Prophet by Facebook

Chris Hunter, Valery Satsevich

What is Prophet by Facebook?

Prophet is a forecasting procedure implemented in R and Python. It is an open source software released by Facebook's Core Data Science team. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is available in Python and R.

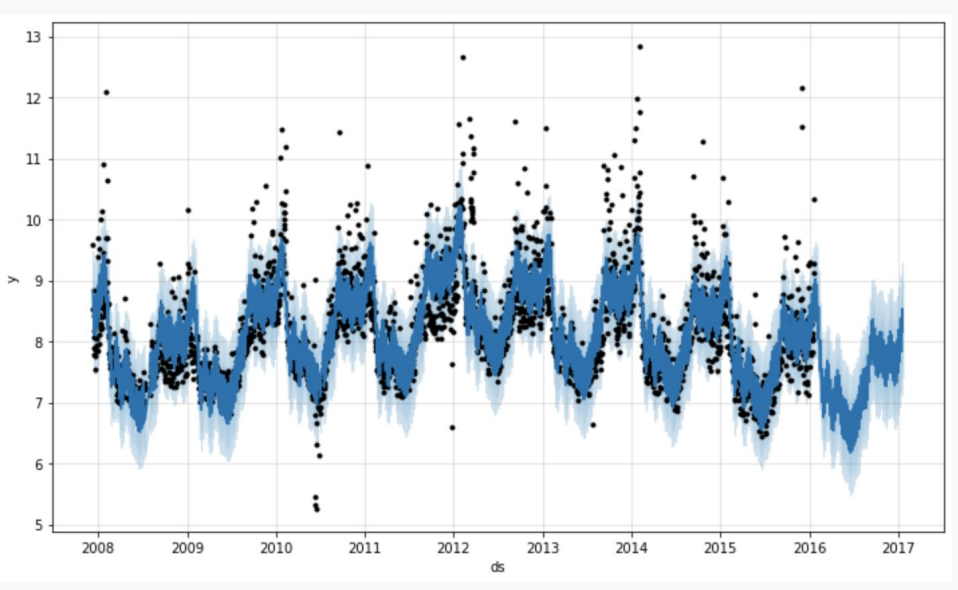
Reasons to use Prophet by Facebook to predict time series data

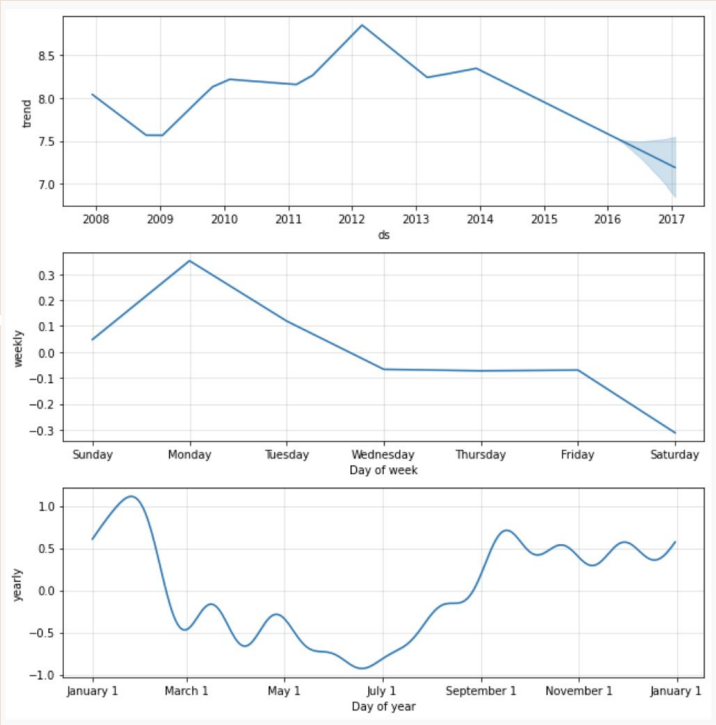
Accurate and fast.
Prophet is used in many applications across Facebook for producing reliable forecasts for planning and goal setting.

Fully automatic.
Get a reasonable forecast on messy data with no manual effort. Prophet is robust to outliers, missing data, and dramatic changes in your time series.

Tunable forecasts.
The Prophet procedure includes many possibilities for users to tweak and adjust forecasts. You can use human-interpretable parameters to improve your forecast by adding your domain knowledge.

Visualisation available with Prophet





Installation in Python

Using Anaconda:

```
conda install -c conda-forge prophet
```

Using pip:

```
python -m pip install prophet
```

Prophet can also be installed in R

Starting

Prophet follows the sklearn model API. We create an instance of the Prophet class and then call its fit and predict methods.

The input to Prophet is always a dataframe with two columns: ds and y. The ds (datestamp) column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The y column must be numeric, and represents the measurement we wish to forecast.

Basic Methods

- **fit(self, df, **kwargs)**
  - Like sklearn methods you're able to fit a model to an input dataframe
- **make_future_dataframe(self, periods, freq='D', include_history=True)**
  - periods: indicate the amount of future iterations you want to generate
  - Freq: the type of iterations you want to generate "H" for hourly, "D" for daily, "M" for monthly, "Y" for yearly
- **predict(self, df=None)**
  - Use the future dataframe as an input to be predicted by the model

Holidays/Seasonality

- Time series data can be influenced by the presence of holidays, which could cause unusual dips or spikes in your data
  - Example: increased amounts of people going out to dinner on Valentine's day
- Seasonality is also something that must be taken into consideration because it's important to take note of patterns that repeat at constant intervals for making future predictions
  - Example: if an ice cream company's sales start trending down in the winter, you can look at past data saying they always trend back upwards in the summer months to help develop a more accurate prediction
- Prophet accounts for both of these!
  - By adding holidays and seasonality considerations to your model, prophet essentially creates a new predictive variable for each to account for the differences that would be present in the forecast

Holiday Methods

- **add_country_holidays(country_name)**
  - Country names can be the actual name or in some cases abbreviation. Ex: ‘US’
- Holidays can also be expressed as a dataframe to be added into the Prophet constructor manually
- **m = Prophet(df, holidays = holidays)**
  - Pass holidays as an argument for the model

If you have holidays or other recurring events that you’d like to model, you must create a dataframe for them. It has two columns (holiday and ds) and a row for each occurrence of the holiday. Once the table is created, holiday effects are included in the forecast by passing them in with the holidays argument.

Seasonality Methods

```
def is_nfl_season(ds):
    date = pd.to_datetime(ds)
    return (date.month > 8 or date.month < 2)

df['on_season'] = df['ds'].apply(is_nfl_season)
df['off_season'] = ~df['ds'].apply(is_nfl_season)
```

- **add_seasonality(name, period, fourier_order, condition_name)**
  - Fourier_order determines how many fourier components are included to model seasonality. More can capture more complex patterns, but may cause overfitting
  - Condition_name allows you to condition the seasonality on the value of another variable (ex. temperature)
- **model = Prophet(weekly_seasonality=True)**
  - Can also be added to the constructor

In some instances the seasonality may depend on other factors, such as a weekly seasonal pattern that is different during the summer than it is during the rest of the year, or a daily seasonal pattern that is different on weekends vs. on weekdays. These types of seasonalities can be modeled using conditional seasonalities.

Additional Regressors

- **add_regressor(name, prior_scale = none, mode = none)**
- Name variable holds value of the new regressor
- Prior_scale determines how strong the regularization on the coefficients of the new regressors is
- Mode determines whether the effect of the new regressor is added or multiplied to the forecast
    - Ex. ‘additive’ or ‘multiplicative’

- Sometimes additional variables need to be accounted for in our forecast that aren’t inherently defined as a holiday or season.
- Market Campaigns
- Weather Patterns

Demo

Imagine you have a business that has relatively high sales on weekdays, but on the weekends experiences a significant dip. How can you forecast this trend using Prophet?

Sources

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

https://www.geeksforgeeks.org/time-series-analysis-using-facebook-prophet/

https://medium.com/illumination/understanding-fb-prophet-a-time-series-forecasting-algorithm-c998bc52ca10#:~:text=The%20core%20idea%20behind%20FBProphet,underlying%20patterns%20in%20the%20data.