Technical Analysis Library in Python Documentation

Release 0.1.4

Dario Lopez Padial (Bukosabino)

Nov 29, 2020
## Contents

1  Installation (python $\geq$ v3.6) ................................................. 3
2  Examples .................................................................................. 5
3  Motivation .................................................................................. 7
4  Contents
   4.1  Documentation .................................................................. 9
5  Indices and tables ....................................................................... 61
   Python Module Index .................................................................. 63
   Index ....................................................................................... 65
It is a Technical Analysis library to financial time series datasets (open, close, high, low, volume). You can use it to do feature engineering from financial datasets. It is builded on Python Pandas library.
CHAPTER
ONE

INSTALLATION (PYTHON >= V3.6)

```bash
> virtualenv -p python3 virtualenvironment
> source virtualenvironment/bin/activate
> pip install ta
```
Example adding all features:

```python
import pandas as pd
from ta import add_all_ta_features
from ta.utils import dropna

# Load datas
df = pd.read_csv('ta/tests/data/datas.csv', sep=',')

# Clean NaN values
df = dropna(df)

# Add ta features filling NaN values
df = add_all_ta_features(
    df, open="Open", high="High", low="Low", close="Close", volume="Volume_BTC",
   fillna=True)
```

Example adding a particular feature:

```python
import pandas as pd
from ta.utils import dropna
from ta.volatility import BollingerBands

# Load datas
df = pd.read_csv('ta/tests/data/datas.csv', sep=',')

# Clean NaN values
df = dropna(df)

# Initialize Bollinger Bands Indicator
indicator_bb = BollingerBands(close=df['Close'], window=20, window_dev=2)

# Add Bollinger Bands features
df['bb_bbm'] = indicator_bb.bollinger_mavg()
df['bb_bbh'] = indicator_bb.bollinger_hband()
df['bb_bbl'] = indicator_bb.bollinger_lband()

# Add Bollinger Band high indicator
df['bb_bbhi'] = indicator_bb.bollinger_hband_indicator()

# Add Bollinger Band low indicator
df['bb_bbli'] = indicator_bb.bollinger_lband_indicator()
```
CHAPTER

THREE

MOTIVATION

- **English:** https://towardsdatascience.com/technical-analysis-library-to-financial-datasets-with-pandas-python-4b2b390d3543
- **Spanish:** https://medium.com/datos-y-ciencia/biblioteca-de-an%C3%A1lisis-t%C3%A9cnico-sobre-series-temporales-financieras-para-machine-learning-con-cb28f9427d0
4.1 Documentation

It is a Technical Analysis library useful to do feature engineering from financial time series datasets (Open, Close, High, Low, Volume). It is built on Pandas and Numpy.

4.1.1 Momentum Indicators

Momentum Indicators.


Awesome Oscillator

From: https://www.tradingview.com/wiki/Awesome_Oscillator_(AO)

The Awesome Oscillator is an indicator used to measure market momentum. AO calculates the difference of a 34 Period and 5 Period Simple Moving Averages. The Simple Moving Averages that are used are not calculated using closing price but rather each bar’s midpoints. AO is generally used to affirm trends or to anticipate possible reversals.

From: https://www.ifcm.co.uk/ntx-indicators/awesome-oscillator

Awesome Oscillator is a 34-period simple moving average, plotted through the central points of the bars (H+L)/2, and subtracted from the 5-period simple moving average, graphed across the central points of the bars (H+L)/2.

MEDIAN PRICE = (HIGH+LOW)/2

AO = SMA(MEDIAN PRICE, 5) - SMA(MEDIAN PRICE, 34)

where

SMA — Simple Moving Average.

Parameters

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- window1 (int) – short period.
- window2 (int) – long period.
- fillna (bool) – if True, fill nan values with -50.
**awesome_oscillator**() → pandas.core.series.Series
Awesome Oscillator

**Returns** New feature generated.

**Return type** pandas.Series

class ta.momentum.KAMAIndicator(close: pandas.core.series.Series, window: int = 10, pow1: int = 2, pow2: int = 30, fillna: bool = False)

Kaufman’s Adaptive Moving Average (KAMA)

Moving average designed to account for market noise or volatility. KAMA will closely follow prices when the price swings are relatively small and the noise is low. KAMA will adjust when the price swings widen and follow prices from a greater distance. This trend-following indicator can be used to identify the overall trend, time turning points and filter price movements.

https://www.tradingview.com/ideas/kama/

**Parameters**
- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- pow1 (int) – number of periods for the fastest EMA constant.
- pow2 (int) – number of periods for the slowest EMA constant.
- fillna (bool) – if True, fill nan values.

**kama**() → pandas.core.series.Series
Kaufman’s Adaptive Moving Average (KAMA)

**Returns** New feature generated.

**Return type** pandas.Series

class ta.momentum.PercentagePriceOscillator(close: pandas.core.series.Series, window_slow: int = 26, window_fast: int = 12, window_sign: int = 9, fillna: bool = False)

The Percentage Price Oscillator (PPO) is a momentum oscillator that measures the difference between two moving averages as a percentage of the larger moving average.


**Parameters**
- window_slow (int) – n period long-term.
- window_fast (int) – n period short-term.
- window_sign (int) – n period to signal.
- fillna (bool) – if True, fill nan values.

**ppo**()
Percentage Price Oscillator Line

**Returns** New feature generated.

**Return type** pandas.Series

**ppo_hist**()
Percentage Price Oscillator Histogram

**Returns** New feature generated.
The Percentage Volume Oscillator (PVO) is a momentum oscillator for volume. The PVO measures the difference between two volume-based moving averages as a percentage of the larger moving average.


Parameters

- `window_slow (int)` – n period long-term.
- `window_fast (int)` – n period short-term.
- `window_sign (int)` – n period to signal.
- `fillna (bool)` – if True, fill nan values.

def pvo():
    return pandas.core.series.Series

PVO Line

Returns New feature generated.
Return type pandas.Series

def pvo_hist():
    return pandas.core.series.Series

Histgram

Returns New feature generated.
Return type pandas.Series

def pvo_signal():
    return pandas.core.series.Series

Signal Line

Returns New feature generated.
Return type pandas.Series

class ta.momentum.PercentageVolumeOscillator:

The Percentage Volume Oscillator (PVO) is a momentum oscillator for volume. The PVO measures the difference between two volume-based moving averages as a percentage of the larger moving average.


Parameters

- `volume (pandas.core.series.Series, window_slow: int = 26, window_fast: int = 12, window_sign: int = 9, fillna: bool = False)`

The Rate-of-Change (ROC) indicator, which is also referred to as simply Momentum, is a pure momentum oscillator that measures the percent change in price from one period to the next. The ROC calculation compares the current price with the price “n” periods ago. The plot forms an oscillator that fluctuates above and below the zero line as the Rate-of-Change moves from positive to negative. As a momentum oscillator, ROC signals include centerline crossovers, divergences and overbought-oversold readings. Divergences fail to foreshadow reversals more often than not, so this article will forgo a detailed discussion on them. Even though centerline crossovers are prone to whipsaw, especially short-term, these crossovers can be used to identify the overall trend. Identifying overbought or oversold extremes comes naturally to the Rate-of-Change oscillator.


Parameters
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

roc() → pandas.core.series.Series
Rate of Change (ROC)

Returns New feature generated.
Return type pandas.Series

class ta.momentum.RSIIndicator (close: pandas.core.series.Series, window: int = 14, fillna: bool = False)
Relative Strength Index (RSI)

Compares the magnitude of recent gains and losses over a specified time period to measure speed and change of
price movements of a security. It is primarily used to attempt to identify overbought or oversold conditions in
the trading of an asset.

https://www.investopedia.com/terms/r/rsi.asp

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

rsi() → pandas.core.series.Series
Relative Strength Index (RSI)

Returns New feature generated.
Return type pandas.Series

class ta.momentum.StochRSIIndicator (close: pandas.core.series.Series, window: int = 14,
smooth1: int = 3, smooth2: int = 3, fillna: bool = False)
Stochastic RSI

The StochRSI oscillator was developed to take advantage of both momentum indicators in order to create a
more sensitive indicator that is attuned to a specific security’s historical performance rather than a generalized
analysis of price change.

https://www.investopedia.com/terms/s/stochrsi.asp

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period
• smooth1 (int) – moving average of Stochastic RSI
• smooth2 (int) – moving average of %K
• fillna (bool) – if True, fill nan values.

stochrsi()  
Stochastic RSI  

Returns New feature generated.
Return type pandas.Series
**stochrsi_d()**
Stochastic RSI %d

- **Returns**: New feature generated.
- **Return type**: pandas.Series

**stochrsi_k()**
Stochastic RSI %k

- **Returns**: New feature generated.
- **Return type**: pandas.Series

**class ta.momentum.StochasticOscillator**

- **parameters**
  - `high`: pandas.core.series.Series
  - `low`: pandas.core.series.Series
  - `close`: pandas.core.series.Series
  - `window`: int = 14
  - `smooth_window`: int = 3
  - `fillna`: bool = False

Stochastic Oscillator

Developed in the late 1950s by George Lane. The stochastic oscillator presents the location of the closing price of a stock in relation to the high and low range of the price of a stock over a period of time, typically a 14-day period.


- **Parameters**
  - `close` (pandas.Series) – dataset ‘Close’ column.
  - `high` (pandas.Series) – dataset ‘High’ column.
  - `low` (pandas.Series) – dataset ‘Low’ column.
  - `window` (int) – n period.
  - `smooth_window` (int) – sma period over stoch_k.
  - `fillna` (bool) – if True, fill nan values.

**stoch()** → pandas.core.series.Series
Stochastic Oscillator

- **Returns**: New feature generated.
- **Return type**: pandas.Series

**stoch_signal()** → pandas.core.series.Series
Signal Stochastic Oscillator

- **Returns**: New feature generated.
- **Return type**: pandas.Series

**class ta.momentum.TSIIndicator**

- **parameters**
  - `close`: pandas.core.series.Series
  - `window_slow`: int = 25
  - `window_fast`: int = 13
  - `fillna`: bool = False

True strength index (TSI)

Shows both trend direction and overbought/oversold conditions.


- **Parameters**
  - `close` (pandas.Series) – dataset ‘Close’ column.
  - `window_slow` (int) – high period.
• **window_fast** *(int)* – low period.
• **fillna** *(bool)* – if True, fill nan values.

```python
tsi() \to \text{pandas.core.series.Series}
```

**True strength index (TSI)**

**Returns** New feature generated.

**Return type** pandas.Series

class **ta.momentum.UltimateOscillator**

```python
(class) \to \text{pandas.core.series.Series}
```

**Ultimate Oscillator**

Larry Williams’ (1976) signal, a momentum oscillator designed to capture momentum across three different timeframes.


BP = Close - Minimum(Low or Prior Close). TR = Maximum(High or Prior Close) - Minimum(Low or Prior Close) Average7 = (7-period BP Sum) / (7-period TR Sum) Average14 = (14-period BP Sum) / (14-period TR Sum) Average28 = (28-period BP Sum) / (28-period TR Sum)

UO = 100 x [(4 x Average7)+(2 x Average14)+Average28]/(4+2+1)

**Parameters**

• **high** *(pandas.Series)* – dataset ‘High’ column.
• **low** *(pandas.Series)* – dataset ‘Low’ column.
• **close** *(pandas.Series)* – dataset ‘Close’ column.
• **window1** *(int)* – short period.
• **window2** *(int)* – medium period.
• **window3** *(int)* – long period.
• **weight1** *(float)* – weight of short BP average for UO.
• **weight2** *(float)* – weight of medium BP average for UO.
• **weight3** *(float)* – weight of long BP average for UO.
• **fillna** *(bool)* – if True, fill nan values with 50.

```python
ultimate_oscillator() \to \text{pandas.core.series.Series}
```

**Williams %R**

Developed by Larry Williams, Williams %R is a momentum indicator that is the inverse of the Fast Stochastic Oscillator. Also referred to as %R, Williams %R reflects the level of the close relative to the highest high for the look-back period. In contrast, the Stochastic Oscillator reflects the level of the close relative to the lowest low. %R corrects for the inversion by multiplying the raw value by -100. As a result, the Fast Stochastic Oscillator
and Williams %R produce the exact same lines, only the scaling is different. Williams %R oscillates from 0 to -100.

Readings from 0 to -20 are considered overbought. Readings from -80 to -100 are considered oversold.

Unsurprisingly, signals derived from the Stochastic Oscillator are also applicable to Williams %R.

\[
\%R = \frac{(\text{Highest High} - \text{Close})}{(\text{Highest High} - \text{Lowest Low})} \times -100
\]

Lowest Low = lowest low for the look-back period
Highest High = highest high for the look-back period
\%R is multiplied by -100 to correct the inversion and move the decimal.


The Williams %R oscillates from 0 to -100. When the indicator produces readings from 0 to -20, this indicates overbought market conditions. When readings are -80 to -100, it indicates oversold market conditions.

**Parameters**

- **high** (*pandas.Series*) – dataset ‘High’ column.
- **close** (*pandas.Series*) – dataset ‘Close’ column.
- **lbp** (*int*) – lookback period.
- **fillna** (*bool*) – if True, fill nan values with -50.

```python
williams_r() \rightarrow pandas.core.series.Series
```

Williams %R

**Returns**

New feature generated.

**Return type**

*pandas.Series*

4.1. Documentation 15
• `window2` (int) – long period.
• `fillna` (bool) – if True, fill nan values with -50.

**Returns**
New feature generated.

**Return type**
pandas.Series

```python
ta.momentum.kama(close, window=10, pow1=2, pow2=30, fillna=False) → pandas.core.series.Series
```

Kaufman’s Adaptive Moving Average (KAMA)

Moving average designed to account for market noise or volatility. KAMA will closely follow prices when the price swings are relatively small and the noise is low. KAMA will adjust when the price swings widen and follow prices from a greater distance. This trend-following indicator can be used to identify the overall trend, time turning points and filter price movements.

https://www.tradingview.com/ideas/kama/

**Parameters**

• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n number of periods for the efficiency ratio.
• pow1 (int) – number of periods for the fastest EMA constant.
• pow2 (int) – number of periods for the slowest EMA constant.
• fillna (bool) – if True, fill nan values.

**Returns**
New feature generated.

**Return type**
pandas.Series

```python
ta.momentum.ppo(close: pandas.core.series.Series, window_slow: int = 26, window_fast: int = 12, window_sign: int = 9, fillna: bool = False) → pandas.core.series.Series
```

The Percentage Price Oscillator (PPO) is a momentum oscillator that measures the difference between two moving averages as a percentage of the larger moving average.


**Parameters**

• close (pandas.Series) – dataset ‘Price’ column.
• window_slow (int) – n period long-term.
• window_fast (int) – n period short-term.
• window_sign (int) – n period to signal.
• fillna (bool) – if True, fill nan values.

**Returns**
New feature generated.

**Return type**
pandas.Series

```python
ta.momentum.ppo_hist(close: pandas.core.series.Series, window_slow: int = 26, window_fast: int = 12, window_sign: int = 9, fillna: bool = False) → pandas.core.series.Series
```

The Percentage Price Oscillator (PPO) is a momentum oscillator that measures the difference between two moving averages as a percentage of the larger moving average.


**Parameters**

• close (pandas.Series) – dataset ‘Price’ column.
• window_slow (int) – n period long-term.
• **window_fast** (int) – n period short-term.
• **window_sign** (int) – n period to signal.
• **fillna** (bool) – if True, fill nan values.

**Returns** New feature generated.

**Return type**  pandas.Series

ta.momentum.ppo_signal(close: pandas.core.series.Series, window_slow=26, window_fast=12, window_sign=9,fillna=False) → pandas.core.series.Series

The Percentage Price Oscillator (PPO) is a momentum oscillator that measures the difference between two moving averages as a percentage of the larger moving average.


**Parameters**

• **close** (pandas.Series) – dataset ‘Price’ column.
• **window_slow** (int) – n period long-term.
• **window_fast** (int) – n period short-term.
• **window_sign** (int) – n period to signal.
• **fillna** (bool) – if True, fill nan values.

**Returns** New feature generated.

**Return type**  pandas.Series

ta.momentum.pvo(volume: pandas.core.series.Series, window_slow: int = 26, window_fast: int = 12, window_sign: int = 9,fillna: bool = False) → pandas.core.series.Series

The Percentage Volume Oscillator (PVO) is a momentum oscillator for volume. The PVO measures the difference between two volume-based moving averages as a percentage of the larger moving average.


**Parameters**

• **volume** (pandas.Series) – dataset ‘Volume’ column.
• **window_slow** (int) – n period long-term.
• **window_fast** (int) – n period short-term.
• **window_sign** (int) – n period to signal.
• **fillna** (bool) – if True, fill nan values.

**Returns** New feature generated.

**Return type**  pandas.Series

ta.momentum.pvo_hist(volume: pandas.core.series.Series, window_slow: int = 26, window_fast: int = 12, window_sign: int = 9,fillna: bool = False) → pandas.core.series.Series

The Percentage Volume Oscillator (PVO) is a momentum oscillator for volume. The PVO measures the difference between two volume-based moving averages as a percentage of the larger moving average.


**Parameters**

• **volume** (pandas.Series) – dataset ‘Volume’ column.
• **window_slow** (int) – n period long-term.
• \texttt{window\_fast} (\texttt{int}) – n period short-term.
• \texttt{window\_sign} (\texttt{int}) – n period to signal.
• \texttt{fillna} (\texttt{bool}) – if True, fill nan values.

\textbf{Returns} New feature generated.

\textbf{Return type} \texttt{pandas.Series}

\texttt{ta.momentum.pvo\_signal} (\texttt{volume: pandas.core.series.Series, window\_slow: int = 26, window\_fast: int = 12, window\_sign: int = 9, fillna: bool = False}) \to \texttt{pandas.core.series.Series}

The Percentage Volume Oscillator (PVO) is a momentum oscillator for volume. The PVO measures the difference between two volume-based moving averages as a percentage of the larger moving average.


\textbf{Parameters}

• \texttt{volume} (\texttt{pandas.Series}) – dataset ‘Volume’ column.
• \texttt{window\_slow} (\texttt{int}) – n period long-term.
• \texttt{window\_fast} (\texttt{int}) – n period short-term.
• \texttt{window\_sign} (\texttt{int}) – n period to signal.
• \texttt{fillna} (\texttt{bool}) – if True, fill nan values.

\textbf{Returns} New feature generated.

\textbf{Return type} \texttt{pandas.Series}

\texttt{ta.momentum.roc} (\texttt{close: pandas.core.series.Series, window: int = 12, fillna: bool = False}) \to \texttt{pandas.core.series.Series}

Rate of Change (ROC)

The Rate-of-Change (ROC) indicator, which is also referred to as simply Momentum, is a pure momentum oscillator that measures the percent change in price from one period to the next. The ROC calculation compares the current price with the price “n” periods ago. The plot forms an oscillator that fluctuates above and below the zero line as the Rate-of-Change moves from positive to negative. As a momentum oscillator, ROC signals include centerline crossovers, divergences and overbought-oversold readings. Divergences fail to foreshadow reversals more often than not, so this article will forgo a detailed discussion on them. Even though centerline crossovers are prone to whipsaw, especially short-term, these crossovers can be used to identify the overall trend. Identifying overbought or oversold extremes comes naturally to the Rate-of-Change oscillator.

https://school.stockcharts.com/doku.php?id=technical\_indicators:rate\_of\_change\_roc\_and\_momentum

\textbf{Parameters}

• \texttt{close} (\texttt{pandas.Series}) – dataset ‘Close’ column.
• \texttt{window} (\texttt{int}) – n periods.
• \texttt{fillna} (\texttt{bool}) – if True, fill nan values.

\textbf{Returns} New feature generated.

\textbf{Return type} \texttt{pandas.Series}

\texttt{ta.momentum.rsi} (\texttt{close, window=14, fillna=False}) \to \texttt{pandas.core.series.Series}

Relative Strength Index (RSI)

Compares the magnitude of recent gains and losses over a specified time period to measure speed and change of price movements of a security. It is primarily used to attempt to identify overbought or oversold conditions in the trading of an asset.
https://www.investopedia.com/terms/r/rsi.asp

**Parameters**
- window (*int*) – n period.
- fillna (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** *pandas.Series*

```python
ta.momentum.stoch(high, low, close, window=14, smooth_window=3, fillna=False) → pandas.core.series.Series
```

Stochastic Oscillator

Developed in the late 1950s by George Lane. The stochastic oscillator presents the location of the closing price of a stock in relation to the high and low range of the price of a stock over a period of time, typically a 14-day period.

https://www.investopedia.com/terms/s/stochasticoscillator.asp

**Parameters**
- window (*int*) – n period.
- smooth_window (*int*) – sma period over stoch_k
- fillna (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** *pandas.Series*

```python
ta.momentum.stoch_signal(high, low, close, window=14, smooth_window=3, fillna=False) → pandas.core.series.Series
```

Stochastic Oscillator Signal

Shows SMA of Stochastic Oscillator. Typically a 3 day SMA.

https://www.investopedia.com/terms/s/stochasticoscillator.asp

**Parameters**
- window (*int*) – n period.
- smooth_window (*int*) – sma period over stoch_k
- fillna (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** *pandas.Series*
Stochastic RSI

The StochRSI oscillator was developed to take advantage of both momentum indicators in order to create a more sensitive indicator that is attuned to a specific security’s historical performance rather than a generalized analysis of price change.

https://www.investopedia.com/terms/s/stochrsi.asp

**Parameters**

- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period
- smooth1 (int) – moving average of Stochastic RSI
- smooth2 (int) – moving average of %K
- fillna (bool) – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  pandas.Series

Stochastic RSI %d

Stochastic RSI %k

The StochRSI oscillator was developed to take advantage of both momentum indicators in order to create a more sensitive indicator that is attuned to a specific security’s historical performance rather than a generalized analysis of price change.

https://www.investopedia.com/terms/s/stochrsi.asp

**Parameters**

- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period
- smooth1 (int) – moving average of Stochastic RSI
- smooth2 (int) – moving average of %K
- fillna (bool) – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  pandas.Series
• **smooth1** *(int)* – moving average of Stochastic RSI
• **smooth2** *(int)* – moving average of %K
• **fillna** *(bool)* – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.momentum.tsi(close, window_slow=25, window_fast=13, fillna=False) → pandas.core.series.Series
```
True strength index (TSI)

Shows both trend direction and overbought/oversold conditions.

https://en.wikipedia.org/wiki/True_strength_index

**Parameters**

• **close** *(pandas.Series)* – dataset ‘Close’ column.
• **window_slow** *(int)* – high period.
• **window_fast** *(int)* – low period.
• **fillna** *(bool)* – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.momentum.ultimate_oscillator(high, low, close, window1=7, window2=14, window3=28, weight1=4.0, weight2=2.0, weight3=1.0, fillna=False) → pandas.core.series.Series
```
Ultimate Oscillator

Larry Williams’ (1976) signal, a momentum oscillator designed to capture momentum across three different timeframes.


BP = Close - Minimum(Low or Prior Close). TR = Maximum(High or Prior Close) - Minimum(Low or Prior Close) Average7 = (7-period BP Sum) / (7-period TR Sum) Average14 = (14-period BP Sum) / (14-period TR Sum) Average28 = (28-period BP Sum) / (28-period TR Sum)

UO = 100 x [(4 x Average7)+(2 x Average14)+Average28]/(4+2+1)

**Parameters**

• **high** *(pandas.Series)* – dataset ‘High’ column.
• **low** *(pandas.Series)* – dataset ‘Low’ column.
• **close** *(pandas.Series)* – dataset ‘Close’ column.
• **window1** *(int)* – short period.
• **window2** *(int)* – medium period.
• **window3** *(int)* – long period.
• **weight1** *(float)* – weight of short BP average for UO.
• **weight2** *(float)* – weight of medium BP average for UO.
• **weight3** *(float)* – weight of long BP average for UO.
• **fillna** *(bool)* – if True, fill nan values with 50.

**Returns** New feature generated.
**Return type**  pandas.Series

ta.momentum.williams_r(high, low, close, lbp=14, fillna=False) → pandas.core.series.Series

Williams %R


Developed by Larry Williams, Williams %R is a momentum indicator that is the inverse of the Fast Stochastic Oscillator. Also referred to as %R, Williams %R reflects the level of the close relative to the highest high for the look-back period. In contrast, the Stochastic Oscillator reflects the level of the close relative to the lowest low. %R corrects for the inversion by multiplying the raw value by -100. As a result, the Fast Stochastic Oscillator and Williams %R produce the exact same lines, only the scaling is different. Williams %R oscillates from 0 to -100.

Readings from 0 to -20 are considered overbought. Readings from -80 to -100 are considered oversold.

Unsurprisingly, signals derived from the Stochastic Oscillator are also applicable to Williams %R.

%R = (Highest High - Close)/(Highest High - Lowest Low) * -100

Lowest Low = lowest low for the look-back period
Highest High = highest high for the look-back period
%R is multiplied by -100 correct the inversion and move the decimal.

From: https://www.investopedia.com/terms/w/williamsr.asp The Williams %R oscillates from 0 to -100. When the indicator produces readings from 0 to -20, this indicates overbought market conditions. When readings are -80 to -100, it indicates oversold market conditions.

**Parameters**

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
- lbp (int) – lookback period.
- fillna (bool) – if True, fill nan values with -50.

**Returns**  New feature generated.

**Return type**  pandas.Series

4.1.2 Volume Indicators

Volume Indicators.


Accumulation/Distribution Index (ADI)

Acting as leading indicator of price movements.

https://school.stockcharts.com/doku.php?id=technical_indicators:accumulation_distribution_line

**Parameters**

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
Technical Analysis Library in Python Documentation, Release 0.1.4

- **fillna** (*bool*) – if True, fill nan values.

acc_dist_index() → *pandas.core.series.Series*
Accumulation/Distribution Index (ADI)

**Returns** New feature generated.

**Return type** *pandas.Series*

```python
class ta.volume.ChaikinMoneyFlowIndicator:
    def __init__(self, high: *pandas.core.series.Series*, low: *pandas.core.series.Series*,
                 close: *pandas.core.series.Series*,
                 volume: *pandas.core.series.Series*,
                 window: int = 20,
                 fillna: bool = False):
        self.returns = chaikin_money_flow()
```

Chaikin Money Flow (CMF)

It measures the amount of Money Flow Volume over a specific period.


**Parameters**
- **high** (*pandas.Series*) – dataset ‘High’ column.
- **close** (*pandas.Series*) – dataset ‘Close’ column.
- **window** (*int*) – n period.
- **fillna** (*bool*) – if True, fill nan values.

```python
chaikin_money_flow() → *pandas.core.series.Series*
```

Chaikin Money Flow (CMF)

**Returns** New feature generated.

**Return type** *pandas.Series*

```python
class ta.volume.EaseOfMovementIndicator:
    def __init__(self, high: *pandas.core.series.Series*,
                 low: *pandas.core.series.Series*,
                 volume: *pandas.core.series.Series*,
                 window: int = 14,
                 fillna: bool = False):
        self.returns = ease_of_movement()
```

Ease of movement (EoM, EMV)

It relates an asset’s price change to its volume and is particularly useful for assessing the strength of a trend.


**Parameters**
- **high** (*pandas.Series*) – dataset ‘High’ column.
- **window** (*int*) – n period.
- **fillna** (*bool*) – if True, fill nan values.

```python
ease_of_movement() → *pandas.core.series.Series*
```

Ease of movement (EoM, EMV)
**sma_ease_of_movement** () → pandas.core.series.Series

Signal Ease of movement (EoM, EMV)

**Returns** New feature generated.

**Return type** pandas.Series


Force Index (FI)

It illustrates how strong the actual buying or selling pressure is. High positive values mean there is a strong rising trend, and low values signify a strong downward trend.


**Parameters**

- **close** (pandas.Series) – dataset ‘Close’ column.
- **volume** (pandas.Series) – dataset ‘Volume’ column.
- **window** (int) – n period.
- **fillna** (bool) – if True, fill nan values.

**force_index** () → pandas.core.series.Series

Force Index (FI)

**Returns** New feature generated.

**Return type** pandas.Series


Money Flow Index (MFI)

Uses both price and volume to measure buying and selling pressure. It is positive when the typical price rises (buying pressure) and negative when the typical price declines (selling pressure). A ratio of positive and negative money flow is then plugged into an RSI formula to create an oscillator that moves between zero and one hundred.


**Parameters**

- **high** (pandas.Series) – dataset ‘High’ column.
- **low** (pandas.Series) – dataset ‘Low’ column.
- **close** (pandas.Series) – dataset ‘Close’ column.
- **volume** (pandas.Series) – dataset ‘Volume’ column.
- **window** (int) – n period.
- **fillna** (bool) – if True, fill nan values.

**money_flow_index** () → pandas.core.series.Series

Money Flow Index (MFI)

**Returns** New feature generated.

Negative Volume Index (NVI)

Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- fillna (bool) – if True, fill nan values with 1000.

negative_volume_index() → pandas.core.series.Series

Negative Volume Index (NVI)

Returns New feature generated.

Return type pandas.Series


On-balance volume (OBV)

It relates price and volume in the stock market. OBV is based on a cumulative total volume.


Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- fillna (bool) – if True, fill nan values.

on_balance_volume() → pandas.core.series.Series

On-balance volume (OBV)

Returns New feature generated.

Return type pandas.Series


Volume-price trend (VPT)

Is based on a running cumulative volume that adds or subtracts a multiple of the percentage change in share price trend and current volume, depending upon the investment’s upward or downward movements.

https://en.wikipedia.org/wiki/Volume%E2%80%93price_trend

Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- fillna (bool) – if True, fill nan values.

volume_price_trend() → pandas.core.series.Series

Volume-price trend (VPT)

Returns New feature generated.
class ta.volume.VolumeWeightedAveragePrice:

Volume Weighted Average Price (VWAP)

VWAP equals the dollar value of all trading periods divided by the total trading volume for the current day. The calculation starts when trading opens and ends when it closes. Because it is good for the current trading day only, intraday periods and data are used in the calculation.


Parameters

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- fillna (bool) – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series

volume_weighted_average_price() → pandas.core.series.Series
 Volume Weighted Average Price (VWAP)

Returns  New feature generated.

Return type  pandas.Series

ta.volume.acc_dist_index (high, low, close, volume, fillna=False)
 Accumulation/Distribution Index (ADI)

Acting as leading indicator of price movements.

https://en.wikipedia.org/wiki/Accumulation/distribution_index

Parameters

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
- fillna (bool) – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series

ta.volume.chaikin_money_flow (high, low, close, volume, window=20, fillna=False)
 Chaikin Money Flow (CMF)

It measures the amount of Money Flow Volume over a specific period.

Parameters

- `window` *(int)* – n period.
- `fillna` *(bool)* – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

```python
ta.volume.ease_of_movement(high, low, volume, window=14, fillna=False)
```

Ease of movement (EoM, EMV)

It relate an asset’s price change to its volume and is particularly useful for assessing the strength of a trend.


Parameters

- `window` *(int)* – n period.
- `fillna` *(bool)* – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

```python
ta.volume.force_index(close, volume, window=13, fillna=False)
```

Force Index (FI)

It illustrates how strong the actual buying or selling pressure is. High positive values mean there is a strong rising trend, and low values signify a strong downward trend.


Parameters

- `window` *(int)* – n period.
- `fillna` *(bool)* – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

```python
ta.volume.money_flow_index(high, low, close, volume, window=14, fillna=False)
```

Money Flow Index (MFI)

Uses both price and volume to measure buying and selling pressure. It is positive when the typical price rises (buying pressure) and negative when the typical price declines (selling pressure). A ratio of positive and negative money flow is then plugged into an RSI formula to create an oscillator that moves between zero and one hundred.
Technical Analysis Library in Python Documentation, Release 0.1.4


Parameters

• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• close (pandas.Series) – dataset ‘Close’ column.
• volume (pandas.Series) – dataset ‘Volume’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

ta.volume.negative_volume_index(close, volume, fillna=False)

Negative Volume Index (NVI)


The Negative Volume Index (NVI) is a cumulative indicator that uses the change in volume to decide when the smart money is active. Paul Dysart first developed this indicator in the 1930s. [...] Dysart’s Negative Volume Index works under the assumption that the smart money is active on days when volume decreases and the not-so-smart money is active on days when volume increases.

The cumulative NVI line was unchanged when volume increased from one period to the other. In other words, nothing was done. Norman Fosback, of Stock Market Logic, adjusted the indicator by substituting the percentage price change for Net Advances.

This implementation is the Fosback version.

If today’s volume is less than yesterday’s volume then: nvi(t) = nvi(t-1) * ( 1 + (close(t) - close(t-1)) / close(t-1) )

Else nvi(t) = nvi(t-1)

Please note: the “stockcharts.com” example calculation just adds the percentage change of price to previous NVI when volumes decline; other sources indicate that the same percentage of the previous NVI value should be added, which is what is implemented here.

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• volume (pandas.Series) – dataset ‘Volume’ column.
• fillna (bool) – if True, fill nan values with 1000.

Returns New feature generated.

Return type pandas.Series

See also:

https://en.wikipedia.org/wiki/Negative_volume_index

ta.volume.on_balance_volume(close, volume, fillna=False)

On-balance volume (OBV)

It relates price and volume in the stock market. OBV is based on a cumulative total volume.

Parameters

- `close (pandas.Series) – dataset ‘Close’ column.
- `fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

ta.volume.sma_ease_of_movement (high, low, volume, window=14, fillna=False)

Ease of movement (EoM, EMV)

It relate an asset’s price change to its volume and is particularly useful for assessing the strength of a trend.


Parameters

- `high (pandas.Series) – dataset ‘High’ column.
- `low (pandas.Series) – dataset ‘Low’ column.
- `window (int) – n period.
- `fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

ta.volume.volume_price_trend (close, volume, fillna=False)

Volume-price trend (VPT)

Is based on a running cumulative volume that adds or substracts a multiple of the percentage change in share price trend and current volume, depending upon the investment’s upward or downward movements.

https://en.wikipedia.org/wiki/Volume%E2%80%93price_trend

Parameters

- `close (pandas.Series) – dataset ‘Close’ column.
- `fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series


Volume Weighted Average Price (VWAP)

VWAP equals the dollar value of all trading periods divided by the total trading volume for the current day. The calculation starts when trading opens and ends when it closes. Because it is good for the current trading day only, intraday periods and data are used in the calculation.


Parameters
• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• close (pandas.Series) – dataset ‘Close’ column.
• volume (pandas.Series) – dataset ‘Volume’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

Returns New feature generated.
Return type pandas.Series

4.1.3 Volatility Indicators

Volatility Indicators.


Average True Range (ATR)
The indicator provide an indication of the degree of price volatility. Strong moves, in either direction, are often accompanied by large ranges, or large True Ranges.


Parameters
• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

average_true_range () → pandas.core.series.Series
Average True Range (ATR)

Returns New feature generated.
Return type pandas.Series


Bollinger Bands


Parameters
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• window_dev (int) – n factor standard deviation
• fillna (bool) – if True, fill nan values.

bollinger_hband () → pandas.core.series.Series
Bollinger Channel High Band
Returns New feature generated.

Return type pandas.Series

```python
def bollinger_hband_indicator() -> pandas.core.series.Series
Bollinger Channel Indicator Crossing High Band (binary).
It returns 1, if close is higher than bollinger_hband. Else, it returns 0.
```

Returns New feature generated.

Return type pandas.Series

```python
def bollinger_lband() -> pandas.core.series.Series
Bollinger Channel Low Band
```

Returns New feature generated.

Return type pandas.Series

```python
def bollinger_lband_indicator() -> pandas.core.series.Series
Bollinger Channel Indicator Crossing Low Band (binary).
It returns 1, if close is lower than bollinger_lband. Else, it returns 0.
```

Returns New feature generated.

Return type pandas.Series

```python
def bollinger_mavg() -> pandas.core.series.Series
Bollinger Channel Middle Band
```

Returns New feature generated.

Return type pandas.Series

```python
def bollinger_pband() -> pandas.core.series.Series
Bollinger Channel Percentage Band
```

From: https://school.stockcharts.com/doku.php?id=technical_indicators:bollinger_band_perce

Returns New feature generated.

Return type pandas.Series

```python
def bollinger_wband() -> pandas.core.series.Series
Bollinger Channel Band Width
```

From: https://school.stockcharts.com/doku.php?id=technical_indicators:bollinger_band_width

Returns New feature generated.

Return type pandas.Series

```python

Donchian Channel
```

https://www.investopedia.com/terms/d/donchianchannels.asp

Parameters

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
• **window** (*int*) – n period.

• **fillna** (*bool*) – if True, fill nan values.

```python
donchian_channel_hband() → pandas.core.series.Series
```
Donchian Channel High Band

**Returns** New feature generated.

**Return type** pandas.Series

```python
donchian_channel_lband() → pandas.core.series.Series
```
Donchian Channel Low Band

**Returns** New feature generated.

**Return type** pandas.Series

```python
donchian_channel_mband() → pandas.core.series.Series
```
Donchian Channel Middle Band

**Returns** New feature generated.

**Return type** pandas.Series

```python
donchian_channel_pband() → pandas.core.series.Series
```
Donchian Channel Percentage Band

**Returns** New feature generated.

**Return type** pandas.Series

```python
donchian_channel_wband() → pandas.core.series.Series
```
Donchian Channel Band Width

**Returns** New feature generated.

**Return type** pandas.Series

```python
```
Keltner Channels are a trend following indicator used to identify reversals with channel breakouts and channel direction. Channels can also be used to identify overbought and oversold levels when the trend is flat.


**Parameters**

• **high** (*pandas.Series*) – dataset ‘High’ column.

• **low** (*pandas.Series*) – dataset ‘Low’ column.

• **close** (*pandas.Series*) – dataset ‘Close’ column.

• **window** (*int*) – n period.

• **window_atr** (*int*) – n atr period. Only valid if original_version param is False.

• **fillna** (*bool*) – if True, fill nan values.

• **original_version** (*bool*) – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

```python
keltner_channel_hband() → pandas.core.series.Series
```
Keltner Channel High Band
**keltner_channel_hband_indicator()** → pandas.core.series.Series
Keltner Channel Indicator Crossing High Band (binary)
It returns 1, if close is higher than keltner_channel_hband. Else, it returns 0.

**Returns**  New feature generated.

**Return type**  pandas.Series

**keltner_channel_lband()** → pandas.core.series.Series
Keltner Channel Low Band

**Returns**  New feature generated.

**Return type**  pandas.Series

**keltner_channel_lband_indicator()** → pandas.core.series.Series
Keltner Channel Indicator Crossing Low Band (binary)
It returns 1, if close is lower than keltner_channel_lband. Else, it returns 0.

**Returns**  New feature generated.

**Return type**  pandas.Series

**keltner_channel_mband()** → pandas.core.series.Series
Keltner Channel Middle Band

**Returns**  New feature generated.

**Return type**  pandas.Series

**keltner_channel_pband()** → pandas.core.series.Series
Keltner Channel Percentage Band

**Returns**  New feature generated.

**Return type**  pandas.Series

**keltner_channel_wband()** → pandas.core.series.Series
Keltner Channel Band Width

**Returns**  New feature generated.

**Return type**  pandas.Series

**class ta.volatility.UlcerIndex(close: pandas.core.series.Series, window: int = 14, fillna: bool = False)**

Ulcer Index


**Parameters**

- **close** (pandas.Series) – dataset ‘Close’ column.
- **window** (int) – n period.
- **fillna** (bool) – if True, fill nan values.

**ulcer_index()** → pandas.core.series.Series
Ulcer Index (UI)

**Returns**  New feature generated.
Return type  pandas.Series
ta.volatility.average_true_range(high, low, close, window=14, fillna=False)
Average True Range (ATR)
The indicator provides an indication of the degree of price volatility. Strong moves, in either direction, are often
accompanied by large ranges, or large True Ranges.

Parameters
  • high (pandas.Series) – dataset ‘High’ column.
  • low (pandas.Series) – dataset ‘Low’ column.
  • close (pandas.Series) – dataset ‘Close’ column.
  • window (int) – n period.
  • fillna (bool) – if True, fill nan values.
Returns  New feature generated.
Return type  pandas.Series
ta.volatility.bollinger_hband(close, window=20, window_dev=2, fillna=False)
Bollinger Bands (BB)
Upper band at K times an N-period standard deviation above the moving average (MA + Kdeviation).
https://en.wikipedia.org/wiki/Bollinger_Bands

Parameters
  • close (pandas.Series) – dataset ‘Close’ column.
  • window (int) – n period.
  • window_dev (int) – n factor standard deviation
  • fillna (bool) – if True, fill nan values.
Returns  New feature generated.
Return type  pandas.Series
ta.volatility.bollinger_hband_indicator(close, window=20, window_dev=2, fillna=False)
Bollinger High Band Indicator
Returns 1, if close is higher than bollinger high band. Else, return 0.
https://en.wikipedia.org/wiki/Bollinger_Bands

Parameters
  • close (pandas.Series) – dataset ‘Close’ column.
  • window (int) – n period.
  • window_dev (int) – n factor standard deviation
  • fillna (bool) – if True, fill nan values.
Returns  New feature generated.
Return type  pandas.Series
ta.volatility.bollinger_lband(close, window=20, window_dev=2, fillna=False)
Bollinger Bands (BB)
Lower band at K times an N-period standard deviation below the moving average (MA  Kdeviation).
https://en.wikipedia.org/wiki/Bollinger_Bands

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• window_dev (int) – n factor standard deviation
• fillna (bool) – if True, fill nan values.

Returns  New feature generated.
Return type  pandas.Series

ta.volatility.bollinger_lband_indicator(close, window=20, window_dev=2, fillna=False)
Bollinger Low Band Indicator
Returns 1, if close is lower than bollinger low band. Else, return 0.
https://en.wikipedia.org/wiki/Bollinger_Bands

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• window_dev (int) – n factor standard deviation
• fillna (bool) – if True, fill nan values.

Returns  New feature generated.
Return type  pandas.Series

ta.volatility.bollinger_mavg(close, window=20, fillna=False)
Bollinger Bands (BB)
N-period simple moving average (MA).
https://en.wikipedia.org/wiki/Bollinger_Bands

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

Returns  New feature generated.
Return type  pandas.Series

ta.volatility.bollinger_pband(close, window=20, window_dev=2, fillna=False)
Bollinger Channel Percentage Band
From: https://school.stockcharts.com/doku.php?id=technical_indicators:bollinger_band_perce

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• *window* (*int*) – n period.
• *window_dev* (*int*) – n factor standard deviation
• *fillna* (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.volatility.bollinger_wband(close, window=20, window_dev=2, fillna=False)
```
Bollinger Channel Band Width

From: https://school.stockcharts.com/doku.php?id=technical_indicators:bollinger_band_width

**Parameters**

• *close* (*pandas.Series*) – dataset ‘Close’ column.
• *window* (*int*) – n period.
• *window_dev* (*int*) – n factor standard deviation
• *fillna* (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.volatility.donchian_channel_hband(high, low, close, window=20, offset=0, fillna=False)
```
Donchian Channel High Band (DC)

The upper band marks the highest price of an issue for n periods.

https://www.investopedia.com/terms/d/donchianchannels.asp

**Parameters**

• *high* (*pandas.Series*) – dataset ‘High’ column.
• *low* (*pandas.Series*) – dataset ‘Low’ column.
• *close* (*pandas.Series*) – dataset ‘Close’ column.
• *window* (*int*) – n period.
• *fillna* (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.volatility.donchian_channel_lband(high, low, close, window=20, offset=0, fillna=False)
```
Donchian Channel Low Band (DC)

The lower band marks the lowest price for n periods.

https://www.investopedia.com/terms/d/donchianchannels.asp

**Parameters**

• *high* (*pandas.Series*) – dataset ‘High’ column.
• *low* (*pandas.Series*) – dataset ‘Low’ column.
• *close* (*pandas.Series*) – dataset ‘Close’ column.
• *window* (*int*) – n period.
• *fillna* (*bool*) – if True, fill nan values.
Returns  New feature generated.

Return type  pandas.Series

```
import ta

ta.volatility.donchian_channel_mband(high, low, close, window=10, offset=0, fillna=False)
```

Donchian Channel Middle Band (DC)

https://www.investopedia.com/terms/d/donchianchannels.asp

Parameters

- **high** (pandas.Series) – dataset ‘High’ column.
- **low** (pandas.Series) – dataset ‘Low’ column.
- **close** (pandas.Series) – dataset ‘Close’ column.
- **window** (int) – n period.
- **fillna** (bool) – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series

```
import ta

ta.volatility.donchian_channel_pband(high, low, close, window=10, offset=0, fillna=False)
```

Donchian Channel Percentage Band (DC)

https://www.investopedia.com/terms/d/donchianchannels.asp

Parameters

- **high** (pandas.Series) – dataset ‘High’ column.
- **low** (pandas.Series) – dataset ‘Low’ column.
- **close** (pandas.Series) – dataset ‘Close’ column.
- **window** (int) – n period.
- **fillna** (bool) – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series

```
import ta

ta.volatility.donchian_channel_wband(high, low, close, window=10, offset=0, fillna=False)
```

Donchian Channel Band Width (DC)

https://www.investopedia.com/terms/d/donchianchannels.asp

Parameters

- **high** (pandas.Series) – dataset ‘High’ column.
- **low** (pandas.Series) – dataset ‘Low’ column.
- **close** (pandas.Series) – dataset ‘Close’ column.
- **window** (int) – n period.
- **fillna** (bool) – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series
`ta.volatility.keltner_channel_hband(high, low, close, window=20, window_atr=10, fillna=False, original_version=True)`

Keltner channel (KC)
Showing a simple moving average line (high) of typical price.

**Parameters**
- `window` *(int)* – n period.
- `window_atr` *(int)* – n atr period. Only valid if original_version param is False.
- `fillna` *(bool)* – if True, fill nan values.
- `original_version` *(bool)* – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

**Returns** New feature generated.

**Return type** pandas.Series

`ta.volatility.keltner_channel_hband_indicator(high, low, close, window=20, window_atr=10, fillna=False, original_version=True)`

Keltner Channel High Band Indicator (KC)
Returns 1, if close is higher than keltner high band channel. Else, return 0.

**Parameters**
- `window` *(int)* – n period.
- `window_atr` *(int)* – n atr period. Only valid if original_version param is False.
- `fillna` *(bool)* – if True, fill nan values.
- `original_version` *(bool)* – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

**Returns** New feature generated.

**Return type** pandas.Series

`ta.volatility.keltner_channel_lband(high, low, close, window=20, window_atr=10, fillna=False, original_version=True)`

Keltner channel (KC)
Showing a simple moving average line (low) of typical price.
Parameters

- `high (pandas.Series)` – dataset ‘High’ column.
- `low (pandas.Series)` – dataset ‘Low’ column.
- `close (pandas.Series)` – dataset ‘Close’ column.
- `window (int)` – n period.
- `window_atr (int)` – n atr period. Only valid if original_version param is False.
- `fillna (bool)` – if True, fill nan values.
- `original_version (bool)` – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

Returns New feature generated.

Return type pandas.Series

```python
ta.volatility.keltner_channel_lband_indicator(high, low, close, window=20, window_atr=10, fillna=False, original_version=True)
```

Keltner Channel Low Band Indicator (KC)

Returns 1, if close is lower than keltner low band channel. Else, return 0.


Parameters

- `high (pandas.Series)` – dataset ‘High’ column.
- `low (pandas.Series)` – dataset ‘Low’ column.
- `close (pandas.Series)` – dataset ‘Close’ column.
- `window (int)` – n period.
- `window_atr (int)` – n atr period. Only valid if original_version param is False.
- `fillna (bool)` – if True, fill nan values.
- `original_version (bool)` – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

Returns New feature generated.

Return type pandas.Series

```python
ta.volatility.keltner_channel_mband(high, low, close, window=20, window_atr=10, fillna=False, original_version=True)
```

Keltner channel (KC)

Showing a simple moving average line (central) of typical price.


Parameters

- `high (pandas.Series)` – dataset ‘High’ column.
- `low (pandas.Series)` – dataset ‘Low’ column.
- `close (pandas.Series)` – dataset ‘Close’ column.
• `window` (*int*) – n period.

• `window_atr` (*int*) – n atr period. Only valid if `original_version` param is False.

• `fillna` (*bool*) – if True, fill nan values.

• `original_version` (*bool*) – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.volatility.keltner_channel_pband(high, low, close, window=20, window_atr=10,
                fillna=False, original_version=True)
```

Keltner Channel Percentage Band


**Parameters**

• `high` (*pandas.Series*) – dataset ‘High’ column.

• `low` (*pandas.Series*) – dataset ‘Low’ column.

• `close` (*pandas.Series*) – dataset ‘Close’ column.

• `window` (*int*) – n period.

• `window_atr` (*int*) – n atr period. Only valid if `original_version` param is False.

• `fillna` (*bool*) – if True, fill nan values.

• `original_version` (*bool*) – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.volatility.keltner_channel_wband(high, low, close, window=20, window_atr=10,
                fillna=False, original_version=True)
```

Keltner Channel Band Width


**Parameters**

• `high` (*pandas.Series*) – dataset ‘High’ column.

• `low` (*pandas.Series*) – dataset ‘Low’ column.

• `close` (*pandas.Series*) – dataset ‘Close’ column.

• `window` (*int*) – n period.

• `window_atr` (*int*) – n atr period. Only valid if `original_version` param is False.

• `fillna` (*bool*) – if True, fill nan values.

• `original_version` (*bool*) – if True, use original version as the centerline (SMA of typical price) if False, use EMA of close as the centerline. More info: https://school.stockcharts.com/doku.php?id=technical_indicators:keltner_channels

**Returns** New feature generated.

**Return type** pandas.Series
ta.volatility.ulcer_index(close, window=14, fillna=False)

Ulcer Index


Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

4.1.4 Trend Indicators

Trend Indicators.


Average Directional Movement Index (ADX)

The Plus Directional Indicator (+DI) and Minus Directional Indicator (-DI) are derived from smoothed averages of these differences, and measure trend direction over time. These two indicators are often referred to collectively as the Directional Movement Indicator (DMI).

The Average Directional Index (ADX) is in turn derived from the smoothed averages of the difference between +DI and -DI, and measures the strength of the trend (regardless of direction) over time.

Using these three indicators together, chartists can determine both the direction and strength of the trend.


Parameters

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- fillna (bool) – if True, fill nan values.

adx() → pandas.core.series.Series

Average Directional Index (ADX)

Returns New feature generated.

Return type pandas.Series

adx_neg() → pandas.core.series.Series

Minus Directional Indicator (-DI)

Returns New feature generated.

Return type pandas.Series

adx_pos() → pandas.core.series.Series

Plus Directional Indicator (+DI)
Returns  New feature generated.
Return type  pandas.Series

class ta.trend.AroonIndicator(close: pandas.core.series.Series, window: int = 25, fillna: bool = False)
Aroon Indicator
Identify when trends are likely to change direction.
Aroon Up = ((N - Days Since N-day High) / N) x 100  Aroon Down = ((N - Days Since N-day Low) / N) x 100
Aroon Indicator = Aroon Up - Aroon Down
https://www.investopedia.com/terms/a/aroon.asp

Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- fillna (bool) – if True, fill nan values.

aroon_down () → pandas.core.series.Series
Aroon Down Channel
Returns  New feature generated.
Return type  pandas.Series

aroon_indicator () → pandas.core.series.Series
Aroon Indicator
Returns  New feature generated.
Return type  pandas.Series

aroon_up () → pandas.core.series.Series
Aroon Up Channel
Returns  New feature generated.
Return type  pandas.Series

class ta.trend.CCIIndicator(high: pandas.core.series.Series, low: pandas.core.series.Series, close: pandas.core.series.Series, window: int = 20, constant: float = 0.015, fillna: bool = False)
Commodity Channel Index (CCI)
CCI measures the difference between a security’s price change and its average price change. High positive readings indicate that prices are well above their average, which is a show of strength. Low negative readings indicate that prices are well below their average, which is a show of weakness.

Parameters

- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- constant (int) – constant.
- fillna (bool) – if True, fill nan values.
```python
ci() → pandas.core.series.Series
Commodity Channel Index (CCI)

Returns New feature generated.

Return type pandas.Series
class ta.trend.DPOIndicator(close: pandas.core.series.Series, window: int = 20, fillna: bool = False)
Detrended Price Oscillator (DPO)
Is an indicator designed to remove trend from price and make it easier to identify cycles.

Parameters
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

dpo() → pandas.core.series.Series
Detrended Price Oscillator (DPO)

Returns New feature generated.

Return type pandas.Series
class ta.trend.EMAIndicator(close: pandas.core.series.Series, window: int = 14, fillna: bool = False)
EMA - Exponential Moving Average

Parameters
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

ema_indicator() → pandas.core.series.Series
Exponential Moving Average (EMA)

Returns New feature generated.

Return type pandas.Series
Ichimoku Kinkō Hyō (Ichimoku)

Parameters
• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• window1 (int) – n1 low period.
• window2 (int) – n2 medium period.
• window3 (int) – n3 high period.
```

4.1. Documentation
• **visual** *(bool)* – if True, shift n2 values.
• **fillna** *(bool)* – if True, fill nan values.

**ichimoku_a** () → pandas.core.series.Series
Senkou Span A (Leading Span A)

**Returns** New feature generated.
**Return type** pandas.Series

**ichimoku_b** () → pandas.core.series.Series
Senkou Span B (Leading Span B)

**Returns** New feature generated.
**Return type** pandas.Series

**ichimoku_base_line** () → pandas.core.series.Series
Kijun-sen (Base Line)

**Returns** New feature generated.
**Return type** pandas.Series

**ichimoku_conversion_line** () → pandas.core.series.Series
Tenkan-sen (Conversion Line)

**Returns** New feature generated.
**Return type** pandas.Series


KST Oscillator (KST Signal)

It is useful to identify major stock market cycle junctures because its formula is weighed to be more greatly influenced by the longer and more dominant time spans, in order to better reflect the primary swings of stock market cycle.


**Parameters**

• **close** *(pandas.Series)* – dataset ‘Close’ column.
• **roc1** *(int)* – roc1 period.
• **roc2** *(int)* – roc2 period.
• **roc3** *(int)* – roc3 period.
• **roc4** *(int)* – roc4 period.
• **window1** *(int)* – n1 smoothed period.
• **window2** *(int)* – n2 smoothed period.
• **window3** *(int)* – n3 smoothed period.
• **window4** *(int)* – n4 smoothed period.
• **nsig** *(int)* – n period to signal.
• **fillna** *(bool)* – if True, fill nan values.
**Technical Analysis Library in Python Documentation, Release 0.1.4**

```python
kst() \rightarrow \text{pandas.core.series.Series}

Know Sure Thing (KST)

\textbf{Returns} New feature generated.

\textbf{Return type} \text{pandas.Series}

\textit{kst\_diff()} \rightarrow \text{pandas.core.series.Series}

Diff Know Sure Thing (KST)

KST - Signal\_KST

\textbf{Returns} New feature generated.

\textbf{Return type} \text{pandas.Series}

\textit{kst\_sig()} \rightarrow \text{pandas.core.series.Series}

Signal Line Know Sure Thing (KST)

tsig-period SMA of KST

\textbf{Returns} New feature generated.

\textbf{Return type} \text{pandas.Series}

\textbf{class} ta.trend.MACD (close: pandas.core.series.Series, window\_slow: int = 26, window\_fast: int = 12,
window\_sign: int = 9, fillna: bool = False)

Moving Average Convergence Divergence (MACD)

Is a trend-following momentum indicator that shows the relationship between two moving averages of prices.


\textbf{Parameters}

- \textit{window\_fast} (int) – n period short-term.
- \textit{window\_slow} (int) – n period long-term.
- \textit{window\_sign} (int) – n period to signal.
- \textit{fillna} (bool) – if True, fill nan values.

\textit{macd()} \rightarrow \text{pandas.core.series.Series}

MACD Line

\textbf{Returns} New feature generated.

\textbf{Return type} \text{pandas.Series}

\textit{macd\_diff()} \rightarrow \text{pandas.core.series.Series}

MACD Histogram

\textbf{Returns} New feature generated.

\textbf{Return type} \text{pandas.Series}

\textit{macd\_signal()} \rightarrow \text{pandas.core.series.Series}

Signal Line

\textbf{Returns} New feature generated.

\textbf{Return type} \text{pandas.Series}
```
class ta.trend.MassIndex:
        Mass Index (MI)
        It uses the high-low range to identify trend reversals based on range expansions. It identifies range bulges that can foreshadow a reversal of the current trend.


Parameters

- **high (pandas.Series)** – dataset ‘High’ column.
- **low (pandas.Series)** – dataset ‘Low’ column.
- **window_fast (int)** – fast period value.
- **window_slow (int)** – slow period value.
- **fillna (bool)** – if True, fill nan values.

mass_index() → pandas.core.series.Series
Mass Index (MI)

Returns  New feature generated.
Return type  pandas.Series

class ta.trend.PSARIndicator:
        Parabolic Stop and Reverse (Parabolic SAR)
        The Parabolic Stop and Reverse, more commonly known as the Parabolic SAR, is a trend-following indicator developed by J. Welles Wilder. The Parabolic SAR is displayed as a single parabolic line (or dots) underneath the price bars in an uptrend, and above the price bars in a downtrend.


Parameters

- **high (pandas.Series)** – dataset ‘High’ column.
- **low (pandas.Series)** – dataset ‘Low’ column.
- **close (pandas.Series)** – dataset ‘Close’ column.
- **step (float)** – the Acceleration Factor used to compute the SAR.
- **max_step (float)** – the maximum value allowed for the Acceleration Factor.
- **fillna (bool)** – if True, fill nan values.

psar() → pandas.core.series.Series
PSAR value

Returns  New feature generated.
Return type  pandas.Series

psar_down() → pandas.core.series.Series
PSAR down trend value

Returns  New feature generated.
Return type  pandas.Series
**psar_down_indicator**() → pandas.core.series.Series
PSAR down trend value indicator

**Returns**  New feature generated.

**Return type**  pandas.Series

**psar_up**() → pandas.core.series.Series
PSAR up trend value

**Returns**  New feature generated.

**Return type**  pandas.Series

**psar_up_indicator**() → pandas.core.series.Series
PSAR up trend value indicator

**Returns**  New feature generated.

**Return type**  pandas.Series

```python
class ta.trend.SMAIndicator(close: pandas.core.series.Series, window: int, fillna: bool = False):
    SMA - Simple Moving Average

    Parameters
    • close (pandas.Series) – dataset ‘Close’ column.
    • window (int) – n period.
    • fillna (bool) – if True, fill nan values.
```

**sma_indicator**() → pandas.core.series.Series
Simple Moving Average (SMA)

**Returns**  New feature generated.

**Return type**  pandas.Series

```python
    Schaff Trend Cycle (STC)

    The Schaff Trend Cycle (STC) is a charting indicator that is commonly used to identify market trends and provide buy and sell signals to traders. Developed in 1999 by noted currency trader Doug Schaff, STC is a type of oscillator and is based on the assumption that, regardless of time frame, currency trends accelerate and decelerate in cyclical patterns.

    https://www.investopedia.com/articles/forex/10/schaff-trend-cycle-indicator.asp
```

**Parameters**

• close (pandas.Series) – dataset ‘Close’ column.

• window_fast (int) – n period short-term.

• window_slow (int) – n period long-term.

• cycle (int) – cycle size

• smooth1 (int) – ema period over stoch_k

• smooth2 (int) – ema period over stoch_kd

• fillna (bool) – if True, fill nan values.
Schaff Trend Cycle

Returns New feature generated.

Return type pandas.Series

class ta.trend.TRIXIndicator (close: pandas.core.series.Series, window: int = 15, fillna: bool = False)

Trix (TRIX)

Shows the percent rate of change of a triple exponentially smoothed moving average.


Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

trix () -> pandas.core.series.Series

Trix (TRIX)

Returns New feature generated.

Return type pandas.Series


Vortex Indicator (VI)

It consists of two oscillators that capture positive and negative trend movement. A bullish signal triggers when the positive trend indicator crosses above the negative trend indicator or a key level.


Parameters

• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

vortex_indicator_diff ()

Diff VI

Returns New feature generated.

Return type pandas.Series

vortex_indicator_neg ()

-VI

Returns New feature generated.

Return type pandas.Series

vortex_indicator_pos ()

+VI
Returns New feature generated.
Return type pandas.Series

class ta.trend.WMAIndicator (close: pandas.core.series.Series, window: int = 9, fillna: bool = False)
WMA - Weighted Moving Average

Parameters
- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- fillna (bool) – if True, fill nan values.

wma () → pandas.core.series.Series
Weighted Moving Average (WMA)

Returns New feature generated.
Return type pandas.Series

ta.trend.adx (high, low, close, window=14, fillna=False)
Average Directional Movement Index (ADX)
The Plus Directional Indicator (+DI) and Minus Directional Indicator (-DI) are derived from smoothed averages of these differences, and measure trend direction over time. These two indicators are often referred to collectively as the Directional Movement Indicator (DMI).
The Average Directional Index (ADX) is in turn derived from the smoothed averages of the difference between +DI and -DI, and measures the strength of the trend (regardless of direction) over time.
Using these three indicators together, chartists can determine both the direction and strength of the trend.

Parameters
- high (pandas.Series) – dataset ‘High’ column.
- low (pandas.Series) – dataset ‘Low’ column.
- close (pandas.Series) – dataset ‘Close’ column.
- window (int) – n period.
- fillna (bool) – if True, fill nan values.

Returns New feature generated.
Return type pandas.Series

ta.trend.adx_neg (high, low, close, window=14, fillna=False)
Average Directional Movement Index Negative (ADX)
The Plus Directional Indicator (+DI) and Minus Directional Indicator (-DI) are derived from smoothed averages of these differences, and measure trend direction over time. These two indicators are often referred to collectively as the Directional Movement Indicator (DMI).
The Average Directional Index (ADX) is in turn derived from the smoothed averages of the difference between +DI and -DI, and measures the strength of the trend (regardless of direction) over time.
Using these three indicators together, chartists can determine both the direction and strength of the trend.

Parameters
• **high** (*pandas.Series*) – dataset ‘High’ column.
• **low** (*pandas.Series*) – dataset ‘Low’ column.
• **close** (*pandas.Series*) – dataset ‘Close’ column.
• **window** (*int*) – n period.
• **fillna** (*bool*) – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  *pandas.Series*

```
```

```python
ta.trend.adx_pos(high, low, close, window=14, fillna=False)
```

*Average Directional Movement Index Positive (ADX)*

The Plus Directional Indicator (+DI) and Minus Directional Indicator (-DI) are derived from smoothed averages of these differences, and measure trend direction over time. These two indicators are often referred to collectively as the Directional Movement Indicator (DMI).

The Average Directional Index (ADX) is in turn derived from the smoothed averages of the difference between +DI and -DI, and measures the strength of the trend (regardless of direction) over time.

Using these three indicators together, chartists can determine both the direction and strength of the trend.


**Parameters**

• **high** (*pandas.Series*) – dataset ‘High’ column.
• **low** (*pandas.Series*) – dataset ‘Low’ column.
• **close** (*pandas.Series*) – dataset ‘Close’ column.
• **window** (*int*) – n period.
• **fillna** (*bool*) – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  *pandas.Series*

```
```

```python
ta.trend.aroon_down(close, window=25, fillna=False)
```

*Aroon Indicator (AI)*

Identify when trends are likely to change direction (downtrend).

Aroon Down - \( \frac{(N - \text{Days Since N-day Low})}{N} \times 100 \)

https://www.investopedia.com/terms/a/aroon.asp

**Parameters**

• **close** (*pandas.Series*) – dataset ‘Close’ column.
• **window** (*int*) – n period.
• **fillna** (*bool*) – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  *pandas.Series*

```
```

```python
ta.trend.aroon_up(close, window=25, fillna=False)
```

*Aroon Indicator (AI)*

Identify when trends are likely to change direction (uptrend).
Aroon Up - \(((N - \text{Days Since N-day High}) / N) \times 100\)

https://www.investopedia.com/terms/a/aroon.asp

**Parameters**
- `close` (`pandas.Series`) – dataset ‘Close’ column.
- `window` (`int`) – n period.
- `fillna` (`bool`) – if True, fill nan values.

**Returns** New feature generated.

**Return type** `pandas.Series`

**ta.trend.cci** `(high, low, close, window=20, constant=0.015, fillna=False)`
Commodity Channel Index (CCI)

CCI measures the difference between a security’s price change and its average price change. High positive readings indicate that prices are well above their average, which is a show of strength. Low negative readings indicate that prices are well below their average, which is a show of weakness.


**Parameters**
- `high` (`pandas.Series`) – dataset ‘High’ column.
- `low` (`pandas.Series`) – dataset ‘Low’ column.
- `close` (`pandas.Series`) – dataset ‘Close’ column.
- `window` (`int`) – n periods.
- `constant` (`int`) – constant.
- `fillna` (`bool`) – if True, fill nan values.

**Returns** New feature generated.

**Return type** `pandas.Series`

**ta.trend.dpo** `(close, window=20, fillna=False)`
Detrended Price Oscillator (DPO)

Is an indicator designed to remove trend from price and make it easier to identify cycles.


**Parameters**
- `close` (`pandas.Series`) – dataset ‘Close’ column.
- `window` (`int`) – n period.
- `fillna` (`bool`) – if True, fill nan values.

**Returns** New feature generated.

**Return type** `pandas.Series`

**ta.trend.ema_indicator** `(close, window=12, fillna=False)`
Exponential Moving Average (EMA)

**Returns** New feature generated.

**Return type** `pandas.Series`
It identifies the trend and look for potential signals within that trend.


### Parameters

- **high** ([pandas.Series](#)) – dataset ‘High’ column.
- **low** ([pandas.Series](#)) – dataset ‘Low’ column.
- **window1** ([int](#)) – n1 low period.
- **window2** ([int](#)) – n2 medium period.
- **visual** ([bool](#)) – if True, shift n2 values.
- **fillna** ([bool](#)) – if True, fill nan values.

### Returns

New feature generated.

**Return type**  pandas.Series

---

It identifies the trend and look for potential signals within that trend.


### Parameters

- **high** ([pandas.Series](#)) – dataset ‘High’ column.
- **low** ([pandas.Series](#)) – dataset ‘Low’ column.
- **window2** ([int](#)) – n2 medium period.
- **window3** ([int](#)) – n3 high period.
- **visual** ([bool](#)) – if True, shift n2 values.
- **fillna** ([bool](#)) – if True, fill nan values.

### Returns

New feature generated.

**Return type**  pandas.Series

---

It identifies the trend and look for potential signals within that trend.


### Parameters

- **high** ([pandas.Series](#)) – dataset ‘High’ column.
- **low** ([pandas.Series](#)) – dataset ‘Low’ column.
- **window1** ([int](#)) – n1 low period.
- **window2** ([int](#)) – n2 medium period.
- **visual** ([bool](#)) – if True, shift n2 values.
- **fillna** ([bool](#)) – if True, fill nan values.

### Returns

New feature generated.

**Return type**  pandas.core.series.Series
Returns New feature generated.

Return type pandas.Series
ta.trend.ichimoku_conversion_line(high, low, window1=9, window2=26, visual=False, fillna=False) → pandas.core.series.Series

Tenkan-sen (Conversion Line)

It identifies the trend and look for potential signals within that trend.

Parameters
• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• window1 (int) – n1 low period.
• window2 (int) – n2 medium period.
• visual (bool) – if True, shift n2 values.
• fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series
ta.trend.kst(close, roc1=10, roc2=15, roc3=20, roc4=30, window1=10, window2=10, window3=10, window4=15, fillna=False)

KST Oscillator (KST)

It is useful to identify major stock market cycle junctures because its formula is weighed to be more greatly influenced by the longer and more dominant time spans, in order to better reflect the primary swings of stock market cycle.

https://en.wikipedia.org/wiki/KST_oscillator

Parameters
• close (pandas.Series) – dataset ‘Close’ column.
• roc1 (int) – r1 period.
• roc2 (int) – r2 period.
• roc3 (int) – r3 period.
• roc4 (int) – r4 period.
• window1 (int) – n1 smoothed period.
• window2 (int) – n2 smoothed period.
• window3 (int) – n3 smoothed period.
• window4 (int) – n4 smoothed period.
• fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series
ta.trend.kst_sig(close, roc1=10, roc2=15, roc3=20, roc4=30, window1=10, window2=10, window3=10, window4=15, nsig=9, fillna=False)

KST Oscillator (KST Signal)
It is useful to identify major stock market cycle junctures because its formula is weighed to be more greatly influenced by the longer and more dominant time spans, in order to better reflect the primary swings of stock market cycle.


Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- roc1 (int) – roc1 period.
- roc2 (int) – roc2 period.
- roc3 (int) – roc3 period.
- roc4 (int) – roc4 period.
- window1 (int) – n1 smoothed period.
- window2 (int) – n2 smoothed period.
- window3 (int) – n3 smoothed period.
- window4 (int) – n4 smoothed period.
- nsig (int) – n period to signal.
- fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

`ta.trend.macd(close, window_slow=26, window_fast=12, fillna=False)`  
Moving Average Convergence Divergence (MACD)

Is a trend-following momentum indicator that shows the relationship between two moving averages of prices.

https://en.wikipedia.org/wiki/MACD

Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- window_fast (int) – n period short-term.
- window_slow (int) – n period long-term.
- fillna (bool) – if True, fill nan values.

Returns New feature generated.

Return type pandas.Series

`ta.trend.macd_diff(close, window_slow=26, window_fast=12, window_sign=9, fillna=False)`  
Moving Average Convergence Divergence (MACD Diff)

Shows the relationship between MACD and MACD Signal.

https://en.wikipedia.org/wiki/MACD

Parameters

- close (pandas.Series) – dataset ‘Close’ column.
- window_fast (int) – n period short-term.
- window_slow (int) – n period long-term.
• `window_sign (int)` – n period to signal.
• `fillna (bool)` – if True, fill nan values.

Returns: New feature generated.

Return type: pandas.Series

```python
ta.trend.macd_signal (close, window_slow=26, window_fast=12, window_sign=9, fillna=False)
```
Moving Average Convergence Divergence (MACD Signal)

Shows EMA of MACD.

https://en.wikipedia.org/wiki/MACD

Parameters

• `close (pandas.Series)` – dataset ‘Close’ column.
• `window_fast (int)` – n period short-term.
• `window_slow (int)` – n period long-term.
• `window_sign (int)` – n period to signal.
• `fillna (bool)` – if True, fill nan values.

Returns: New feature generated.

Return type: pandas.Series

```python
ta.trend.mass_index (high, low, window_fast=9, window_slow=25, fillna=False)
```
Mass Index (MI)

It uses the high-low range to identify trend reversals based on range expansions. It identifies range bulges that can foreshadow a reversal of the current trend.

http://stockcharts.com/school/doku.php?id=technical_indicators:mass_index

Parameters

• `high (pandas.Series)` – dataset ‘High’ column.
• `low (pandas.Series)` – dataset ‘Low’ column.
• `window_fast (int)` – fast window value.
• `window_slow (int)` – slow window value.
• `fillna (bool)` – if True, fill nan values.

Returns: New feature generated.

Return type: pandas.Series

```python
ta.trend.psar_down (high, low, close, step=0.02, max_step=0.2, fillna=False)
```
Parabolic Stop and Reverse (Parabolic SAR)

Returns the PSAR series with non-N/A values for downward trends


Parameters

• `high (pandas.Series)` – dataset ‘High’ column.
• `low (pandas.Series)` – dataset ‘Low’ column.
• `close (pandas.Series)` – dataset ‘Close’ column.
• **step** (*float*) – the Acceleration Factor used to compute the SAR.
• **max_step** (*float*) – the maximum value allowed for the Acceleration Factor.
• **fillna** (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.trend.psar_down_indicator(high, low, close, step=0.02, max_step=0.2, fillna=False)
```

Parabolic Stop and Reverse (Parabolic SAR) Downward Trend Indicator

Returns 1, if there is a reversal towards an downward trend. Else, returns 0.


**Parameters**

• **high** (*pandas.Series*) – dataset ‘High’ column.
• **low** (*pandas.Series*) – dataset ‘Low’ column.
• **close** (*pandas.Series*) – dataset ‘Close’ column.
• **step** (*float*) – the Acceleration Factor used to compute the SAR.
• **max_step** (*float*) – the maximum value allowed for the Acceleration Factor.
• **fillna** (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.trend.psar_up(high, low, close, step=0.02, max_step=0.2, fillna=False)
```

Parabolic Stop and Reverse (Parabolic SAR) Upward Trend Indicator

Returns the PSAR series with non-N/A values for upward trends


**Parameters**

• **high** (*pandas.Series*) – dataset ‘High’ column.
• **low** (*pandas.Series*) – dataset ‘Low’ column.
• **close** (*pandas.Series*) – dataset ‘Close’ column.
• **step** (*float*) – the Acceleration Factor used to compute the SAR.
• **max_step** (*float*) – the maximum value allowed for the Acceleration Factor.
• **fillna** (*bool*) – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

```python
ta.trend.psar_up_indicator(high, low, close, step=0.02, max_step=0.2, fillna=False)
```

Parabolic Stop and Reverse (Parabolic SAR) Upward Trend Indicator

Returns 1, if there is a reversal towards an upward trend. Else, returns 0.


**Parameters**

• **high** (*pandas.Series*) – dataset ‘High’ column.
• **low** *(pandas.Series)* – dataset ‘Low’ column.
• **close** *(pandas.Series)* – dataset ‘Close’ column.
• **step** *(float)* – the Acceleration Factor used to compute the SAR.
• **max_step** *(float)* – the maximum value allowed for the Acceleration Factor.
• **fillna** *(bool)* – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  pandas.Series

```python
ta.trend.sma_indicator(close, window=12, fillna=False)
```
Simple Moving Average (SMA)

**Returns**  New feature generated.

**Return type**  pandas.Series

```python
ta.trend.stc(close, window_slow=50, window_fast=23, cycle=10, smooth1=3, smooth2=3, fillna=False)
```
Schaff Trend Cycle (STC)

The Schaff Trend Cycle (STC) is a charting indicator that is commonly used to identify market trends and provide buy and sell signals to traders. Developed in 1999 by noted currency trader Doug Schaff, STC is a type of oscillator and is based on the assumption that, regardless of time frame, currency trends accelerate and decelerate in cyclical patterns.

[https://www.investopedia.com/articles/forex/10/schaff-trend-cycle-indicator.asp](https://www.investopedia.com/articles/forex/10/schaff-trend-cycle-indicator.asp)

**Parameters**

• **close** *(pandas.Series)* – dataset ‘Close’ column.
• **window_fast** *(int)* – n period short-term.
• **window_slow** *(int)* – n period long-term.
• **cycle** *(int)* – n period
• **smooth1** *(int)* – ema period over stoch_k
• **smooth2** *(int)* – ema period over stoch_kd
• **fillna** *(bool)* – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  pandas.Series

```python
ta.trend.trix(close, window=15, fillna=False)
```
Trix (TRIX)

Shows the percent rate of change of a triple exponentially smoothed moving average.


**Parameters**

• **close** *(pandas.Series)* – dataset ‘Close’ column.
• **window** *(int)* – n period.
• **fillna** *(bool)* – if True, fill nan values.

**Returns**  New feature generated.

**Return type**  pandas.Series
ta.trend.vortex_indicator_neg(high, low, close, window=14, fillna=False)

Vortex Indicator (VI)

It consists of two oscillators that capture positive and negative trend movement. A bearish signal triggers when the negative trend indicator crosses above the positive trend indicator or a key level.


Parameters

• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series

parameters
ta.trend.vortex_indicator_pos(high, low, close, window=14, fillna=False)

Vortex Indicator (VI)

It consists of two oscillators that capture positive and negative trend movement. A bullish signal triggers when the positive trend indicator crosses above the negative trend indicator or a key level.


Parameters

• high (pandas.Series) – dataset ‘High’ column.
• low (pandas.Series) – dataset ‘Low’ column.
• close (pandas.Series) – dataset ‘Close’ column.
• window (int) – n period.
• fillna (bool) – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series

ta.trend.wma_indicator(close, window=9, fillna=False)

Weighted Moving Average (WMA)

Returns  New feature generated.

Return type  pandas.Series

4.1.5 Others Indicators

Others Indicators.

class ta.others.CumulativeReturnIndicator(close: pandas.core.series.Series, fillna: bool = False)

Cumulative Return (CR)

Parameters

• close (pandas.Series) – dataset ‘Close’ column.
• `fillna (bool)` – if True, fill nan values.

**cumulative_return**() → pandas.core.series.Series
Cumulative Return (CR)

**Returns** New feature generated.

**Return type** pandas.Series

**class** `ta.others.DailyLogReturnIndicator (close: pandas.core.series.Series, fillna: bool = False)`
Daily Log Return (DLR)


**Parameters**

• `close (pandas.Series)` – dataset ‘Close’ column.

• `fillna (bool)` – if True, fill nan values.

**daily_log_return**() → pandas.core.series.Series
Daily Log Return (DLR)

**Returns** New feature generated.

**Return type** pandas.Series

**class** `ta.others.DailyReturnIndicator (close: pandas.core.series.Series, fillna: bool = False)`
Daily Return (DR)

**Parameters**

• `close (pandas.Series)` – dataset ‘Close’ column.

• `fillna (bool)` – if True, fill nan values.

**daily_return**() → pandas.core.series.Series
Daily Return (DR)

**Returns** New feature generated.

**Return type** pandas.Series

**ta.others.cumulative_return (close, fillna=False)**
Cumulative Return (CR)

**Parameters**

• `close (pandas.Series)` – dataset ‘Close’ column.

• `fillna (bool)` – if True, fill nan values.

**Returns** New feature generated.

**Return type** pandas.Series

**ta.others.daily_log_return (close, fillna=False)**
Daily Log Return (DLR)


**Parameters**

• `close (pandas.Series)` – dataset ‘Close’ column.

• `fillna (bool)` – if True, fill nan values.

**Returns** New feature generated.
Return type  pandas.Series

`ta.others.daily_return(close, fillna=False)`
Daily Return (DR)

Parameters

- `fillna` *(bool)* – if True, fill nan values.

Returns  New feature generated.

Return type  pandas.Series
INDICES AND TABLES

- genindex
- modindex
- search
m
momentum, 9

o
others, 58

r
    ta, 9
    ta.momentum, 9
    ta.others, 58
    ta.trend, 41
    ta.volatility, 30
    ta.volume, 22
    trend, 41

v
volatility, 30
volume, 22
INDEX

A
acc_dist_index() (in module ta.volume), 26
acc_dist_index() (ta.volume.AccDistIndexIndicator method), 23
AccDistIndexIndicator (class in ta.volume), 22
adx() (in module ta.trend), 49
adx() (ta.trend.ADXIndicator method), 41
adx_neg() (in module ta.trend), 49
adx_neg() (ta.trend.ADXIndicator method), 41
adx_pos() (in module ta.trend), 50
adx_pos() (ta.trend.ADXIndicator method), 41
ADXIndicator (class in ta.trend), 41
aroon_down() (in module ta.trend), 50
aroon_down() (ta.trend.AroonIndicator method), 42
aroon_indicator() (ta.trend.AroonIndicator method), 42
aroon_up() (in module ta.trend), 50
aroon_up() (ta.trend.AroonIndicator method), 42
AroonIndicator (class in ta.trend), 42
average_true_range() (in module ta.volatility), 34
average_true_range() (ta.volatility.AverageTrueRange method), 30
AverageTrueRange (class in ta.volatility), 30
awesome_oscillator() (in module ta.momentum), 15
awesome_oscillator() (ta.momentum.AwesomeOscillatorIndicator method), 9
AwesomeOscillatorIndicator (class in ta.momentum), 9

B
bollinger_hband() (in module ta.volatility), 34
bollinger_hband() (ta.volatility.BollingerBands method), 30
bollinger_hband_indicator() (in module ta.volatility), 34
bollinger_hband_indicator() (ta.volatility.BollingerBands method), 31
bollinger_lband() (in module ta.volatility), 34
bollinger_lband() (ta.volatility.BollingerBands method), 31
bollinger_lband_indicator() (in module ta.volatility), 35
bollinger_lband_indicator() (ta.volatility.BollingerBands method), 31
bollinger_mavg() (in module ta.volatility), 35
bollinger_mavg() (ta.volatility.BollingerBands method), 31
bollinger_pbands() (in module ta.volatility), 35
bollinger_pbands() (ta.volatility.BollingerBands method), 31
bollinger_wbands() (in module ta.volatility), 36
bollinger_wbands() (ta.volatility.BollingerBands method), 31
BollingerBands (class in ta.volatility), 30

ci() (in module ta.trend), 51
CCIIndicator (class in ta.trend), 42
CCIIndicator (class in ta.trend), 42
chaikin_money_flow() (in module ta.volume), 26
chaikin_money_flow() (ta.volume.ChaikinMoneyFlowIndicator method), 23
ChaikinMoneyFlowIndicator (class in ta.volume), 23
cumulative_return() (in module ta.others), 59
cumulative_return() (ta.others.CumulativeReturnIndicator method), 59
CumulativeReturnIndicator (class in ta.others), 58

daily_log_return() (in module ta.others), 59
daily_log_return() (ta.others.DailyLogReturnIndicator method), 59
daily_return() (in module ta.others), 60
daily_return() (ta.others.DailyReturnIndicator method), 59
DailyLogReturnIndicator (class in ta.others), 59
DailyReturnIndicator (class in ta.others), 59
donchian_channel_hband() (in module ta.volatility), 36
donchian_channel_hband() (ta.volatility.DonchianChannel 32
donchian_channel_lband() (in module ta.volatility), 36
donchian_channel_lband() (ta.volatility.DonchianChannel 32
donchian_channel_mband() (in module ta.volatility), 37
donchian_channel_mband() (ta.volatility.DonchianChannel 32
donchian_channel_pband() (in module ta.volatility), 37
donchian_channel_pband() (ta.volatility.DonchianChannel 32
donchian_channel_wband() (in module ta.volatility), 37
donchian_channel_wband() (ta.volatility.DonchianChannel 32
DonchianChannel (class in ta.volatility), 31
dpo() (in module ta.trend), 51
dpo() (ta.trend.DPOIndicator method), 43
DPOIndicator (class in ta.trend), 43

E
ease_of_movement() (in module ta.volume), 27
ease_of_movement() (ta.volumeEaseOfMovementIndicator method), 23
EaseOfMovementIndicator (class in ta.volume), 23
dma_indicator() (in module ta.trend), 51
dma_indicator() (ta.trend.EMAIndicator method), 43
EMAIndicator (class in ta.trend), 43

F
force_index() (in module ta.volume), 27
force_index() (ta.volumeForceIndexIndicator method), 24
ForceIndexIndicator (class in ta.volume), 24

I
ichimoku_a() (in module ta.trend), 51
ichimoku_a() (ta.trend.IchimokuIndicator method), 44
ichimoku_b() (ta.trend.IchimokuIndicator method), 44
ichimoku_base_line() (in module ta.trend), 52
ichimoku_base_line() (ta.trend.IchimokuIndicator method), 44
ichimoku_conversion_line() (in module ta.trend), 53
ichimoku_conversion_line() (ta.trend.IchimokuIndicator method), 44
IchimokuIndicator (class in ta.trend), 43

K
kama() (in module ta.momentum), 16
kama() (ta.momentum.KAMAIndicator method), 10
KAMAIndicator (class in ta.momentum), 10
keltner_channel_hband() (in module ta.volatility), 37
keltner_channel_hband() (ta.volatility.KeltnerChannel method), 32
keltner_channel_hband_indicator() (in module ta.volatility), 38
keltner_channel_hband_indicator() (ta.volatility.KeltnerChannel method), 33
keltner_channel_lband() (in module ta.volatility), 38
keltner_channel_lband() (ta.volatility.KeltnerChannel method), 33
keltner_channel_lband_indicator() (in module ta.volatility), 39
keltner_channel_lband_indicator() (ta.volatility.KeltnerChannel method), 33
keltner_channel_mband() (in module ta.volatility), 39
keltner_channel_mband() (ta.volatility.KeltnerChannel method), 33
keltner_channel_pband() (in module ta.volatility), 40
keltner_channel_pband() (ta.volatility.KeltnerChannel method), 33
keltner_channel_wband() (in module ta.volatility), 40
keltner_channel_wband() (ta.volatility.KeltnerChannel method), 33
KeltnerChannel (class in ta.volatility), 32
kst() (in module ta.trend), 53
kst() (ta.trend.KSTIndicator method), 44
kst_diff() (ta.trend.KSTIndicator method), 45
kst_sig() (in module ta.trend), 53
kst_sig() (ta.trend.KSTIndicator method), 45
KSTIndicator (class in ta.trend), 44

M
MACD (class in ta.trend), 45
macd() (in module ta.trend), 54
macd() (ta.trend.MACD method), 45
macd_diff() (in module ta.trend), 54
macd_diff() (ta.trend.MACD method), 45
macd_signal() (in module ta.trend), 55
macd_signal() (ta.trend.MACD method), 45
mass_index() (in module ta.trend), 55
mass_index() (ta.trend.MassIndex method), 46
money_flow_index() (in module ta.volume), 27
money_flow_index() (ta.volume.MFIIndicator method), 24

N
negative_volume_index() (in module ta.volume), 28
negative_volume_index()
  (ta.volume.NegativeVolumeIndexIndicator method), 25
NegativeVolumeIndexIndicator (class in ta.volume), 25

O
on_balance_volume() (in module ta.volume), 28
on_balance_volume()
  (ta.volume.OnBalanceVolumeIndicator method), 25
OnBalanceVolumeIndicator (class in ta.volume), 25
others (module), 58

P
PercentagePriceOscillator (class in ta.momentum), 10
PercentageVolumeOscillator (class in ta.momentum), 11
psar() (in module ta.trend), 16
psar() (ta.trend.PSARIndicator method), 10
psar_hist() (in module ta.trend), 16
psar_hist() (ta.trend.PercentagePriceOscillator method), 10
psar_signal() (in module ta.trend), 17
psar_signal() (ta.trend.PercentagePriceOscillator method), 11
psar() (ta.trend.PSARIndicator method), 46
psar_down() (in module ta.trend), 55
psar_down() (ta.trend.PSARIndicator method), 46
psar_down_indicator() (in module ta.trend), 56
psar_down_indicator() (ta.trend.PSARIndicator method), 46
psar_up() (in module ta.trend), 56

R
roc() (in module ta.momentum), 18
roc() (ta.momentum.ROCIndicator method), 12
ROCIndicator (class in ta.momentum), 11
rsi() (in module ta.momentum), 18
rsi() (ta.momentum.RSIIndicator method), 12
RSIIndicator (class in ta.momentum), 12

S
sma ease of movement() (in module ta.volume), 29
sma ease of movement()
  (ta.volume.EaseOfMovementIndicator method), 24
sma_indicator() (in module ta.trend), 57
sma_indicator()
  (ta.trend.SMAIndicator method), 47
SMAIndicator (class in ta.trend), 47
stc() (in module ta.trend), 57
stc() (ta.trend.STCIndicator method), 47
STCIndicator (class in ta.trend), 47
stoch() (in module ta.momentum), 19
stoch() (ta.momentum.StochasticOscillator method), 13
stoch_signal() (in module ta.momentum), 19
stoch_signal() (ta.momentum.StochasticOscillator method), 13
StochasticOscillator (class in ta.momentum), 13
stochrsi() (in module ta.momentum), 19
stochrsi()
  (ta.momentum.StochRSIIndicator method), 12
stochrsi_d() (in module ta.momentum), 20
stochrsi_d()
  (ta.momentum.StochRSIIndicator method), 20
stochrsi_k() (in module ta.momentum), 20
stochrsi_k()
  (ta.momentum.StochRSIIndicator method), 13
StochRSIIndicator (class in ta.momentum), 12

Index
Technical Analysis Library in Python Documentation, Release 0.1.4

T

ta (module), 9
ta.momentum (module), 9
ta.others (module), 58
ta.trend (module), 41
ta.volatility (module), 30
ta.volume (module), 22
trend (module), 41
trix() (in module ta.trend), 57
trix() (ta.trend.TRIXIndicator method), 48
TRIXIndicator (class in ta.trend), 48
tsi() (in module ta.momentum), 21
tsi() (ta.momentum.TSIIndicator method), 13
TSIIndicator (class in ta.momentum), 13

U

ulcer_index() (in module ta.volatility), 40
ulcer_index() (ta.momentum.UlcerIndex method), 33
UlcerIndex (class in ta.momentum), 33
ultimate_oscillator() (in module ta.momentum), 21
ultimate_oscillator() (ta.momentum.UltimateOscillator method), 14
UltimateOscillator (class in ta.momentum), 14

V

volatility (module), 30
volume (module), 22
volume_price_trend() (in module ta.volume), 29
volume_price_trend()
  (ta.volume.VolumePriceTrendIndicator method), 25
volume_weighted_average_price() (in module ta.volume), 29
volume_weighted_average_price()
  (ta.volume.VolumeWeightedAveragePrice method), 26
VolumePriceTrendIndicator (class in ta.volume), 25
VolumeWeightedAveragePrice (class in ta.volume), 26
vortex_indicator_diff()
  (ta.trend.VortexIndicator method), 48
vortex_indicator_neg() (in module ta.trend), 58
vortex_indicator_neg()
  (ta.trend.VortexIndicator method), 48
vortex_indicator_pos() (in module ta.trend), 58
vortex_indicator_pos()
  (ta.trend.VortexIndicator method), 48
VortexIndicator (class in ta.trend), 48

W

williams_r() (in module ta.momentum), 22
williams_r() (ta.momentum.WilliamsRIndicator method), 15
WilliamsRIndicator (class in ta.momentum), 14
wma() (ta.trend.WMAIndicator method), 49
wma_indicator() (in module ta.trend), 58
WMAIndicator (class in ta.trend), 49