## Prophet

### 介绍：

fbprophet是Facebook开源的一个时间序列预测算法。时间序列常见的分析方法是把时间序列分解(Decomposition of Time Series)成几个部分，分别是季节项，趋势项  ，剩余项 。也就是对所有的都有

除了加法的形式，还有乘法的形式

而prophet的算法里面考虑了四项，分别为：趋势项、季节项、剩余项和节假日效应。

其中为趋势项，表示时间序列在非周期上面的变化趋势。为周期项或季节项，代表节假日项，表示在当天是否存在节假日。表示误差项或者剩余项。Prophet算法就是通过拟合这几项，最后累加起来就得到了时间序列的预测值。

### 1，简单使用

# Python

import pandas as pd

from fbprophet import Prophet

# 读入数据集

df = pd.read\_csv('CVDI\_0001\_eta1.csv')

# 拟合模型

m = Prophet()

m.fit(df)

# 构建待预测日期数据框，periods = 365 代表除历史数据的日期外再往后推 365 天future = m.make\_future\_dataframe(periods=365)

# 预测数据集forecast = m.predict(future)

# 展示预测结果m.plot(forecast);

# 预测的成分分析绘图，展示预测中的趋势、周效应和年度效应m.plot\_components(forecast);

### 2，饱和预测

默认情况下， Prophet使用线性模型进行预测。Prophet可使用 [logistic 增长](https://en.wikipedia.org/wiki/Logistic_function)趋势模型进行预测，同时指定承载能力。

df['cap'] = 25

df['floor'] = 23

m = Prophet(growth='logistic')m.fit(df)

### 3，趋势突变点

#### 3.1 Prophet中的自动检测突变点

Prophet首先是通过在大量潜在的突变点中进行识别来监测突变点。之后对趋势变化的幅度做稀疏先验。潜在突变点的数量可以通过设置n\_changepoints 参数来指定。

from fbprophet.plot import add\_changepoints\_to\_plot

m = Prophet(changepoint\_range=0.9)

fig = m.plot(forecast)a = add\_changepoints\_to\_plot(fig.gca(), m, forecast)

#### 3.2 调整趋势的灵活性

如果趋势的变化被过度拟合或者拟合不足，可以利用输入参数 changepoint\_prior\_scale来调整稀疏先验的程度。默认这个参数被指定为0.05。

m = Prophet(changepoint\_prior\_scale=0.001)

#### 3.3指定突变点的位置

手动指定潜在突变点的位置而不是利用自动的突变点监测，可以使用changepoints参数

m = Prophet(changepoints=['2020-12-10'])

### 4，季节性，假期效果和回归量

#### 4.1对假期和特征事件建模

如果需要专门对节假日或者其它的事件进行建模，你就必须得为此创建一个新dataframe，这个数据框必须包含所有出现的节假日，不仅是历史数据集中还是待预测的时期中的。

holidays = pd.DataFrame({ 'holiday': ' holidays ', 'ds': pd.to\_datetime(['2008-01-13', '2009-01-03', '2010-01-16', '2010-01-24', '2010-02-07', '2011-01-08', '2013-01-12', '2014-01-12', '2014-01-19', '2014-02-02', '2015-01-11', '2016-01-17', '2016-01-24', '2016-02-07']), 'lower\_window': 0, 'upper\_window': 1,})

一旦这个数据框创建好了，就可以通过传入 holidays参数使得在预测时考虑上节假日效应

m = Prophet(holidays=holidays)可通过 forecast 数据框，来展示节假日效应：

forecast[(forecast['playoff']).abs() > 0]

#### 4.2 季节性的傅里叶级数

采用傅里叶级数近似于一个线性周期信号。部分和(order)中的项数是一个参数，它决定了季节性的变化有多快。默认值10通常是合适的，但是当季节性需要适应更高频率的变化时，可以增加。

m = Prophet(yearly\_seasonality=20)

#### 4.3 自定义季节性因素

如果时间序列超过两个周期，Prophet将默认适合**每周**和**每年**的季节性。对于子日（sub-daily ）时间序列，它也将适合每日的季节性。可以使用add\_seasality方法添加其它季节性。

m = Prophet(weekly\_seasonality=False)m.add\_seasonality(name='monthly', period=30.5, fourier\_order=5)

#### 4.4 对节假日和季节性设定先验规模

如果发现节假日效应被过度拟合了，通过设置参数 holidays\_prior\_scale 可以调整它们的先验规模来使之平滑，默认下该值取10。减少这个参数会降低假期效果:

m = Prophet(holidays=holidays, holidays\_prior\_scale=0.05)

seasonality\_prior\_scale参数可以用来调整模型对于季节性的拟合程度。

m.add\_seasonality( name='weekly', period=7, fourier\_order=3, prior\_scale=0.1)

#### 4.5 附加的回归量

可以使用add\_regressor方法将附加的回归量添加到模型的线性部分。

m = Prophet()m.add\_regressor('nfl\_sunday')

### 5，乘法季节性

默认情况下，Prophet能够满足附加的季节性，这意味着季节性的影响是加到趋势中得到了最后的预报。

Prophet可以通过设置seasonality\_mode='multiplicative'来建模乘法季节性

m = Prophet(seasonality\_mode='multiplicative')

乘法模型，有：forecast['yhat'] = forecast['trend']  \* (1+forecast['multiplicative\_terms']) + forecast['additive\_terms']

### 6，预测区间

默认情况下， Prophet 的返回结果中会包括预测值yhat的预测区间。当然，预测区间的估计需建立在一些重要的假设前提下。在预测时，不确定性主要来源于三个部分：趋势中的不确定性、季节效应估计中的不确定性和观测值的噪声影响。

#### 6.1趋势的不确定性

预测中，不确定性最大的来源就在于未来趋势改变的不确定性。预测区间的宽度（默认下，是 80% ）可以通过设置 interval\_width 参数来控制：

m = Prophet(interval\_width=0.95)

由于预测区间估计时假定未来将会和过去保持一样的变化频率和幅度，而这个假定可能并不正确，所以**预测区间的估计不可能完全准确。**

#### 6.2季节的不确定性

默认情况下，Prophet只会返回趋势中的不确定性和观测值噪声的影响。必须使用贝叶斯取样的方法来得到季节效应的不确定性，可通过设置 mcmc.samples 参数(默认下取0)来实现。

m = Prophet(mcmc\_samples=500)

### 7，异常值

Prophet虽能够处理历史数据中的异常值，但仅仅是将它们与趋势的变化拟合在一起，认为未来也会有类似的趋势变化。处理异常值最好的方法是移除它们，Prophet是能够处理缺失数据的。如果在历史数据中某行的值为空(NA)，但是在待预测日期数据框future中仍保留这个日期，那么Prophet依旧可以给出该行的预测值。

# 将2021年一年的数据设为缺失df.loc[(df['ds'] > '2021-01-01') & (df['ds'] < '2022-01-01'), 'y'] = Nonemodel = Prophet().fit(df)model.plot(model.predict(future));

最好的解决方法就是移除这些异常值

### 8，非日数据

#### 8.1子日数据

Prophet可以通过在ds列中传递一个带有时间戳的dataframe来对时间序列进行子日(Sub-daily)观测。时间戳的格式应该是YYYY-MM-DD - HH:MM:SS。当使用子日数据时，日季节性将自动匹配。

future = m.make\_future\_dataframe(periods=300, freq='H')

#### 8.2有规则间隔的数据

如果数据集只有每天早上6点之前的观测值，解决方案是只对有历史数据的时间窗进行预测。这里，这意味着限制未来dataframe的时间。

future2 = future.copy()future2 = future2[future2['ds'].dt.hour < 6]fcst = m.predict(future2)

#### 8.3月数据

可以使用Prophet来匹配每月的数据。然而，Prophet的基本模型是连续时间的，这意味着如果将模型与每月的数据相匹配，然后要求每天的预测，可能会得到奇怪的结果。当使用Prophet拟合月度数据时，可以通过在make\_future\_dataframe中传入频率参数只做月度的预测。

future = m.make\_future\_dataframe(periods=120, freq='M')fcst = m.predict(future)

### 9，诊断

Prophet包含时间序列交叉验证功能，以测量使用历史数据的预测误差。通过在历史记录中选择截止点来完成

from fbprophet.diagnostics import cross\_validationdf\_cv = cross\_validation(m, initial='730 days', period='180 days', horizon = '365 days')initial代表了一开始的时间是多少，period代表每隔多长时间设置一个cutoff，horizon代表每次从cutoff往后预测多少天。