The forecasting model: Facebook's Prophet

The most commonly used models for forecasting predictions are the autoregressive models. Briefly, the autoregressive model specifies that the output variable depends linearly on its own previous values and on a stochastic term (an imperfectly predictable term).

Recently, in an attempt to develop a model that could capture seasonality in time-series data, Facebook developed the famous Prophet model that is publicly available for everyone. We will use this state-of-the-art model: the Prophet model. Prophet is able to capture daily, weekly and yearly seasonality along with holiday effects, by implementing additive regression models. The mathematical equation behind the Prophet model is defined as:

y(t) = g(t) + s(t) + h(t) + e(t)

with, g(t) representing the trend. Prophet uses a piecewise linear model for trend forecasting.

s(t) represents periodic changes (weekly, monthly, yearly).

h(t) represents the effects of holidays (recall: Holidays impact businesses).

e(t) is the error term.

The Prophet model fitting procedure is usually very fast (even for thousands of observations) and it does not require any data pre-processing. It deals also with missing data and outliers.

TESLA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# Load the dataset using pandas
data = pd.read_csv("../input/tesla-share/TSLA (1).csv")
data.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2010-06-29	3.800	5.000	3.508	4.778	4.778	93831500
1	2010-06-30	5.158	6.084	4.660	4.766	4.766	85935500
2	2010-07-01	5.000	5.184	4.054	4.392	4.392	41094000
3	2010-07-02	4.600	4.620	3.742	3.840	3.840	25699000
4	2010-07-06	4.000	4.000	3.166	3.222	3.222	34334500

```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	2683.000000	2683.000000	2683.000000	2683.000000	2683.000000	2.683000e+03
mean	70.914358	72.474707	69.285925	71.013916	71.013916	3.177323e+07
std	127.180265	130.197125	123.962726	127.525574	127.525574	2.913656e+07
min	3.228000	3.326000	2.996000	3.160000	3.160000	5.925000e+05
25%	7.592000	7.697000	7.424000	7.578000	7.578000	1.131475e+07
50%	44.698002	45.500000	44.049999	44.660000	44.660000	2.505200e+07
75%	61.400002	62.524000	60.233999	61.590999	61.590999	4.107750e+07
max	891.380005	900.400024	871.599976	883.090027	883.090027	3.046940e+08

```
# Select only the important features i.e. the date and price
data = data[["Date","Close"]] # select Date and Price
# Rename the features: These names are NEEDED for the model fitting
data = data.rename(columns = {"Date":"ds","Close":"y"}) #renaming the columns of the dataset
data.head()
```

	ds	y
0	2010-06-29	4.778
1	2010-06-30	4.766
2	2010-07-01	4.392
3	2010-07-02	3.840
4	2010-07-06	3.222

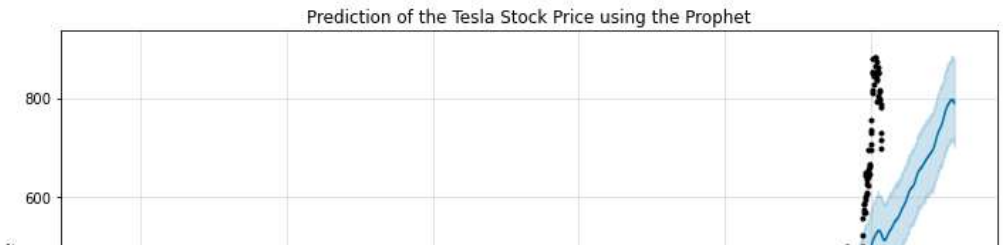
```
!pip install fbprophet
```

```
Requirement already satisfied: fbprophet in /opt/conda/lib/python3.7/site-packages (0.7.1)
Requirement already satisfied: Cython>=0.22 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (0.29.21)
Requirement already satisfied: cmdstanpy==0.9.5 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (0.9.5)
Requirement already satisfied: pystan>=2.14 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (2.19.1.1)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (1.19.5)
Requirement already satisfied: pandas>=1.0.4 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (1.2.0)
Requirement already satisfied: matplotlib>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (3.3.3)
Requirement already satisfied: LunarCalendar>=0.0.9 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (0.0.9)
Requirement already satisfied: convertdate>=2.1.2 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (2.3.1)
Requirement already satisfied: holidays>=0.10.2 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (0.10.5.2)
Requirement already satisfied: setuptools-git>=1.2 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (1.2)
Requirement already satisfied: python-dateutil>=2.8.0 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (2.8.1)
Requirement already satisfied: tqdm>=4.36.1 in /opt/conda/lib/python3.7/site-packages (from fbprophet) (4.55.1)
Requirement already satisfied: pymeeus!=0.3.8,<=1,>=0.3.6 in /opt/conda/lib/python3.7/site-packages (from convertdate>=2.1.2->f
Requirement already satisfied: pytz>=2014.10 in /opt/conda/lib/python3.7/site-packages (from convertdate>=2.1.2->fbprophet) (20
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages (from holidays>=0.10.2->fbprophet) (1.15.0)
Requirement already satisfied: korean-lunar-calendar in /opt/conda/lib/python3.7/site-packages (from holidays>=0.10.2->fbprophe
Requirement already satisfied: hijri-converter in /opt/conda/lib/python3.7/site-packages (from holidays>=0.10.2->fbprophet) (2.
Requirement already satisfied: ephem>=3.7.5.3 in /opt/conda/lib/python3.7/site-packages (from LunarCalendar>=0.0.9->fbprophet)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->fbprophet)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->fbprophet) (7.2
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->fbprophet) (0.10
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /opt/conda/lib/python3.7/site-packages (from matplot
```

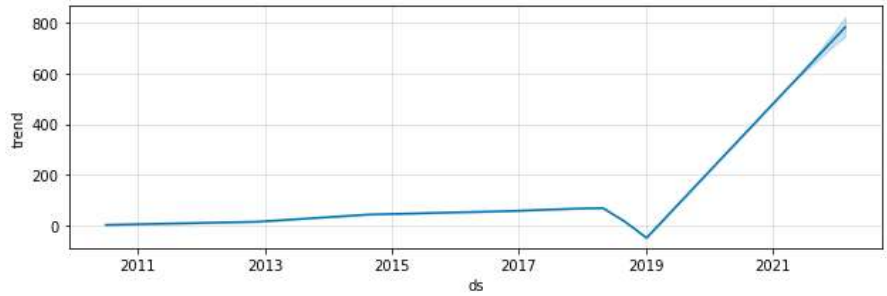
```
from fbprophet import Prophet
m = Prophet(daily_seasonality = True) # the Prophet class (model)
m.fit(data) # fit the model using all data
```

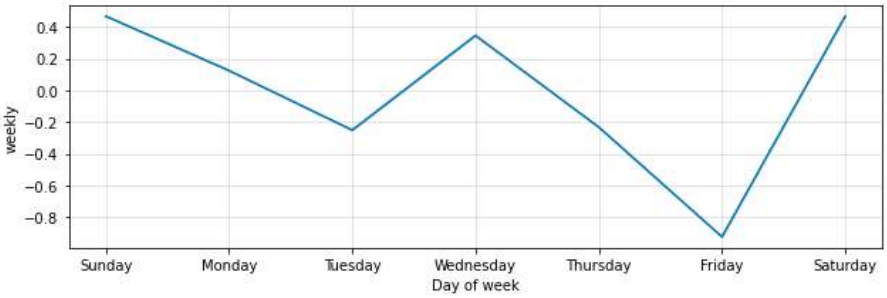
```
<fbprophet.forecaster.Prophet at 0x7f8252b72a10>
```

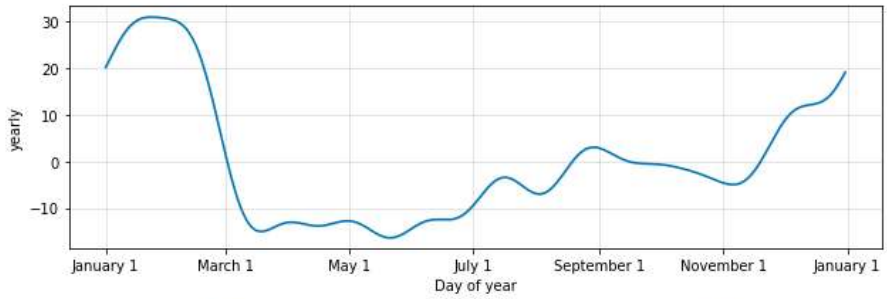
```
future = m.make_future_dataframe(periods=365) #we need to specify the number of days in future
prediction = m.predict(future)
m.plot(prediction)
plt.title("Prediction of the Tesla Stock Price using the Prophet")
plt.xlabel("Date")
plt.ylabel("Close Stock Price")
plt.show()
```

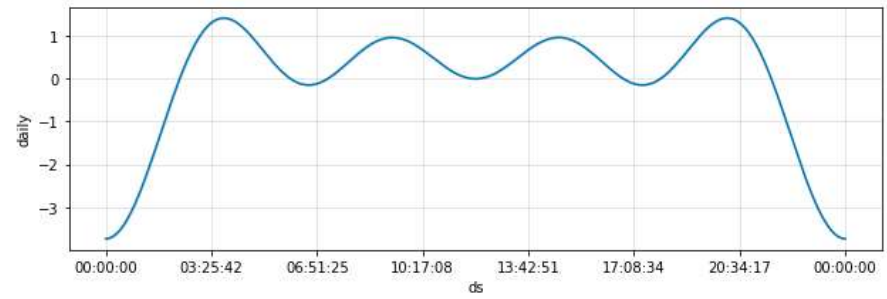



```
m.plot_components(prediction)
plt.show()
```









TCS

```
# Load the dataset using pandas
data = pd.read_csv("../input/tcs-share/TCS.NS (1).csv")
data.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-25	149.837494	149.837494	122.375000	123.493752	92.474930	136928.0
1	2004-08-26	124.000000	124.625000	121.912498	122.375000	91.637184	40443200.0
2	2004-08-27	122.800003	122.800003	119.820000	120.332497	90.107712	30646000.0

```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	4070.000000	4070.000000	4070.000000	4070.000000	4070.000000	4.070000e+03
mean	917.644542	928.156148	906.372446	917.141210	834.383126	3.739476e+06
std	715.504286	722.729269	707.584970	715.011337	709.753664	3.325465e+06
min	112.000000	116.112503	103.837502	111.550003	88.550110	0.000000e+00
25%	268.562500	272.050003	263.525002	267.206238	207.602928	1.983859e+06
50%	655.000000	661.000000	646.837524	654.975006	551.727844	2.895070e+06
75%	1282.399994	1295.587494	1271.612518	1282.168701	1156.497222	4.491241e+06
max	3308.949951	3339.800049	3278.649902	3308.800049	3308.800049	8.806715e+07

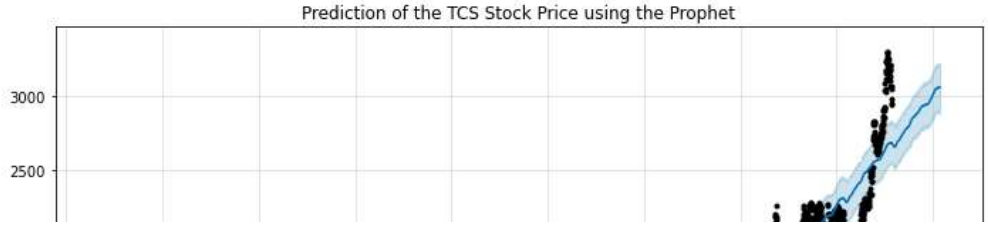
```
# Select only the important features i.e. the date and price
data = data[["Date","Close"]] # select Date and Price
# Rename the features: These names are NEEDED for the model fitting
data = data.rename(columns = {"Date":"ds","Close":"y"}) #renaming the columns of the dataset
data.head()
```

	ds	y
0	2004-08-25	123.493752
1	2004-08-26	122.375000
2	2004-08-27	120.332497
3	2004-08-30	123.345001
4	2004-08-31	123.512497

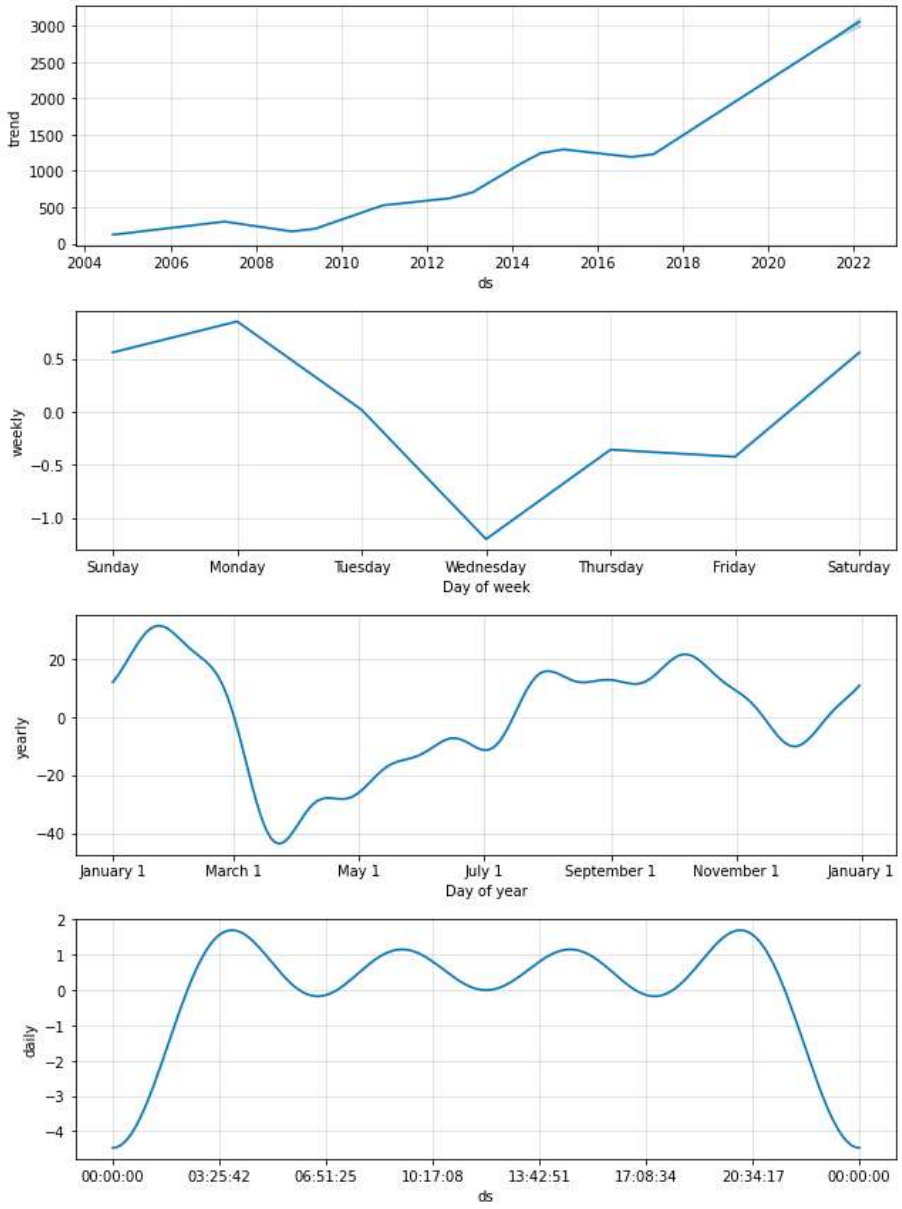
```
from fbprophet import Prophet
m = Prophet(daily_seasonality = True) # the Prophet class (model)
m.fit(data) # fit the model using all data
```

```
<fbprophet.forecaster.Prophet at 0x7f823006c5d0>
```

```
future = m.make_future_dataframe(periods=365) #we need to specify the number of days in future
prediction = m.predict(future)
m.plot(prediction)
plt.title("Prediction of the TCS Stock Price using the Prophet")
plt.xlabel("Date")
plt.ylabel("Close Stock Price")
plt.show()
```




```
m.plot_components(prediction)
plt.show()
```



S&P Global

```
# Load the dataset using pandas
data = pd.read_csv("../input/sp-global/GSPC.csv")
data.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	1927-12-30	17.660000	17.660000	17.660000	17.660000	17.660000	0
1	1928-01-03	17.760000	17.760000	17.760000	17.760000	17.760000	0

```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	23398.000000	23398.000000	23398.000000	23398.000000	23398.000000	2.339800e+04
mean	476.694263	500.156864	494.122959	497.340753	497.340753	7.825474e+08
std	759.883402	751.975489	743.557358	748.073634	748.073634	1.502654e+09
min	0.000000	4.400000	4.400000	4.400000	4.400000	0.000000e+00
25%	9.280000	23.910000	23.910000	23.910000	23.910000	1.300000e+06
50%	30.670000	100.535000	98.924999	99.715000	99.715000	1.759500e+07
75%	849.460007	864.167496	840.169998	850.110001	850.110001	5.666275e+08
max	3939.610107	3950.429932	3923.850098	3934.830078	3934.830078	1.145623e+10

```
# Select only the important features i.e. the date and price
data = data[["Date","Close"]] # select Date and Price
# Rename the features: These names are NEEDED for the model fitting
data = data.rename(columns = {"Date":"ds","Close":"y"}) #renaming the columns of the dataset
data.head()
```

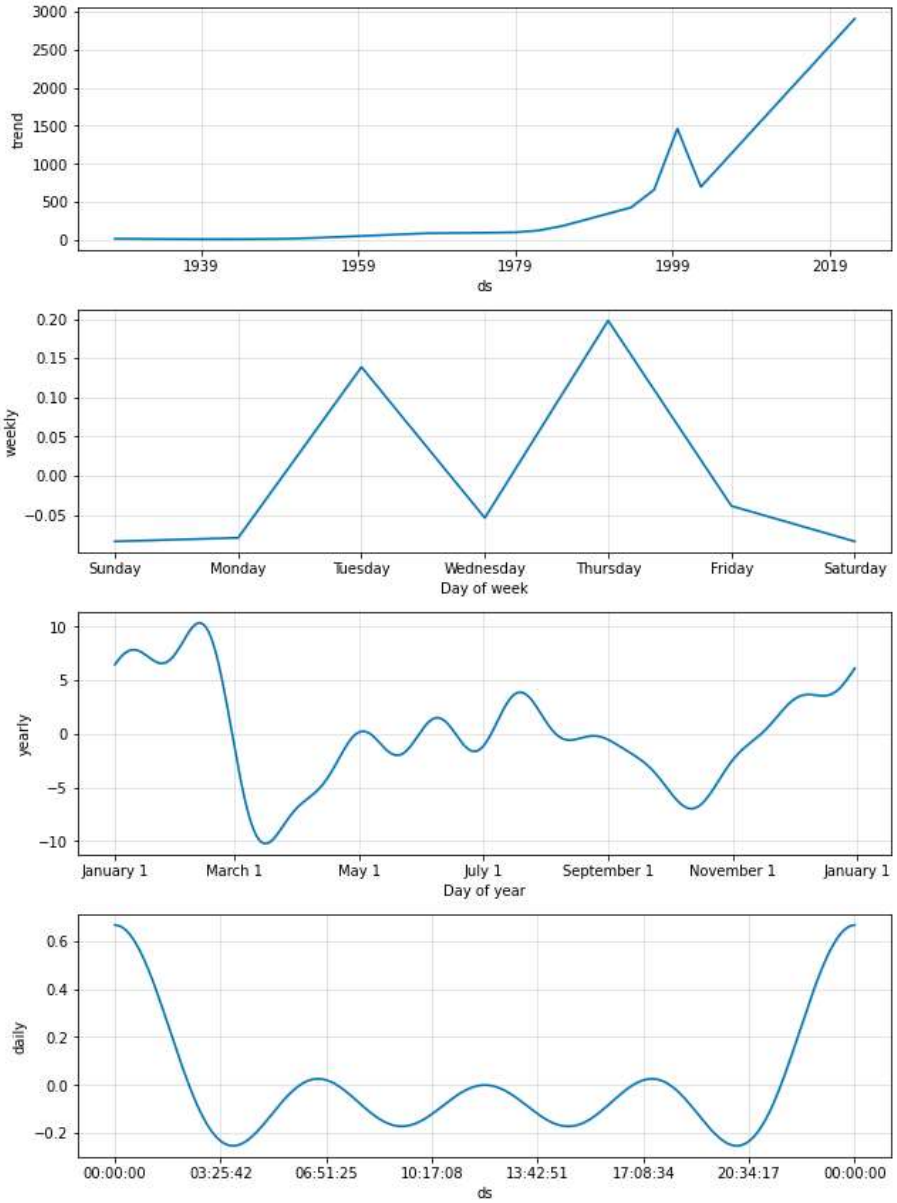
	ds	y
0	1927-12-30	17.660000
1	1928-01-03	17.760000
2	1928-01-04	17.719999
3	1928-01-05	17.549999
4	1928-01-06	17.660000

```
from fbprophet import Prophet
m = Prophet(daily_seasonality = True) # the Prophet class (model)
m.fit(data) # fit the model using all data
```

```
<fbprophet.forecaster.Prophet at 0x7f822e65f250>
```

```
future = m.make_future_dataframe(periods=365) #we need to specify the number of days in future
prediction = m.predict(future)
m.plot(prediction)
plt.title("Prediction of the S&P Global Stock Price using the Prophet")
plt.xlabel("Date")
plt.ylabel("Close Stock Price")
plt.show()
```


```
m.plot_components(prediction)
plt.show()
```



BitCoin

```
# Load the dataset using pandas
data = pd.read_csv("../input/bitcoin/BTC-USD.csv")
data.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	465.864014	468.174011	452.421997	457.334015	457.334015	21056800.0
1	2014-09-18	456.859985	456.859985	413.104004	424.440002	424.440002	34483200.0

```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	2349.000000	2349.000000	2349.000000	2349.000000	2349.000000	2.349000e+03
mean	5721.921570	5883.101031	5556.000000	5741.127480	5741.127480	1.052634e+10
std	7156.218635	7420.178134	6874.922195	7205.304192	7205.304192	1.595112e+10
min	176.897003	211.731003	171.509995	178.102997	178.102997	5.914570e+06
25%	450.559998	455.587006	444.330994	450.303986	450.303986	6.520380e+07
50%	4000.256836	4082.216064	3906.179932	4005.526611	4005.526611	3.488450e+09
75%	8720.080078	8890.456055	8492.932617	8728.469727	8728.469727	1.674266e+10
max	57532.738281	58330.570313	55672.609375	57539.945313	57539.945313	1.233206e+11

```
# Select only the important features i.e. the date and price
data = data[["Date","Close"]] # select Date and Price
# Rename the features: These names are NEEDED for the model fitting
data = data.rename(columns = {"Date":"ds","Close":"y"}) #renaming the columns of the dataset
data.head()
```

	ds	y
0	2014-09-17	457.334015
1	2014-09-18	424.440002
2	2014-09-19	394.795990
3	2014-09-20	408.903992
4	2014-09-21	398.821014

```
from fbprophet import Prophet
m = Prophet(daily_seasonality = True) # the Prophet class (model)
m.fit(data) # fit the model using all data
```

```
<fbprophet.forecaster.Prophet at 0x7f82301e9cd0>
```

```
future = m.make_future_dataframe(periods=365) #we need to specify the number of days in future
prediction = m.predict(future)
m.plot(prediction)
plt.title("Prediction of the Bitcoin Stock Price using the Prophet")
plt.xlabel("Date")
plt.ylabel("Close Stock Price")
plt.show()
```


```
m.plot_components(prediction)
plt.show()
```