fbprophet环境配置可以分为以下几个步骤：

（1）第一步，先安装VS2015，确保C++组件完全安装；

（2）第二步，必须构建虚拟环境（这点非常重要），不要使用anaconda的基础环境，避免因基础环境的库太多太杂而报错。用管理员身份运行anaconda Prompt，进入命令页面，执行：conda create -n 虚拟环境名 python==XX.XX，创建虚拟环境，然后再执行命令：conda activate 虚拟环境名，以激活虚拟环境，便于接下来安装库的位置都是在创建的虚拟环境下；

（3） 第三步，安装MingW-w64编译器类型，执行命令：conda install libpython m2w64-toolchain -c msys2；

（4）第四步，在\虚拟环境位置\ Lib \ distutils中检查是否有distutils.cfg文件,一般执行上面的命令后，都会有这个文件；

（5）第五步，执行命令：conda install numpy cython -c conda-forge；

（6）第六步，执行命令：conda install matplotlib scipy pandas -c conda-forge；

（7）第七步，执行命令：conda install pystan -c conda-forge；

（8）第八步，执行命令：conda install fbprophet -c conda-forge.

[注] fbprophet 虽然可以下载使用，但是官方文档中已经将 fbprophet 更名为 prophet ，即现在可以直接安装 prophet 。官方文档如下：

https://facebook.github.io/prophet/docs/quick\_start.html

* [Docs](https://facebook.github.io/prophet/docs/)
* [GitHub](https://github.com/facebook/prophet)
* [Docs](https://facebook.github.io/prophet/docs/)
* [GitHub](https://github.com/facebook/prophet)

[**Docs**](https://facebook.github.io/docs/)

***+*Documentation**

* [Installation](http://facebook.github.io/prophet/docs/installation.html)
  + [Using R](http://facebook.github.io/prophet/docs/installation.html#r)
  + [Using Python](http://facebook.github.io/prophet/docs/installation.html#python)
* [Quick Start](http://facebook.github.io/prophet/docs/quick_start.html)
  + [Python API](http://facebook.github.io/prophet/docs/quick_start.html#python-api)
  + [R API](http://facebook.github.io/prophet/docs/quick_start.html#r-api)
* [Saturating Forecasts](http://facebook.github.io/prophet/docs/saturating_forecasts.html)
  + [Forecasting Growth](http://facebook.github.io/prophet/docs/saturating_forecasts.html#forecasting-growth)
  + [Saturating Minimum](http://facebook.github.io/prophet/docs/saturating_forecasts.html#saturating-minimum)
* [Trend Changepoints](http://facebook.github.io/prophet/docs/trend_changepoints.html)
  + [Automatic changepoint detection in Prophet](http://facebook.github.io/prophet/docs/trend_changepoints.html#automatic-changepoint-detection-in-prophet)
  + [Adjusting trend flexibility](http://facebook.github.io/prophet/docs/trend_changepoints.html#adjusting-trend-flexibility)
  + [Specifying the locations of the changepoints](http://facebook.github.io/prophet/docs/trend_changepoints.html#specifying-the-locations-of-the-changepoints)
* [Seasonality, Holiday Effects, And Regressors](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html)
  + [Modeling Holidays and Special Events](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#modeling-holidays-and-special-events)
  + [Built-in Country Holidays](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#built-in-country-holidays)
  + [Fourier Order for Seasonalities](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#fourier-order-for-seasonalities)
  + [Specifying Custom Seasonalities](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#specifying-custom-seasonalities)
  + [Seasonalities that depend on other factors](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#seasonalities-that-depend-on-other-factors)
  + [Prior scale for holidays and seasonality](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#prior-scale-for-holidays-and-seasonality)
  + [Additional regressors](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#additional-regressors)
  + [Coefficients of additional regressors](http://facebook.github.io/prophet/docs/seasonality%2C_holiday_effects%2C_and_regressors.html#coefficients-of-additional-regressors)
* [Multiplicative Seasonality](http://facebook.github.io/prophet/docs/multiplicative_seasonality.html)
* [Uncertainty Intervals](http://facebook.github.io/prophet/docs/uncertainty_intervals.html)
  + [Uncertainty in the trend](http://facebook.github.io/prophet/docs/uncertainty_intervals.html#uncertainty-in-the-trend)
  + [Uncertainty in seasonality](http://facebook.github.io/prophet/docs/uncertainty_intervals.html#uncertainty-in-seasonality)
* [Outliers](http://facebook.github.io/prophet/docs/outliers.html)
* [Non-Daily Data](http://facebook.github.io/prophet/docs/non-daily_data.html)
  + [Sub-daily data](http://facebook.github.io/prophet/docs/non-daily_data.html#sub-daily-data)
  + [Data with regular gaps](http://facebook.github.io/prophet/docs/non-daily_data.html#data-with-regular-gaps)
  + [Monthly data](http://facebook.github.io/prophet/docs/non-daily_data.html#monthly-data)
  + [Holidays with aggregated data](http://facebook.github.io/prophet/docs/non-daily_data.html#holidays-with-aggregated-data)
* [Diagnostics](http://facebook.github.io/prophet/docs/diagnostics.html)
  + [Cross validation](http://facebook.github.io/prophet/docs/diagnostics.html#cross-validation)
  + [Parallelizing cross validation](http://facebook.github.io/prophet/docs/diagnostics.html#parallelizing-cross-validation)
  + [Hyperparameter tuning](http://facebook.github.io/prophet/docs/diagnostics.html#hyperparameter-tuning)
* [Handling Shocks](http://facebook.github.io/prophet/docs/handling_shocks.html)
  + [Treating COVID-19 lockdowns as a one-off holidays](http://facebook.github.io/prophet/docs/handling_shocks.html#treating-covid-19-lockdowns-as-a-one-off-holidays)
  + [Sense checking the trend](http://facebook.github.io/prophet/docs/handling_shocks.html#sense-checking-the-trend)
  + [Changes in seasonality between pre- and post-COVID](http://facebook.github.io/prophet/docs/handling_shocks.html#changes-in-seasonality-between-pre--and-post-covid)
* [Additional Topics](http://facebook.github.io/prophet/docs/additional_topics.html)
  + [Saving models](http://facebook.github.io/prophet/docs/additional_topics.html#saving-models)
  + [Flat trend and custom trends](http://facebook.github.io/prophet/docs/additional_topics.html#flat-trend-and-custom-trends)
  + [Updating fitted models](http://facebook.github.io/prophet/docs/additional_topics.html#updating-fitted-models)
  + [External references](http://facebook.github.io/prophet/docs/additional_topics.html#external-references)
* [Getting Help and Contributing](http://facebook.github.io/prophet/docs/contributing.html)

**Quick Start**

# Python API

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.

As an example, let’s look at a time series of the log daily page views for the Wikipedia page for [Peyton Manning](https://en.wikipedia.org/wiki/Peyton_Manning). We scraped this data using the [Wikipediatrend](https://cran.r-project.org/package%3Dwikipediatrend) package in R. Peyton Manning provides a nice example because it illustrates some of Prophet’s features, like multiple seasonality, changing growth rates, and the ability to model special days (such as Manning’s playoff and superbowl appearances). The CSV is available [here](https://github.com/facebook/prophet/blob/main/examples/example_wp_log_peyton_manning.csv).

First we’ll import the data:

1. # Python
2. import pandas as pd
3. from prophet import Prophet
4. # Python
5. df = pd.read\_csv('https://raw.githubusercontent.com/facebook/prophet/main/examples/example\_wp\_log\_peyton\_manning.csv') 3 df.head()



**0**2007-12-10 9.590761

**1**2007-12-11 8.519590

**2**2007-12-12 8.183677

**3**2007-12-13 8.072467

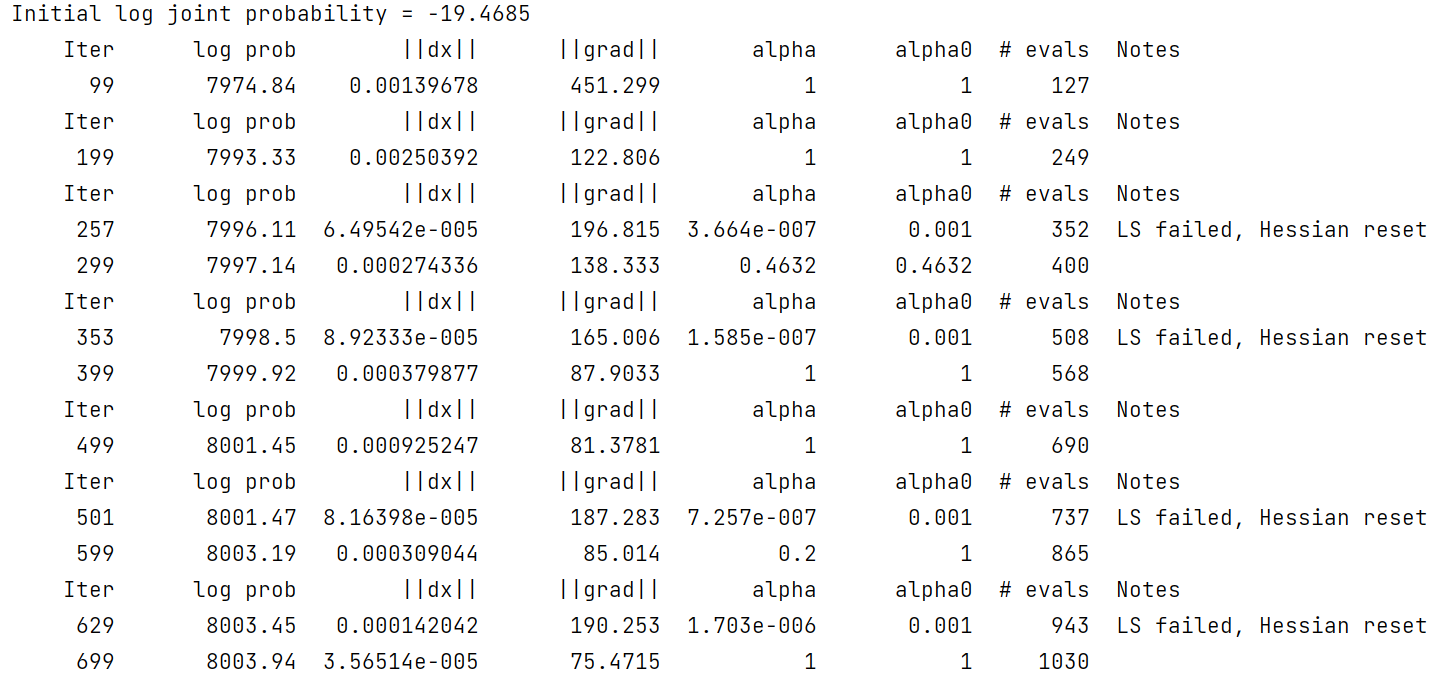
**4**2007-12-14 7.893572

**y**

**ds**

We fit the model by instantiating a new **Prophet** object. 我们通过实例化一个新的Prophet对象来拟合模型。 Any settings to the forecasting procedure are passed into the constructor. 预测过程的任何设置都会传递给构造函数。Then you call its **fit** method and pass in the historical dataframe. 然后调用它的fit方法并传入历史数据框。Fitting should take 1-5 seconds.**fit**过程耗时1-5秒。

**【注】 fit() 方法就是根据历史数据拟合、训练模型的过程。代码输出如下：**



1. # Python
2. m = Prophet() 3 m.fit(df)

Predictions are then made on a dataframe with a column **ds** containing the dates for which a prediction is to be made. 然后在dataframe上进行预测，其中**ds**列包含要进行预测的日期。You can get a suitable dataframe that extends into the future a specified nu-mber of days using the helper method **Prophet.make\_future\_dataframe**. 你可以使用辅助方法Prophet.make\_future\_dataframe得到一个合适的dataframe，它延伸到未来指定的天数。 By default it will also include the dates from the history, so we will see the model fit as well. 默认情况下，它还将包括历史日期，因此我们将看到模型也拟合。

【注】**Prophet.make\_future\_dataframe 辅助函数的作用：在我们预测之前，需要创建一些新的dataframe规定未来的时间戳。这个辅助函数就是用来创建一些新的dataframe时间戳。**

**make\_future\_dataframe()参数说明：**

**periods: 向前预测步数**

**freq: 预测单位小时为’H’，天为’D’，月为’M’**

**include\_history: 是否包含历史数据的预测（一般情况下保持默认即可）**

1. # Python
2. future = m.make\_future\_dataframe(periods=365) 3 future.tail()

|  |  |
| --- | --- |
|  | **ds** |
| **3265**20 | 17-01-15 |
| **3266**20 | 17-01-16 |
| **3267**20 | 17-01-17 |
| **3268**20 | 17-01-18 |
| **3269**20 | 17-01-19 |

The **predict** method will assign each row in **future** a predicted value which it names **yhat**. **predict方法**将为未来的每一行分配一个预测值，并将其命名为**yhat**。If you pass in historical dates, it will provide an in-sample fit. 如果传入历史日期，它将提供样本内拟合。The **forecast** object here is a new dataframe that includes a column **yhat** with the forecast, as well as columns for components and uncertainty intervals. 这里的**forecast**对象是一个新的**dataframe**，它包含一个包含预测的**yhat**列，以及用于表示**components**和不确定性区间的列（即预测上限 yhat\_upper \ 预测下限 yhat\_upper）。

1. # Python
2. forecast = m.predict(future)
3. forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper']].tail()

**3265**2017-01-15 8.211542 7.444742 8.903545

**3266**2017-01-16 8.536553 7.847804 9.211145

**3267**2017-01-17 8.323968 7.541829 9.035461

**3268**2017-01-18 8.156621 7.404457 8.830642

**3269**2017-01-19 8.168561 7.438865 8.908668

**yhat\_upper**

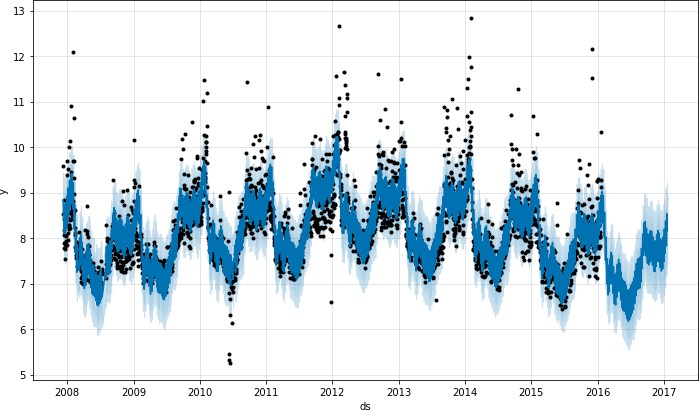
**yhat\_lower**

**yhat**

**ds**

You can plot the forecast by calling the **Prophet.plot** method and passing in your forecast dataframe.你可以使用**Prophet.plot** 绘图并传入datafraame

1. # Python
2. fig1 = m.plot(forecast)

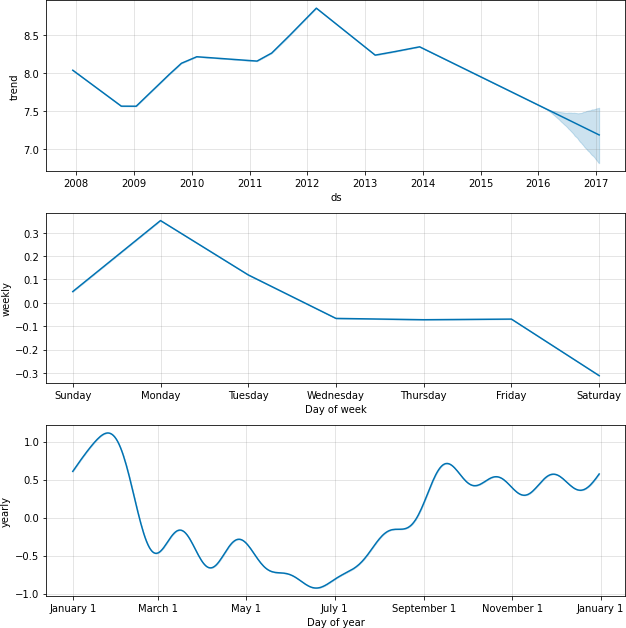


If you want to see the forecast components, you can use the Prophet.plot\_components method. By default you’ll see the trend, yearly seasonality, and weekly seasonality of the time series. If you include holidays, you’ll see those here, too.

Prophet.plot\_components()方法会画出一些细节图

一般包含序列的趋势、年季节性、周季节性

1. # Python
2. fig2 = m.plot\_components(forecast)



An interactive figure of the forecast and components can be created with plotly. 可以使用plotly创建预测和组件的交互式图形 You will need to install plotly 4.0 or above separately, as it will not by default be installed with prophet. You will also need to install the notebook and ipywidgets packages. 你需要单独安装plotly 4.0或更高版本，因为它默认不会与prophet一起安装。你还需要安装notebook和ipywidgets包。

1. # Python
2. from prophet.plot import plot\_plotly, plot\_components\_plotly 3

4 plot\_plotly(m, forecast)

1. # Python
2. plot\_components\_plotly(m, forecast)