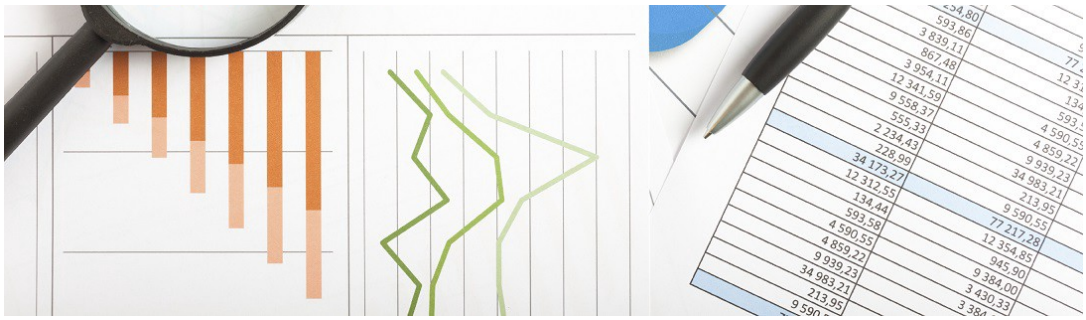
towardsdatascience.com

# A Quick Start of Time Series Forecasting with a Practical Example using FB Prophet

*Yang Lyla*

11-14 minutos



1. Introduction

- Time Series Analysis
- Why Facebook Prophet?

2. The Prophet Forecasting Model

- Saturating growth
- Trend Change points
- Seasonality, Holiday Effects, And Regressors

3. Case study: forecasting advertising spend with Prophet

4. Closing Summary

## 1.1 Time Series Analysis

Time series analysis is an approach to analyze time series data to extract meaningful characteristics of data and generate other useful insights applied in business situation. Generally, time-series data is a sequence of observations stored in time order. Time-series data often stands out when tracking business metrics, monitoring industrial processes and etc.

Time series analysis helps understand time based patterns of a set of metric data points which is critical for any business. Techniques of time series forecasting could answer business questions like how much inventory to

maintain, how much website traffic do you expect in your e-store to how many product will be sold in the next month — all of these are important time series problems to solve. The basic objective of time series analysis usually is to determine a model that describes the pattern of the time series and could be used for forecasting.

Classical time series forecasting techniques build on stats models which requires lots of effort to tune models and expect in data and industry. The person has to tune the parameters of the method with regards to the specific problem when a forecasting model doesn't perform as expected. Tuning these methods requires a thorough understanding of how the underlying time series models work. It's difficult for some organizations to handling those forecasting without data science teams. And it might seem doesn't profitable for an organization to have a bunch of expects on board if there is no need a build a complex forecasting platform or other services.

### 1.2 Why Facebook Prophet?

Facebook developed an open sourcing Prophet, a forecasting tool available in both Python and R. It provides intuitive parameters which are easy to tune. Even someone who lacks deep expertise in time-series forecasting models can use this to generate meaningful predictions for a variety of problems in business scenarios.

From Facebook Prophet website:

" Producing high quality forecasts is not an easy problem for either machines or for most analysts. We have observed two main themes in the practice of creating a variety of business

forecasts:

- Completely automatic forecasting techniques can be brittle and they are often too inflexible to incorporate useful assumptions or heuristics.
- Analysts who can product high quality forecasts are quite rare because forecasting is a specialized data science skill requiring substantial experience. ”

1.3 Highlights of Facebook Prophet

- Very fast, since it’s built in Stan, a programming language for statistical inference written in C++.
- An additive regression model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects: 1. A piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data 2. A yearly seasonal component modeled using Fourier series 3. A weekly seasonal component using dummy variables 4. A user-provided list of important holidays.
- Robust to missing data and shifts in the trend, and typically handles outliers .
- Easy procedure to tweak and adjust forecast while adding domain knowledge or business insights.

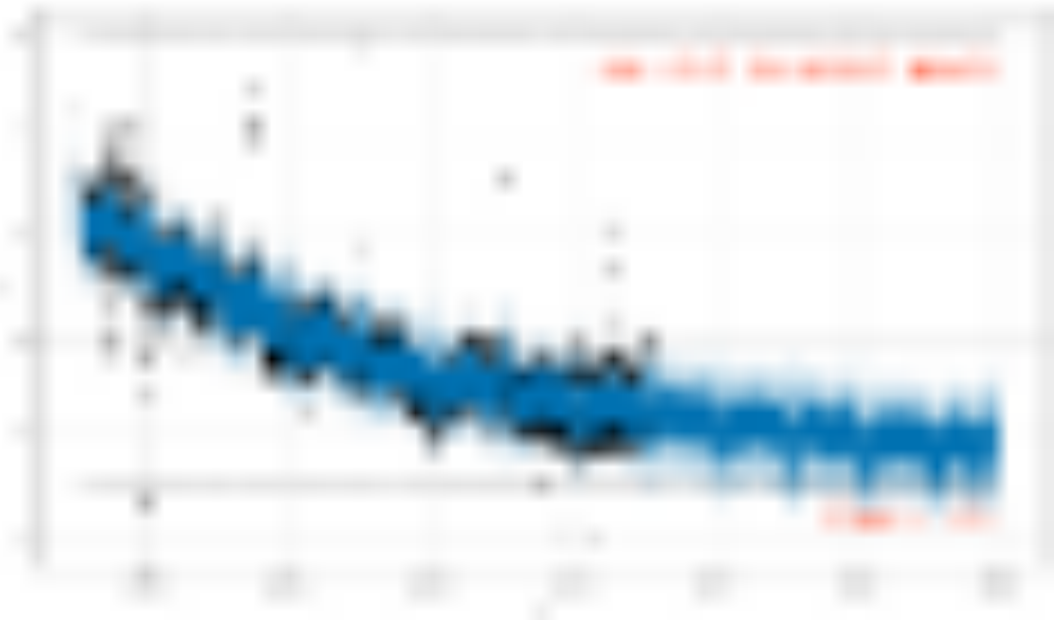
## 2.1 The Prophet Forecasting Model

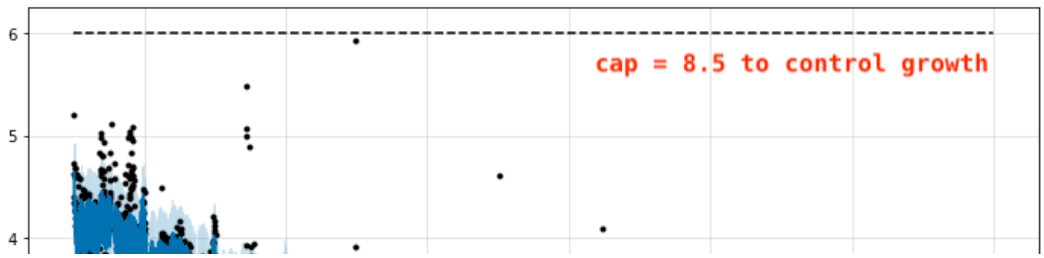
The Prophet uses a decomposable time series model with three main model components: trend, seasonality, and holidays. They are combined in the following equation:

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

- g(t): piecewise linear or logistic growth curve for modeling non-periodic changes in time series
- s(t): periodic changes (e.g. weekly/yearly seasonality)
- h(t): effects of holidays (user provided) with irregular schedules
- εt: error term accounts for any unusual changes not accommodated by the model
- Using time as a regressor, Prophet is trying to fit several linear and non linear functions of time as components. Modeling seasonality as an additive component is the same approach taken by exponential smoothing in Holt-Winters technique . Prophet is framing the forecasting problem as a curve-fitting exercise rather than looking explicitly at the time based dependence of each observation within a time series.

### 2.2 Saturating growth



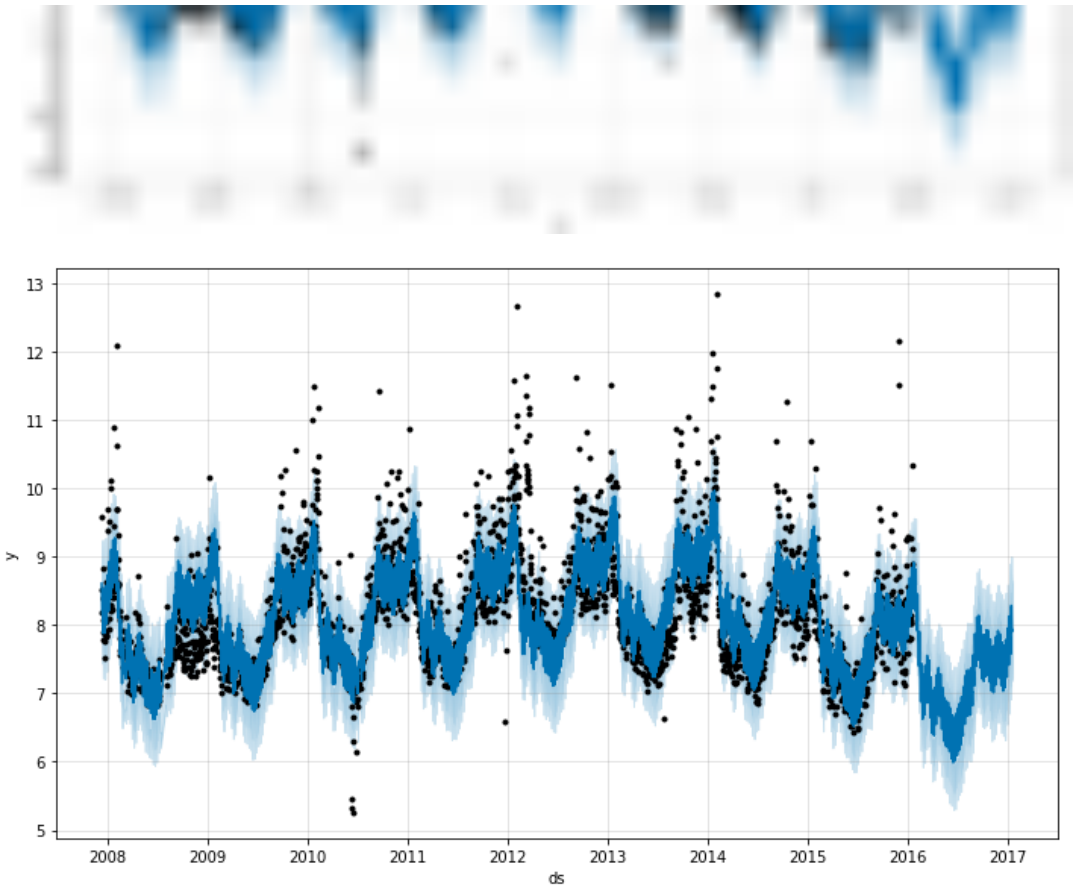



- Set a carrying capacity `cap`to specify the maximum achievable point due to the business scenarios or constraints: market size, total population size, maximum budget, etc.
- A saturating minimum, which is specified with a column `floor` in the same way as the `cap` column specifies the maximum.

  2.3 Trend Changepoints

  The model could be overfitting or underfitting while working with the trend component. The input of changepoints built in Prophet allowed is increased the fit becomes more flexible.

  Here, you can nicely apply your business insights: big jump of sales during holidays, cost decreasing in future by purpose and etc. A user can also manually feed the changepoints with those business insights if it is required. In the below plot, the dotted lines represent the changepoints for the given time series.




## 2.4 Seasonality, Holiday Effects, And Regressors

Seasonal effects s(t) are approximated by the following function:

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

P is the period (365.25 for yearly data and 7 for weekly data)

Parameters [$a_1$, $b_1$, ....., $a_N$, $b_N$] need to be estimated for a given N to model seasonality.

Prophet has a built-in holiday feature which allows inputs of customized recurring events.

Finally, action time!

## 3. Case study: forecasting advertising spend with Prophet in Python

I took the sample data of advertising spend from a digital marketing platform. I also did some changes on purpose to make it a 'fake' data source in order to use in this case study.

Here, we try to use last 17 month data to predict the next 30 days ad spend.

Step 1: Import libraries and data set:

[Code]:

```
import pandas as pd
pd.set_option('display.max_columns', None)

import numpy as np
from fbprophet import Prophet

%matplotlib inline
import matplotlib.pyplot as plt

sample=pd.read_csv('/.../ad_spend.csv')
```

Step 2: Check data info

[Code]:

```
print(sample.shape[0], 'rows','AND', sample.shape[1],'columns')
print('Min Date:',sample.Date.min(),'AND','Max Date:',sample.Date.max())
# Print 5 rows at the beginning
sample.head()
```

```
577 rows AND 2 columns
Min Date: 2017-06-01 00:00:00 AND Max Date: 2018-12-30 00:00:00
```

| | Date | Spend |
|---|---|---|
| 116 | 2017-06-01 | $289,890 |
| 117 | 2017-06-02 | $197,763 |
| 118 | 2017-06-03 | $111,162 |
| 119 | 2017-06-04 | $127,185 |
| 120 | 2017-06-05 | $260,577 |

From the above, the data set contains one and half year daily advertising spend from 2017–06–01 to 2018–12–30. There are 577 rows and two columns( date and spend) in the data frame.

Let's check the missing value:

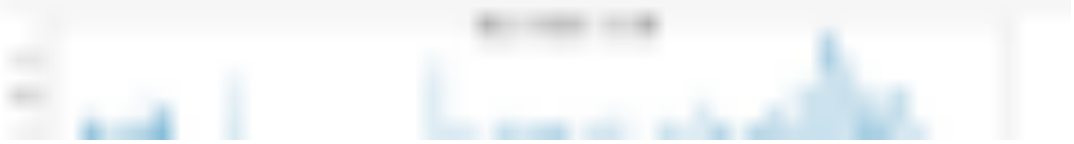
there is no missing value (from the able below) which is great!👏

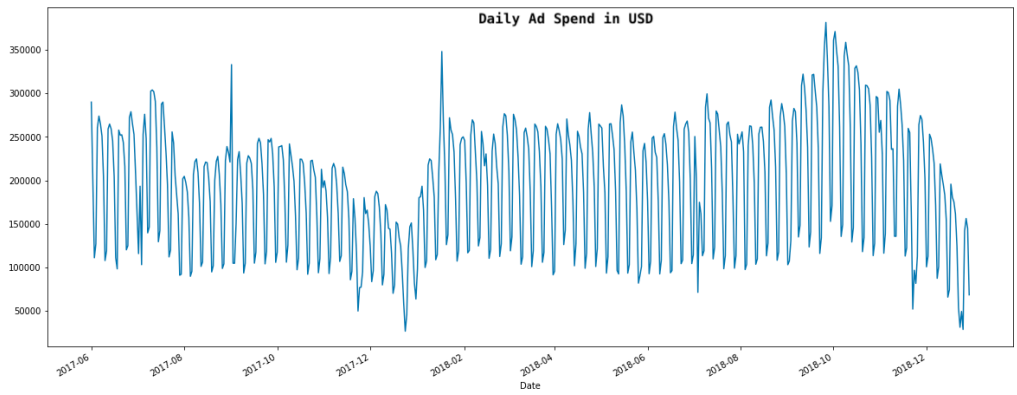
[Code]:

```
sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 577 entries, 116 to 692
Data columns (total 2 columns):
Date     577 non-null datetime64[ns]
Spend    577 non-null object
dtypes: datetime64[ns](1), object(1)
memory usage: 13.5+ KB
```

Step 3: Plot time-series data






Y-Axis: Ad Spend; X-Axis: Date

It can be seen from the plot that there is roughly constant level (the mean of daily spend: 200K USD). The seasonal fluctuation and random fluctuations roughly are constant in size over time. This suggests that it's probably appropriate to describe the data using an additive model which is Prophet built on.

Step 4: Modeling

Split data set into training set and test set. The training set contains daily ad spend from 2017–06–01 to 2018–11–30 while the test set contains daily ad spend from 2018–12–01 to 2018–12–30. Here we would like to use training data set to predict next 30 days ad spend.

Let's try first model by itself without giving any parameters.

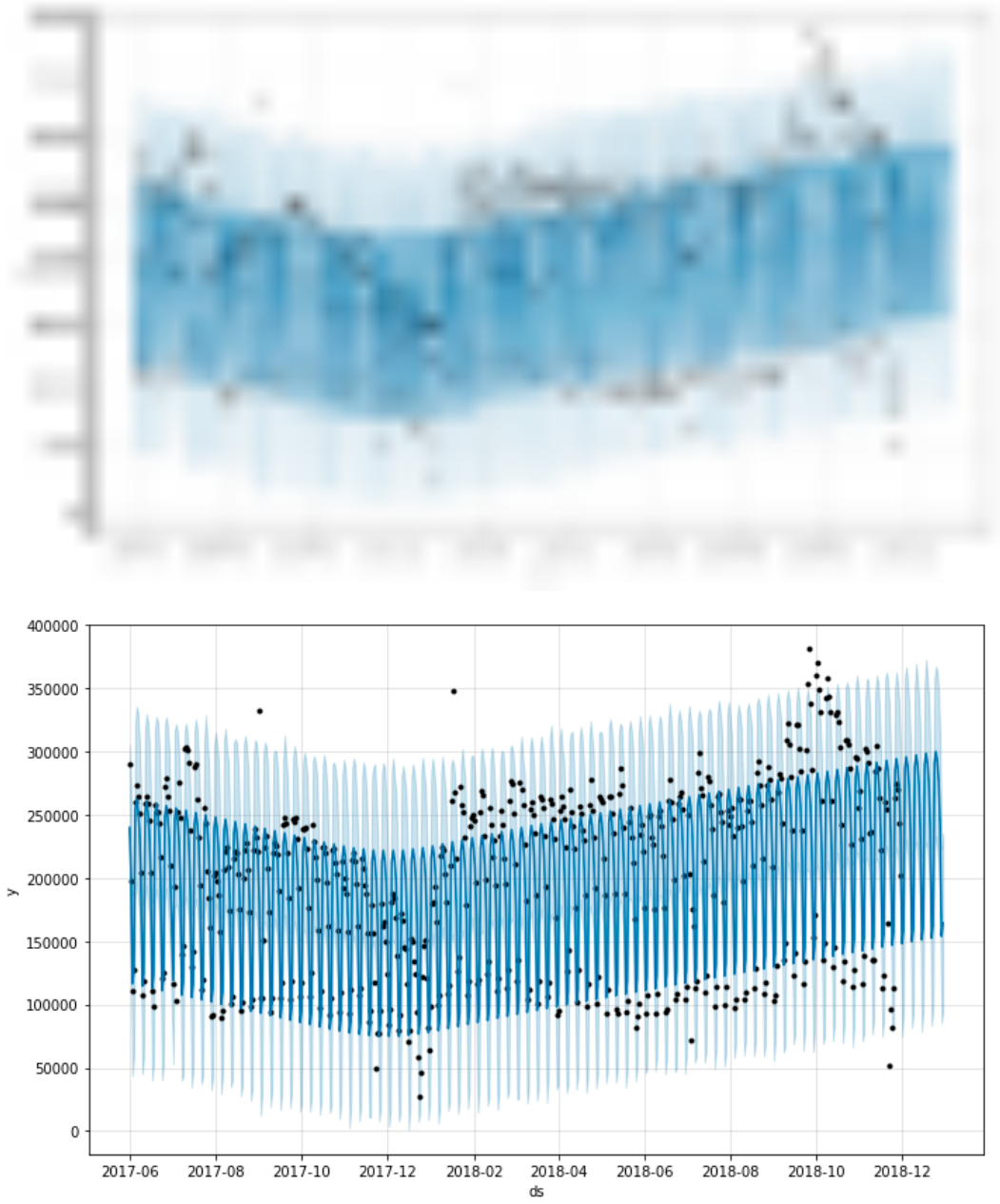
[Code]:

*model1=Prophet(interval_width=0.95)* # by default is 80%

*'interval_width=0.95'*, this sets the uncertainty interval to

produce a confidence interval around the forecast.

Generate the forecasting plot below:




Y-Axis: Ad Spend; X-Axis: Date.

It's always nice to check how does the model perform on historical data. (Deep blue line is forecasting spend numbers, black dots are actually spend numbers. The light blue shade is 95% confidence interval around the forecast.) From the plot, thought the model tries to fit all data point smoothly but it fails to catch the seasonality. The first model

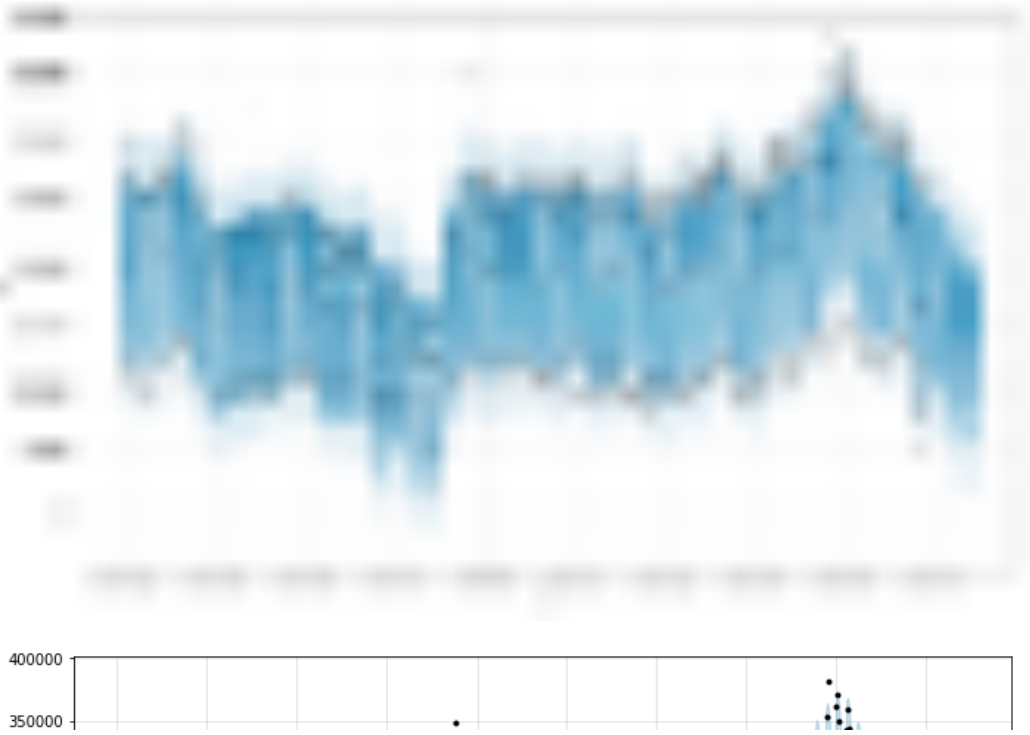
is not doing a good job on fitting the data just by applying Prophet itself.

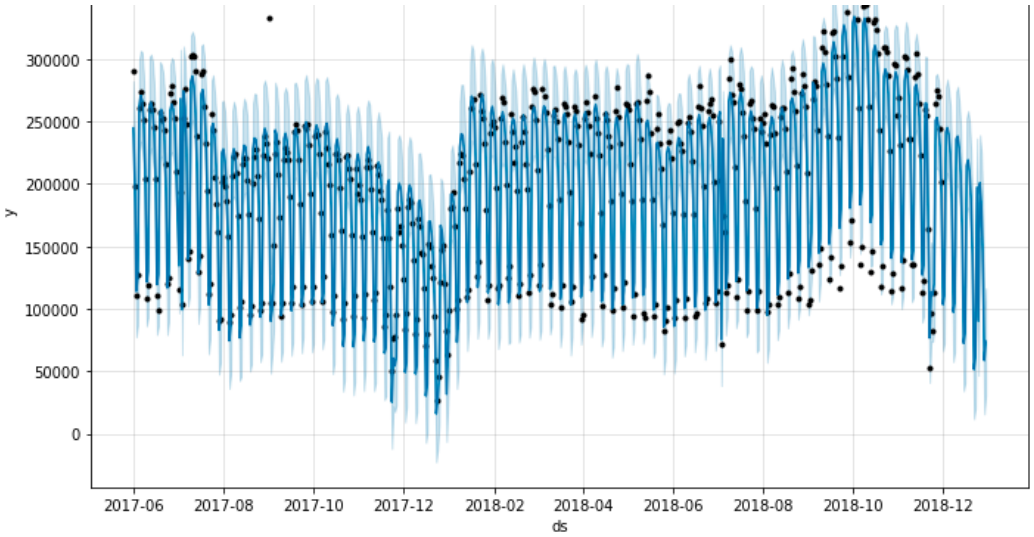
For second model, let's apply some business insights to tweak the first model. Just asking some business questions such as seasonality trends and holiday event affects, it's easy to input those information into Prophet.

We applied: *yearly_seasonality, weekly_seasonality ,holidays*( holiday events manually created here but you could also apply *Country Holidays built-in by Prophet*) and *changepoint_prior_scale* to make the model more flexible to fit the data points. Then, we added monthly seasonality.

[Code]:
*model2=Prophet(interval_width=0.95, yearly_seasonality=True, weekly_seasonality=True, holidays=us_public_holidays, changepoint_prior_scale=2)*
*model2.add_seasonality(name='monthly', period=30.5, fourier_order=5, prior_scale=0.02).*

Generate the forecasting plot below:

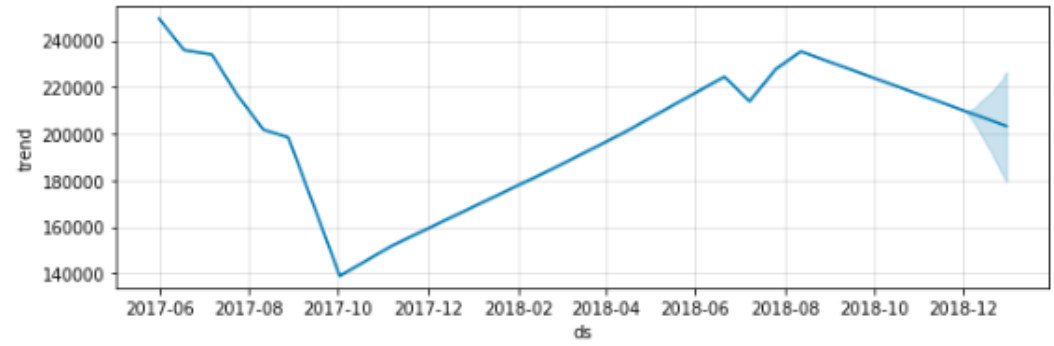


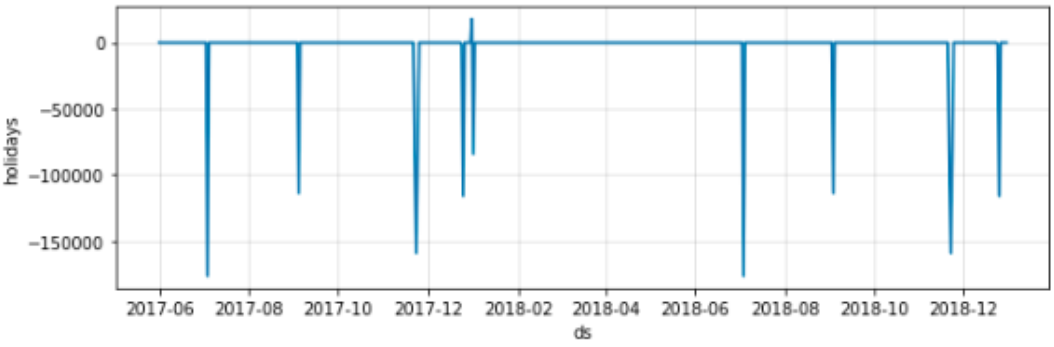


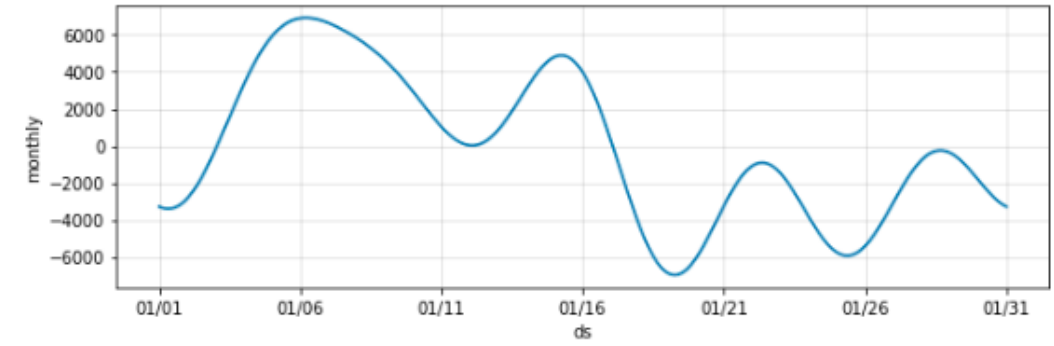

Y-Axis: Ad Spend; X-Axis: Date.

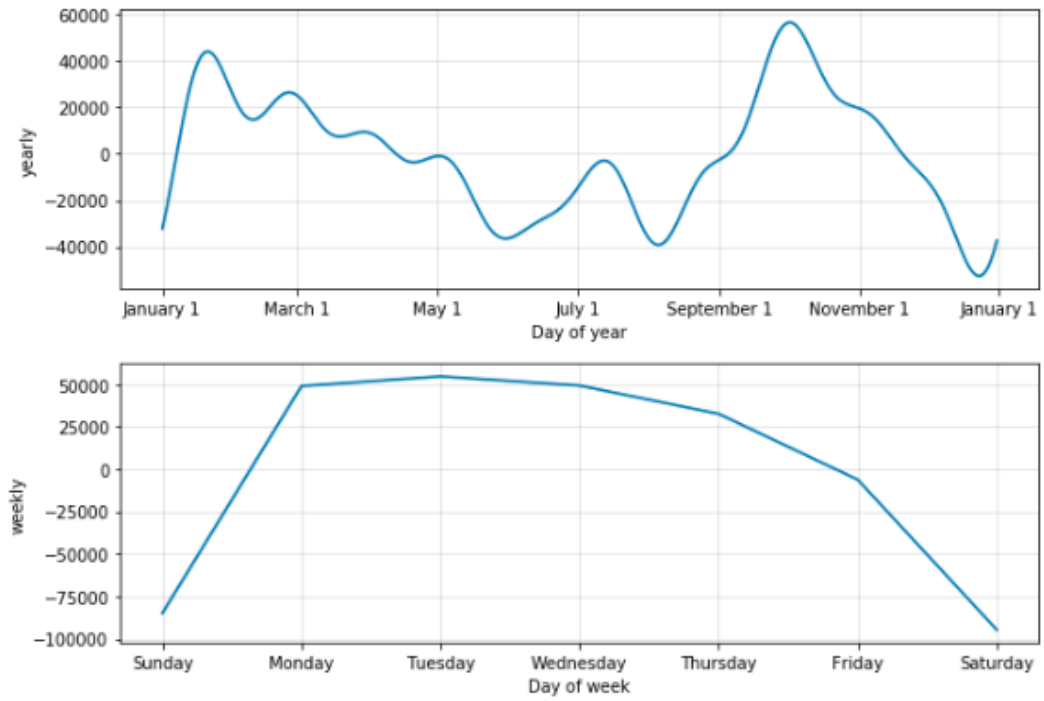
From the plot, it seems that the second model is able to catch the seasonality and fit historical data very well. (Deep blue line is forecasting spend numbers, black dots are actually spend numbers. The light blue shade is 95% confidence interval around the forecast.)

Check the trends and seasonality components:


trend
240000
220000
200000
180000
160000
140000
2017-06
2017-08
2017-10
2017-12
2018-02
2018-04
2018-06
2018-08
2018-10
2018-12
ds


holidays
0
−50000
−100000
−150000
2017-06
2017-08
2017-10
2017-12
2018-02
2018-04
2018-06
2018-08
2018-10
2018-12
ds

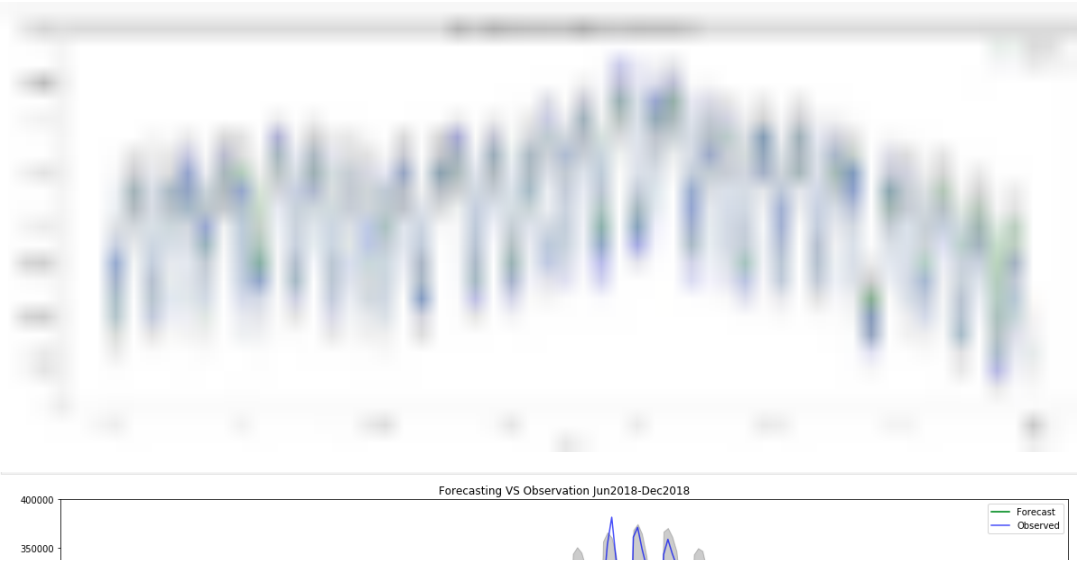

monthly
6000
4000
2000
0
−2000
−4000
−6000
01/01
01/06
01/11
01/16
01/21
01/26
01/31
ds

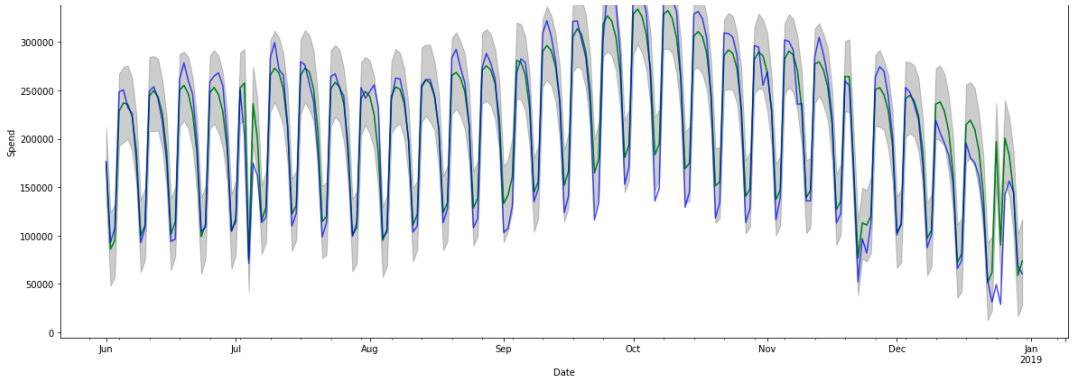



From the yearly trend, spend went up right at the beginning of the year and deeply down during the Jun, Aug and Dec. The weekly trend shows that weekdays played a big role here. And holiday events has negative affect on ad spend which means it drove ad spend decreasing and so on. You could probably check those information with the business domain knowledge.

Step 5: Validation

First let's check the fit by visualizing the forecasting line and observed line:






Y-Axis: Ad Spend; X-Axis: Date.

From the plot, it seems that the model is able to fit the data points very well thought it doesn't catch the pattern at the end of Dec. However, remember it just takes about roughly about 15mins to input all business information to get such a fair result. It doesn't require an experience expertise in time-series modeling or Machine-learning knowledge to build. Almost every analyst is able to do it ( however, skill set of Python or R is a must -have. 🙃 )

Usually, some popular error terms such Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) are used during the modeling evaluation. But I wouldn't like to discuss those errors terms here since there is only one model. ( I will discuss those error terms in my next post when comparing Prophet and classic time-series models)

Let look at the model performance by comparing the forecast value and observed value:

[Code]:

```
print("Observed value", int(forecast.y.sum()), ";Forecasting value", int(forecast.yhat.sum()), ";Forecasting lower bound",
      int(forecast.yhat_lower.sum()), ";Forecsting upper bound", int(forecast.yhat_upper.sum()))
```

```
Observed value 4404660 ;Forecasting value 5008346 ;Forecasting lower bound 3795730 ;Forecsting upper bound 6211103
```

Though the predicted value is about 13% higher than the actual value, but interval between predicted value and lower

bound is able to catch the actual value. So far, the model is doing fairly good and it takes about 15mins.

5. Closing Summary

There are many time-series analysis we can explore from now on, such as anomaly detection, forecast time-series with external data source. We have only just started.

From the practical example, it seems that Prophet provides completely automated forecasts just as its official document states. It's fast and productive which would be very useful if your organization doesn't have a very solid data science team handing predictive analytics. It saves your time to answer internal stakeholder's or client's forecasting questions without spending too much effort to build an amazing model based on classic time-series modeling techniques.

Next post, I will compare Prophet and Classic time-series forecasting techniques such as ARMIA model focusing on efficiency and performance.

## Reference and useful sources:

Facebook Prophet official document, must read if you would like to play with Prophet.

An Intro to Facebook Prophet, it generally explain what is times-series analysis and gives an overview of Facebook Prophet.

Generate Quick and Accurate Time Series Forecasts using Facebook's Prophet (with Python & R codes), it covers brief introduction of Facebook Prophet in both R and Python. It

might be useful to you if you are a R user.