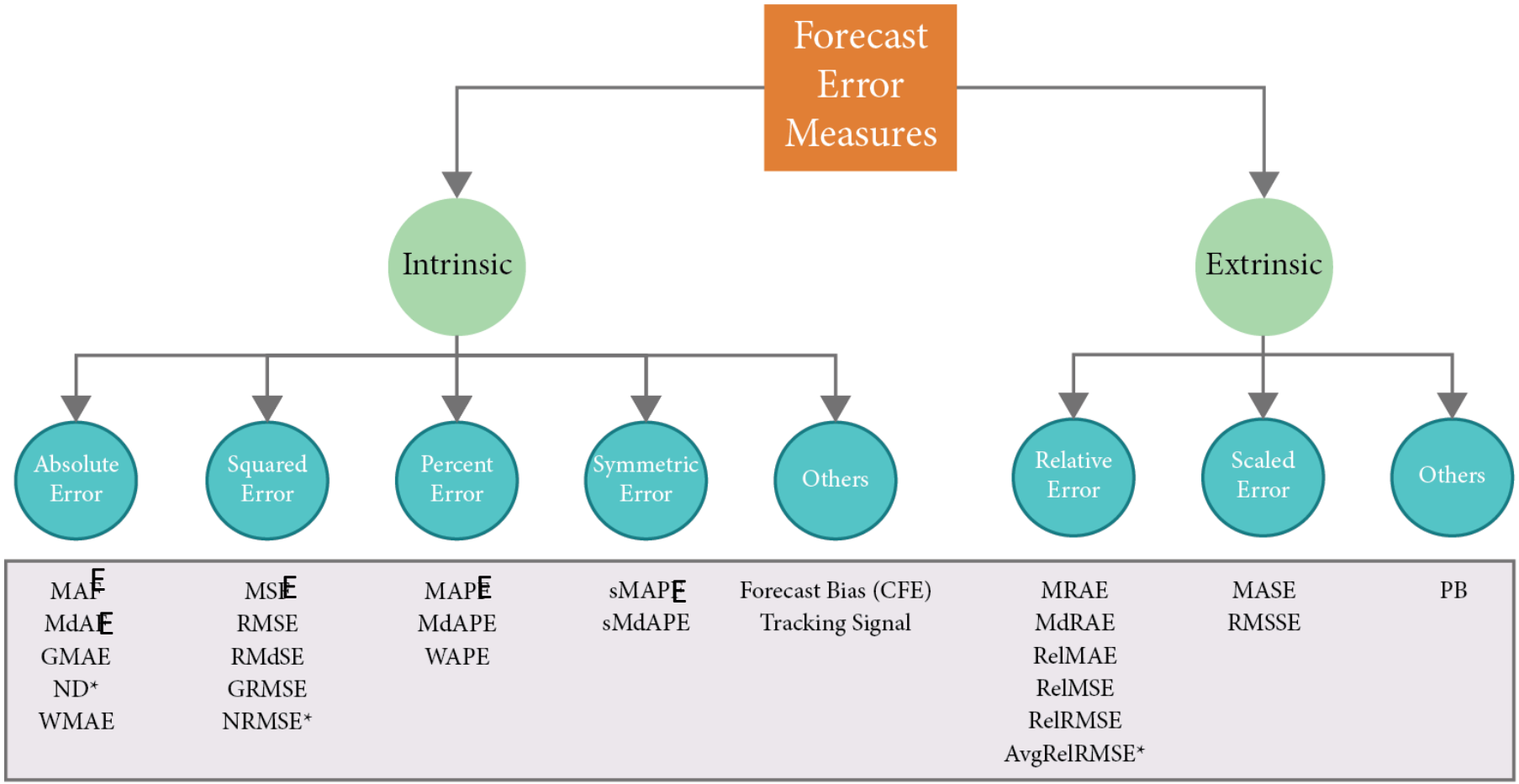
	horizon	mse	rmse	mae	mdape	coverage
0	1 days	6.350924	2.520104	2.070329	0.147237	0.846154
1	2 days	4.685452	2.164590	1.745606	1.139852	0.846154
2	3 days	10.049956	3.170167	2.661797	0.147149	0.769231
3	4 days	8.686183	2.947233	2.377724	0.195119	0.769231
4	5 days	8.250061	2.872292	2.569552	0.196067	0.692308

```
from fbprophet.plot import plot_cross_validation_metric
fig7 = plot_cross_validation_metric(df_cv, metric='mae')
```



Taxonomy of forecast error measures

There are more metrics in time series forecasting. We can semantically separate the different forecast metrics into two buckets – intrinsic and extrinsic.



* - Strictly Aggregate Measures

Absolute error

- **Mean Absolute Error (MAE)**: $\text{MAE} = \frac{1}{H}\sum_{t=1}^{H}|e_t|$
- **Median Absolute Error**: $MdAE = median(|e_t|)$
- **Geometric Mean Absolute Error**: $GMAE = \sqrt[H]{\prod_{t=1}^{H}|e_t|}$
- **Weighted Mean Absolute Error**: This is a more esoteric method that lets you put more weight on a particular timestep in the horizon:

$$WMAE = \frac{\sum_{t=1}^{H} w_t|e_t|}{\sum_{t=1}^{H} w_t}$$

Here, w_t is the weight of a particular timestep. This can be used to assign more weight to special days (such as weekends or promotion days).

Squared error

- **Mean Squared Error**: $\text{MSE} = \frac{1}{H}\sum_{t=1}^{H}(e_t^2)$
- **Root Mean Squared Error (RMSE)**: $\text{RMSE} = \sqrt{\frac{1}{H}\sum_{t=1}^{H}(e_t^2)}$
- **Root Median Squared Error**: $\text{RMdSE} = median\left(\sqrt{e_t^2}\right)$
- **Geometric Root Mean Squared Error**: $GRMSE = \sqrt[2n]{\prod_{t=1}^{H}(e_t^2)}$

Percent error

- **Mean Absolute Percent Error (MAPE)** – $\text{MAPE} = \sum_{t=1}^{H} \frac{100|e_t|}{y_t}$
- **Median Absolute Percent Error** – $\text{MdAPE} = \text{median}\left(\frac{100|e_t|}{y_t}\right)$

Extrinsic metrics

There are two major buckets of metrics under the extrinsic umbrella – relative error and scaled error.

One of the problems of intrinsic metrics is that they don't mean a lot unless a benchmark score exists. For instance, if we hear that the MAPE is 5%, it doesn't mean a lot because we don't know how forecastable that time series is. Maybe 5% is a bad error rate. Relative error solves this by including a benchmark forecast in the calculation so that the errors of the forecast we are measuring are measured against the benchmark

4. Hyperparameter tuning

Prophet comes with many parameters that can be fine-tuned by more advanced users in order to produce better forecasts. Four parameters are usually tuned:

- `changepoint_prior_scale`,
- `seasonality_prior_scale`,
- `holidays_prior_scale`,
- `seasonality_mode`.

Forecasting process using Prophet

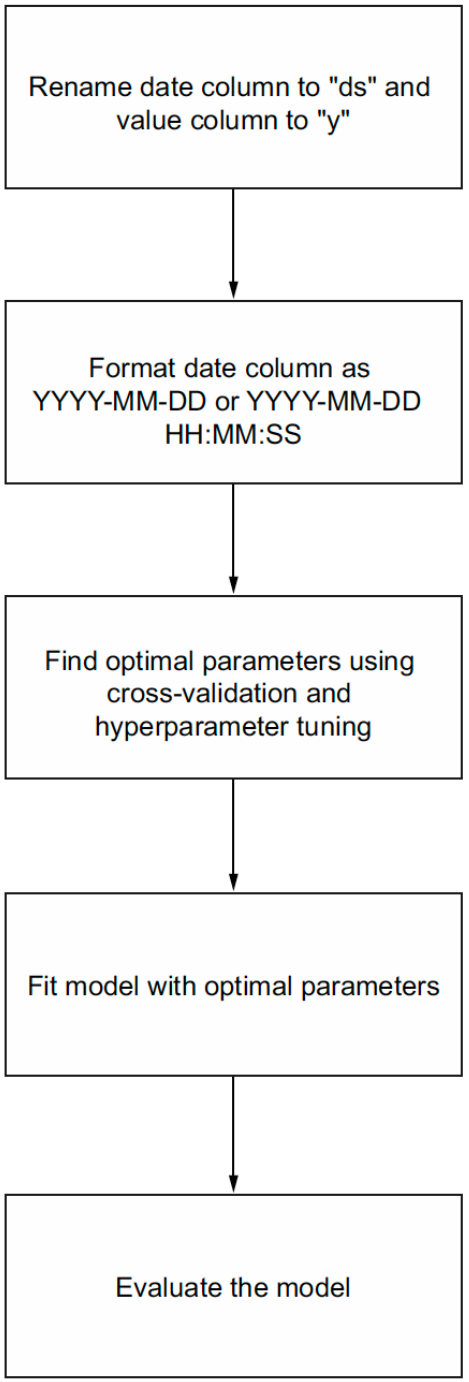
First, we'll ensure that the dataset has the right column names for Prophet

The date is expressed correctly as a datestamp or a timestamp.

Then, we'll combine hyperparameter tuning with cross-validation to obtain the optimal parameters for our model.

Finally fit the model using the optimal parameters

Evaluate it on a test set.



References:

Facebook Prophet:

https://facebook.github.io/prophet/

Forecasting at scale

https://peerj.com/preprints/3190/

Textbook: Time Series Forecasting in Python:
https://github.com/marcopeix/TimeSeriesForecastingInPython

Modern Time Series Forecasting with Python
https://github.com/PacktPublishing/Modern-Time-Series-Forecasting-with-Python

Is Facebook PROPHET Superior than Hybrid ARIMA Model to Forecast Crude Oil Price?

http://www.ukm.my/jsm/pdf_files/SM-PDF-51-8-2022/22.pdf