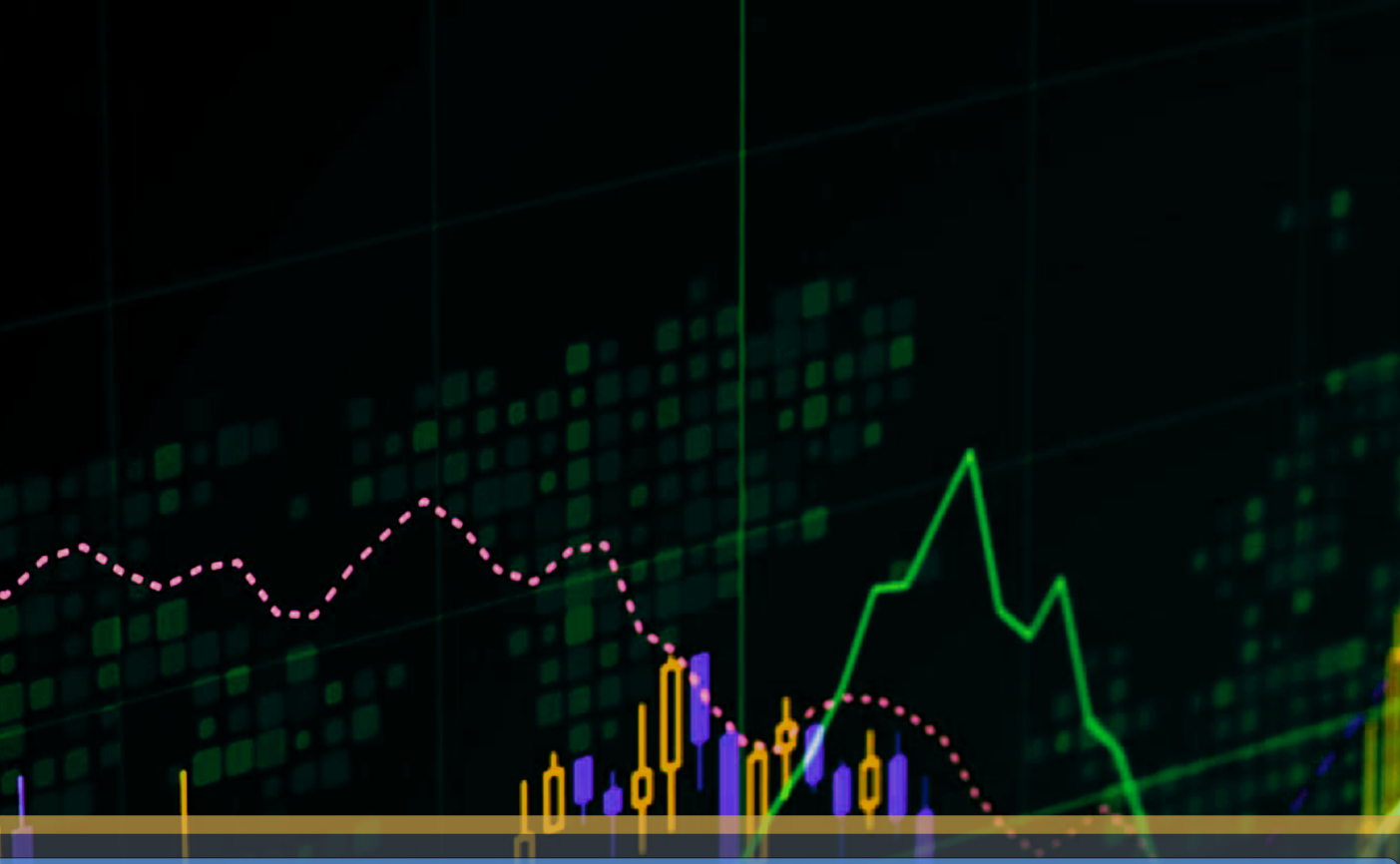
IMPORTANT DISCLOSURE

THE CODE IN THIS BOOK IS OUTDATED. I HIGHLY RECOMMEND YOU FOCUS ON THE THEORETICAL CONCEPTS AND NOT ON THE CODE.



CONTRARIAN TRADING STRATEGIES IN PYTHON

CONTRARIAN INDICATORS & STRATEGIES

SOFIEN KAABAR

To my parents for always believing in me.

To my computer for putting up with my constant need for performance.

To my body for handling so much food to refrain from sleeping at nights writing this book.

To my eyes for burdening them with the constant need to stay awake during the nights.

To my wife who supported my absence because I was writing this book.

To everyone who reads this and gains something…

Q&A

What is this book all about?

This book is a modest attempt at presenting a more modern version of technical analysis based on objective measures rather than subjective ones. A sizeable chunk of this beautiful type of analysis revolves around technical indicators which is what this book covers. I believe it is time to be creative with indicators. The following chapters present contrarian indicators and how to code/use them. The code included in the book is available in the GitHub repository. A QR code link will be provided in the book.

What am I going to gain?

You will gain exposure to many new indicators and strategies that will change the way you think about trading, and you will find yourself busy experimenting and choosing the strategy that suits you the best.

How is it organized?

The order of the chapter is not very important, although reading the introductory Python chapter is helpful. The book is divided into four parts: Part 1 deals with primary contrarian indicators, Part 2 deals with secondary contrarian indicators, Part 3 deals with new indicators that I have developed, and finally, Part 4 will present many different contrarian technical strategies.

What level of knowledge do I need to follow this book?

Although a basic or a good understanding of trading and coding is considered very helpful, it is not necessary. At the beginning of the book, I have included a chapter that deals with some Python concepts, but this book is not about Python.

Table of Contents

INTRODUCTION

Contrarian trading is a forecasting technique that aims to profit from the expected short-term and long-term market reactions. We also tend to say mean reversion to signify trades that aim to expect a return to normality. In a bullish (rising) market, a contrarian trader would short[1] (sell) and expect a bearish reaction. In parallel, in a bearish (falling) market, a contrarian trader would long (buy) and expect a bullish reaction.

Contrarian trading can be based on fundamental analysis, technical analysis, and even quantitative analysis. This book deals with technical analysis and discusses a wide array of technical indicators and strategies that aim to fade the current direction. It is composed of four different parts with **Part 1** reserved for the well-known contrarian indicators and how to use them properly. **Part 2** can be used as a mini encyclopedia of contrarian indicators that are less known than the ones presented in the first part. **Part 3** discusses the indicators that I have personally developed, they are part of K's collection which aims to create enhanced technical indicators. And finally, **Part 4** presents 15 contrarian strategies created after combining different indicators. The aim is to provide structured strategies and other techniques uncovered in previous parts.

You must understand that the techniques and strategies of the book are not recommendations, and you must back-test them while adding your personal risk management and the necessary optimization.

Get your code on!

[1] Short selling is the act of borrowing an asset from a third party and selling it to another while waiting for the price to go down before buying it back at a lower price and returning it to the third party, thus pocketing the price difference and benefitting from the price decrease.

INTRODUCTION TO DATA ANALYSIS IN PYTHON

In research and trading, the idea generation phase must have enough time to deliver good results, and this cannot be done if you have a long process of transforming the ideas into visible results. Automating the process of historical data import is valuable as it allows you to focus on more important and non-repetitive tasks.

By creating a few simple functions, you will be able to call thousands of OHLC[2] data in seconds. For this, you need to install a library and a software described in this chapter. But first, you need to download a Python interpreter such as SPYDER[3], PyCharm, or Jupyter.

METATRADER5 SOFTWARE & LIBRARY

One of the most famous trading platforms in the retail community is the MetaTrader5 software. It is a powerful tool that comes with its own programming language and its huge online community support. Most importantly, it offers the possibility to export its historical short-term and long-term data. The first thing you need to do is to simply download the software from the official website[4]. Then, after creating the unlimited demo account, you are ready to import the library in Python that allows you to fetch the OHLC data from MetaTrader5. A library is a group of structured functions that can be imported into the Python interpreter from where you can call and use the ones you want. The easiest way to install the MT5 library is to go to the Python prompt and type:

```
pip install MetaTrader5
```

[2] Open, High, Low, and Close values are the most basic blocks of financial data.
[3] https://www.anaconda.com/
[4] https://www.metatrader5.com/en

This should install the library in your local Python. Now, you must import it to the Python interpreter such as SPYDER so that you can use it.

You do not need to copy any code (except for the previous one) in this book as you can download it from GitHub's link below[5]. Otherwise, you can just scan the following QR code to get to the page of the GitHub's repository. For now, focus on understanding the concepts and the ideas as there are plenty.

To start the importing process, you must write the following syntax which lets SPYDER (or the interpreter you have chosen) know the functions you want to use. This is known as importing the necessary libraries.

```
import datetime # Date acquiring
import pytz # Time zone management
import pandas as pd # Mostly for Data frame manipulation
import MetaTrader5 as mt5 # Importing OHLC data
import matplotlib.pyplot as plt # Plotting charts
import numpy as np # Mostly for array manipulation
```

Anything that comes after **as** is a shortcut. The **plt** shortcut is there so that each time you want to call a function from that library, you do not have to type the

[5] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

full **matplotlib.pyplot** statement. The official documentation for the library can be found in the official website[6]. Let's get started with the importing process

CREATING THE IMPORTING FUNCTIONS STEP-BY-STEP

The main goal is to have OHLC data arrays in SPYDER (or the interpreter you have chosen). The first thing to do is to select which time frame[7] you want to import (and analyze). I will primarily use the hourly time frame throughout the book but let's also consider adding the daily time frame as well:

```
# Choosing the hourly time frame
frame_H1   = mt5.TIMEFRAME_H1
# Choosing the daily time frame
frame_D1   = mt5.TIMEFRAME_D1
```

Then, by staying in the spirit of importing variables, you can define the variable that shows the current time and date. This helps the algorithm know the stopping date of the import. To do this, use the following code:

```
# Defining the variable now to give out the current date
now = datetime.datetime.now()
```

Reminder

You do not need to copy the code as you read through the book because everything is organized in the repository. The aim from presenting the code in the book is to explain it.

The next syntax shows the list of assets that you will be able to import.

[6] https://www.mql5.com/en/docs/integration/python_metatrader5

[7] A time frame is the frequency of change between price points (or bars). An hourly time frame records and shows the OHLC data every 60 minutes while a daily time frame records and shows the OHLC every 24 hours.

```
assets = ['EURUSD', 'USDCHF', 'GBPUSD',
          'AUDUSD', 'USDCAD', 'XAUUSD',
          'XAGUSD', 'NI225', 'SP500m']
```

The assets that I present and use as examples are summarized as follows:

- **EURUSD** is the value of 1 EUR in USD terms. It is the most liquid currency pair in the world.
- **USDCHF** is the value of 1 USD in CHF terms. It is one of the major currency pairs and is generally regarded as a safe haven[8] currency.
- **GBPUSD** is the value of 1 GBP in USD terms. It is one of the major currency pairs and is heavily correlated to the UK's economy.
- **USDCAD** is the value of 1 USD in CAD terms. It is one of the major currency pairs and is generally negatively correlated to oil.
- **XAUUSD** is the value of Gold in USD terms. It is a benchmark in the world of precious metals.
- **XAGUSD** is the value of Silver in USD terms. It is also a benchmark in the world of precious metals but also in the industrial world.
- **NI225** is the benchmark of Japanese equities. The Nikkei225 measures the overall health of Japan's stock market.
- **SP500m** is the benchmark of US equities. The S&P500 measures the overall health of the US stock market.

The next function establishes a connection to MetaTrader5 (which needs to be open), applies the current date, and extracts the needed data. I have inputted **Europe/Paris** as my time zone, you should use your time zone to get more accurate data.

[8] A safe haven currency is heavily traded during times of political and economic crises. In an event called flight to safety, market participants generally buy CHF as it tends to appreciate in value. Another safe haven currency is the Japanese yen (JPY).

```
def get_quotes(time_frame, year = 2005, month = 1, day = 1, asset = "EURUSD"):
    if not mt5.initialize():
    print("initialize() failed, error code =", mt5.last_error())
    quit()
    timezone = pytz.timezone("Europe/Paris")
    time_from = datetime.datetime(year, month, day, tzinfo = timezone)
    time_to = datetime.datetime.now(timezone) + datetime.timedelta(days=1)
    rates = mt5.copy_rates_range(asset, time_frame, time_from, time_to)
    rates_frame = pd.DataFrame(rates)
    return rates_frame
```

And finally, you should use the **mass_import( )** function that incorporates the previous function to import the data and present it as a clean OHLC array.

```
def mass_import(asset, time_frame):
    if time_frame == 'H1':
        data = get_quotes(frame_H1, 2020, 1, 1, asset = assets[asset])
        data = data.iloc[:, 1:5].values
        data = data.round(decimals = 5)
    if time_frame == 'D1':
        data = get_quotes(frame_D1, 2000, 1, 1, asset = assets[asset])
        data = data.iloc[:, 1:5].values
        data = data.round(decimals = 5)
    return data
```

Important

If you are a macOS user, please refer to the Appendix to see how to get the historical data in another method. Generally, the technique presented in this chapter works well for Windows users.

Finally, you are done building the blocks necessary to import the data. Let's see how to apply the previous code. To import EURUSD's OHLC historical data, follow the next line of code:

```
# Choosing the asset
pair = 0
# Time Frame
horizon = 'H1'
# Importing the asset as an array
my_data = mass_import(pair, horizon)
```

Now, you have an array called EURUSD with historical OHLC data inside. The first phase is done and now, you will proceed to data analysis. The utility of coding is bigger than ever nowadays. A trader or an analyst must know at least the basics of how algorithms work and must also know some functions and the logic behind the calculations. Many analysts use indicators but do not know how they are calculated nor how they can be coded.

As this book presents many indicators, you need to understand the basics, and prepare the necessary functions to analyze the data.

BASIC SYNTAX AND REQUIRED FUNCTIONS

Python is rapidly growing to be the go-to language when it comes to creating and testing strategies. It is a fast, versatile, and easy language that allows you to perform many operations. Basically, for the most part of the book, you will be using 3 known libraries to handle, manipulate, and plot the data. Let's start with the two similar libraries, **numpy** [9] and **pandas** [10] (which will be used to a much lesser extent than **numpy**). Every python developer must imperatively

[9] https://docs.scipy.org/doc/numpy/user/whatisnumpy.html
[10] https://pandas.pydata.org/pandas-docs/stable/

know a thing or two about them. You use **numpy** to manipulate arrays and perform operations and other useful mathematical functions, it is the benchmark package when you are dealing with time series or any other dataset as it is fast due to vectorization.

As for **pandas**, it is also similar to numpy with some differences. Of course, these are simplistic definitions, and the libraries can do much more than the described tasks. The library **matplotlib** is used to do fancy plotting and charting.

In python, you import a library using the **import** statement followed by the name of the library and an optional shortcut or nickname for that library so you can call it later.

The word **np** is the standard numpy shortcut while **pd** is the standard pandas shortcut. Also, **plt** is the standard matplotlib shortcut.

The next key task is to add columns which will later be populated by values of technical indicators or signal proxies[11].

Let's start by adding a column. Remember, you have four columns containing OHLC data and you want to add a fifth column containing zero values so that you create an indicator[12] function and populate the empty column with its values. To add a new empty column, you can use the following function.

[11] A signal proxy is a value that designates either a buy or a sell signal. In this book, I use 1 as a buy signal and -1 as a sell signal.

[12] A technical indicator is a price-derived transformation that gives you statistical and mathematical information about the current and past behavior of the price around certain subjective or objective benchmark points.

```
def add_column(data, times):
    for i in range(1, times + 1):
        new = np.zeros((len(data), 1), dtype = float)
        data = np.append(data, new, axis = 1)
    return data
```

Sometimes, you need to remove columns when you have more than enough or when you want to clean out the data. Use the following function:

```
def delete_column(data, index, times):
    for i in range(1, times + 1):
        data = np.delete(data, index, axis = 1)
    return data
```

It is worth mentioning that Python starts indexing at zero, therefore column 5 is chronologically the sixth column and not the fifth column. Similarly, row number 5 is the sixth row and not the fifth row.

Sometimes, you calculate indicators that start from the 100th line[13], therefore, the first 100 lines will have no use. The next page shows a function to delete several rows from an array that is descending in order. This means that the first values are the earliest ones with the latest values below being the latest ones.

```
def delete_row(data, number):
    data = data[number:, ]
    return data
```

[13] Consider an indicator that uses the last 100 data to have a current value. Naturally, you should not have any values for the indicator in the first 99 data because it could not be calculated due to insufficient data.

PLOTTING DATA AND VISUALIZATION

Plotting and visualization is very easy in Python and does not require any sophisticated code. Let's consider as usual that you have an OHLC array, and you want to visualize the latest 500 closing prices in black with a legend that describes what you are seeing. You also want to see a grid on the chart to facilitate the visual interpretation. For this, use the following syntax.

```
# Plotting
plt.plot(my_data[-500:, 3], color = 'black', label = 'Close Price')
# Creating the grid
plt.grid()
# Calling the legend Function
plt.legend()
```

The above function states that you want to plot the latest 500 values (hence, the negative sign) of the fourth column (hence, index 3 as Python starts indexing from zero). You also want the color to be black which is why the argument of **color** equals what you want it to be. The variable **label** is a string (text) of what you want to describe. This is what will appear on the chart in the form of a text box called later by the **legend( )** function. Finally, the **grid( )** function is self-explanatory. It is optional but it adds a visual effect to the chart.

CODING THE SIGNALS

When you want to create trading techniques and strategies out of indicators, you must deal with two actions:

- Create the technical indicator(s) in the column(s) next to the OHLC data.
- Create the conditions of buy and sell proxies and populate the required columns.

Let's code the most basic of technical indicators, the simple moving average and then, the conditions of the simplest trading strategy in existence, the cross technique. A moving average is the mean of a moving window of the close prices which can be used to filter the trend or to generate signals.

The next code snippet defines the moving average function which is a column created after the OHLC data and contains the moving average of the close price using a lookback period that the user chooses through the **lookback** variable.

```
def ma(data, lookback, close, position):
        data = add_column(data, 1)
        for i in range(len(data)):
                try:
                        data[i, position] = (data[i - lookback + 1:i + 1, close].mean())
                except IndexError:
                        pass
        data = delete_row(data, lookback)
        return data
```

The next code calls the **ma( )** function and the **signal( )** function which has the following conditions:

- **A long (buy) signal is generated whenever the close price surpasses its 20-period moving average.**
- **A short (sell) signal is generated whenever the close price breaks its 20-period moving average.**

The way to code this algorithmically is to impose a condition on the current close price relative to the current moving average and the previous close price relative to the previous moving average.

```
# Calling a 20-period moving average (populating column 4)
my_data = ma(my_data, 20, 3, 4)
# Defining the signal function that populates the buy/sell columns
def signal(data, close, ma_col, buy, sell):
    data = add_column(data, 2)
    for i in range(len(data)):
        if data[i, close] > data[i, ma_col] and data[i - 1, close] < data[i - 1, ma_col]:
            data[i + 1, buy] = 1
        if data[i, close] < data[i, ma_col] and data[i - 1, close] > data[i - 1, ma_col]:
            data[i + 1, sell] = -1
    return data
# Using the signal function
my_data = signal(my_data, 4, 5, 6)
```

The **signal( )** function defines the signals where the variable data refers to the OHLC array containing the historical data, the variable close refers to the column of the close price, the variable ma_col refers to the column of the moving average. The buy and sell variables refer to the columns where bullish and bearish signals are outputted.

After execution, you should have a data array with the following elements:

- OHLC data populating the first four columns.
- A 20-period moving average in the fifth column (index = 4).
- Buy signals with the value of 1 in the sixth column (index = 5).
- Sell signals with the value of -1 in the seventh column (index = 6).

Note that I use the notation **i + 1** in the signal proxy to state that the position is opened in the next open price as opposed to the current close. This is to make the simulation more realistic as it is difficult to open a position at exactly the close price.

A quick and easy way to plot the signals is to place upward pointing arrows whenever a buy signal is generated and downward pointing arrows whenever a sell signal is generated. Since the signal chart function is lengthy, I have chosen to include it only in the GitHub repository. You do not need to worry about it as it is present in the code of every technique and strategy and thus, you will always see the signals superimposed on the chart when you run the code.

Important

Make sure you download the **Master_Functions.py** file and execute all the code inside. By doing this, you will be able to smoothly run the code from the other techniques and strategies seen in this book.

To summarize, in this chapter, I have shown you how to automatically import financial OHLC data from Metatrader5 and how to code the moving average indicator. Also, I have shown the rationale and the intuition of the signal function which is the core of the algorithm as it represents the conditions necessary for any technique or strategy (when to buy and when to sell). You must understand that throughout the book, you will be dealing with columns within an array which is mostly organized as OHLC data followed by the indicator(s), the signals (conditions), and the performance metrics (in the case of strategies in the last part of the book).

PART 1 PRIMARY CONTRARIAN INDICATORS

Primary technical indicators are the ones that are considered to be the pillars of technical analysis. These indicators are present in almost every technical analysis or technical trading system. In my experience, the primary contrarian indicators are the following:

- The relative strength index.
- The stochastic oscillator.
- The Bollinger bands.

Every indicator in the list has its uniqueness and its strengths but also its weaknesses. Your main goal from this chapter is to grasp the concepts of the three indicators and understand how they are calculated and used. As of now, I will present:

- **Techniques**: These are presented in the first three parts of the book. Techniques are simply ways to use the indicator but are not considered pure strategies.
- **Strategies**: These are structured conditions based on a combination of indicators or a special way to use one indicator that should add a predictive value when applied in a full trading framework[14].

Hence, for every indicator in this part, I will present a number of techniques, the code used to generate the signals, and the limitations of the technique. Your main learning outcome is to see different ways to use the primary contrarian indicators.

[14] A full trading framework must include a set of objective conditions with clear entry and exit rules based on extensive back-testing and non-simplistic methods. It must also include a full risk management system and a position-sizing algorithm.

CHAPTER 1 THE RELATIVE STRENGTH INDEX

The relative strength index (RSI) is regarded as the most known and used technical indicator and this is no surprise as its simplicity and versatility make it very accessible. Created by Welles Wilder Jr., the RSI is a contrarian indicator bounded between 0 and 100 that reflects the price action for a specified moving window. Let's see the steps needed to create the RSI:

- Calculate the difference between the current close price and the previous close price.
- Separate the positive changes (where the current close price is greater than the previous close price) from the negative changes (where the current close price is lower than the previous close price) as you will transform them individually in the next step.
- Calculate a 14-period smoothed moving average[15] on the positive changes and on the absolute[16] values of the negative changes.
- Divide the first moving average values (applied on the positive changes) by the second moving average values (applied on the absolute negative changes). The result of this step is called the relative strength (RS).
- To calculate the relative strength index, use the below formula:

$$RSI_i = 100 - \left(\frac{100}{1 + RS_i}\right)$$

The RSI is always between 0 and 100, therefore values close to the extremes may signal a market reaction.

[15] The smoothed version is a special type of moving averages that uses a different weighting method than a plain simple one. I have discussed it in detail in my previous book "Trend following strategies in Python".
[16] Absolute refers to the act of imposing any negative value to be positive.

Figure 1-1 shows the values of the hourly USDCHF values with the 14-period RSI.

Figure 1-1 USDCHF hourly values with the 14-period RSI.

Values close to 30 signal an oversold market while values close to 70 signal an overbought market. The below shows the difference between the two concepts:

- An **oversold condition** is when a market has fallen and reached an area historically associated with a bounce or a change of dynamic.
- An **overbought condition** is when a market has risen and reached an area historically associated with a pause or a change of dynamic.

Before defining the RSI, you need to define the smoothed moving average which in turn uses the code from the simple moving average I have presented earlier.

```
def smoothed_ma(data, alpha, lookback, close, position):
    lookback = (2 * lookback) - 1
    alpha = alpha / (lookback + 1.0)
    beta  = 1 - alpha
    data = ma(data, lookback, close, position)
    data[lookback + 1, position] = (data[lookback + 1, close] * alpha) +
                                   (data[lookback, position] * beta)
    for i in range(lookback + 2, len(data)):
        try:
            data[i, position] = (data[i, close] * alpha) + (data[i - 1, position] * beta)
        except IndexError:
            pass
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable alpha[17] is by default 2, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

Now, let's take a look at the RSI function in the next code block.

[17] This is a constant used in the calculation of the smoothed moving average. Some traders may put 3 but it is very uncommon.

```
def rsi(data, lookback, close, position):
        data = add_column(data, 5)
        for i in range(len(data)):
                data[i, position] = data[i, close] - data[i - 1, close]
        for i in range(len(data)):
                if data[i, position] > 0:
                        data[i, position + 1] = data[i, position]
                elif data[i, position] < 0:
                        data[i, position + 2] = abs(data[i, position])
        data = smoothed_ma (data, 2, lookback, position + 1, position + 3)
        data = smoothed_ma (data, 2, lookback, position + 2, position + 4)
        data[:, position + 5] = data[:, position + 3] / data[:, position + 4]
        data[:, position + 6] = (100 - (100 / (1 + data[:, position + 5])))
        data = delete_column(data, position, 6)
        data = delete_row(data, lookback)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

Let's now get started with the techniques that you can apply using this indicator. Remember that techniques are not strategies but merely simple ways and ideas to use the indicators. Also, remember that the word long is synonymous with buying and the word sell is synonymous with profiting from a falling price in a process referred to as short selling.

SECTION 1.1 THE AGGRESSIVE TECHNIQUE

The first technique generally taught when it comes to the RSI is to initiate trades the moment it reaches the implied extremes which are also known as oversold and overbought levels. Generally, with a 14-period RSI, the oversold level is 30 and the overbought level is 70. These are agreed-upon levels where participants think that the market is outstretched and is ready for a counter reaction. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the RSI reaches the oversold level.**
- **A short signal is generated whenever the RSI reaches the overbought level.**

First, let's start with the signal function which is basically the core of the technique and the one telling the algorithm where to place the trades .

```
def signal(data, rsi_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, rsi_column] < lower_barrier and data[i - 1, rsi_column] > lower_barrier:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, rsi_column] > upper_barrier and data[i - 1, rsi_column] < upper_barrier:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

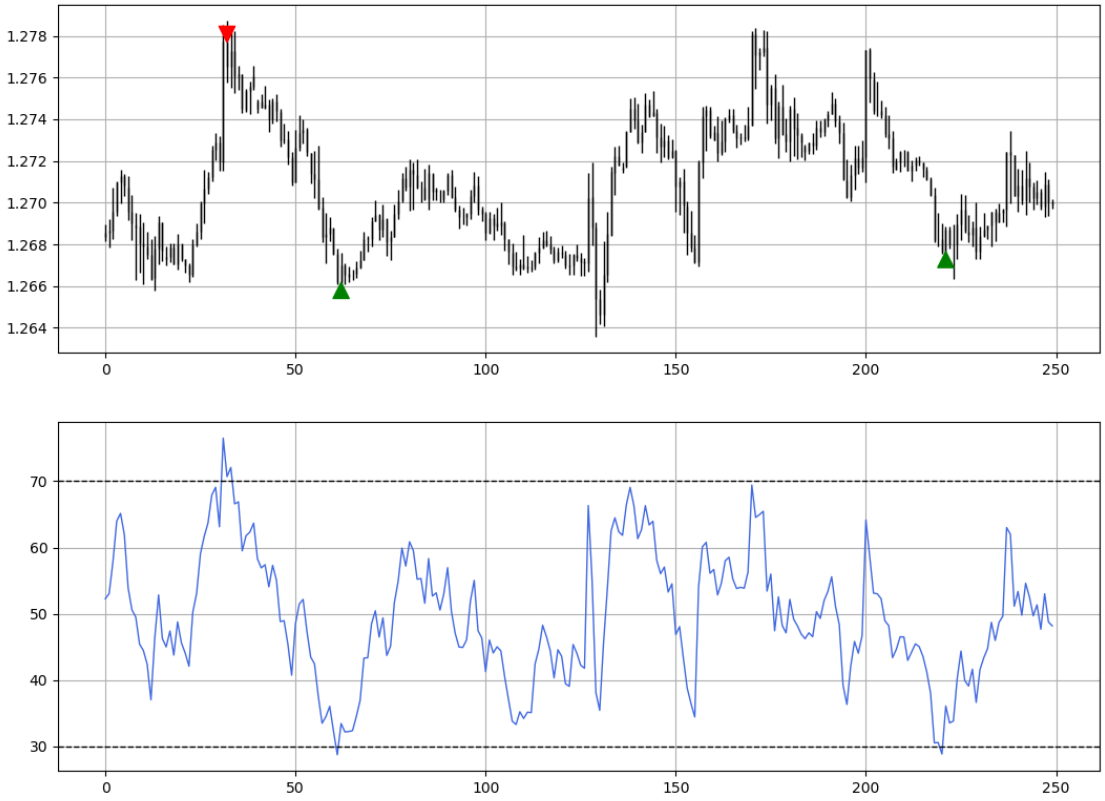


Figure 1-2 USDCAD hourly values with the 14-period RSI.

Remember, to code the techniques you need to follow the below framework:

- Call the indicator's function so that it is added to the OHLC array.
- Call the signal function of the conditions so that they are applied.
- Chart the signals and interpret them.

The performance metrics will be seen later in the strategies chapter since you will use them for evaluation, at the moment, I will not evaluate the techniques but rather code them and discuss their particularities.

The advantage of this technique is that it is heavily used by traders as it is supposed to be the default way of using this indicator. It is recommended in a

ranging market but highly discouraged in a trending market. For example, you would not use this technique to find selling opportunities on the daily values of the S&P500, a generally bullish market as the odds are against you, however, you could use it to find buying opportunities in what is called buying the dips activity[18]. The main disadvantage of the technique is that oversold and overbought conditions may persist while the market continues its initial move without any reaction. Even though the correlation is high between the underlying market and its RSI, it does not mean that an oversold RSI must signal an imminent rise in the market. This is due to the varying magnitudes and sensitivity of the market relative to its RSI.

In conclusion, an RSI around the extreme levels does not guarantee a reaction and back-tests have proven that this technique may only be used as a confirmation factor. The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter[19].
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

[18] Buying the dips in a bullish trend is the act of waiting for small corrections to buy at a cheaper price and ride the trend higher.
[19] A trend filter refers to the first step where buy signals are only taken in a bullish market and sell signals are only taken in a bearish market.

Figure 1-3 shows a number of sell signals in a bullish market. Notice the low quality due to the absence of the invisible hand of the trend that gives the extra probabilistic push towards a profitable trade.

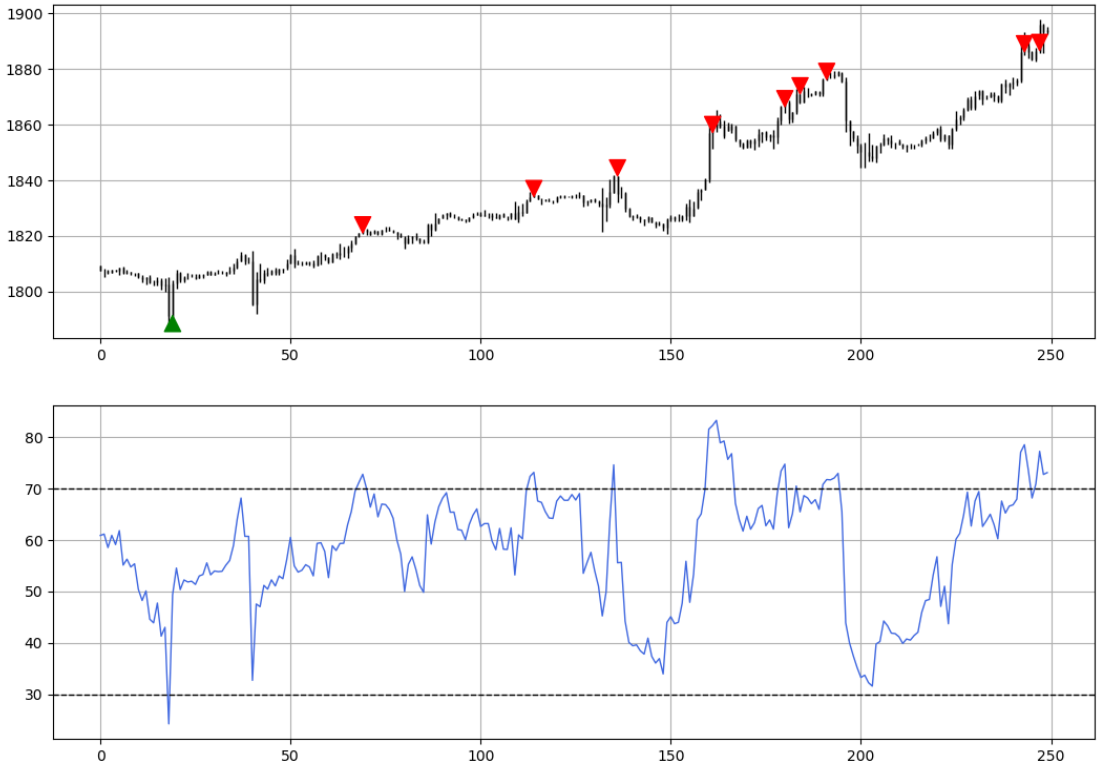


Figure 1-3 Gold hourly values with the 14-period RSI.

If you look closely at the previous chart, you can notice that whenever the RSI approaches the oversold level ~ 30, the market bounces. This is why you need the invisible hand to help you time the entries. Remember the previous list where I have mentioned that the odds are greatly in your favor when you initiate buy signals in a bullish market, this is a clear example that you need to have as many allies as you can on your side to maximize the gain probability.

SECTION 1.2 THE CONSERVATIVE TECHNIQUE

As opposed to the first technique, this one is more conservative and awaits the first hint of return to normality before initiating a trade. Traders typically prefer this method as oversold and overbought conditions may persist. Ideally, this should be the default way of using the RSI. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the RSI exits the oversold level.**
- **A short signal is generated whenever the RSI exits the overbought level.**

The signal function for this technique is as follows:

```
def signal(data, rsi_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, rsi_column] > lower_barrier and data[i - 1, rsi_column] < lower_barrier:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, rsi_column] < upper_barrier and data[i - 1, rsi_column] > upper_barrier:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 1-4 shows a sample of signals on EURUSD using the conservative technique.

Figure 1-4 EURUSD hourly values with the 14-period RSI.

The advantage of this technique is that it is also followed by traders. It is recommended in a ranging market but highly discouraged in a trending market. The main disadvantage of the technique is that oversold and overbought conditions may persist while the market continues its initial move. Even though the exit from the extreme levels may seem to remedy this, you should know that the RSI might go back to the extreme levels directly. For instance, to find a buy signal, you are looking for an RSI that is above 30 while the previous value was below 30, however, upon entering, you notice that the RSI dips back below 30 and stays there. This is an example of a common

occurrence. The technique has the following biases that impact its predictability which are naturally similar to the first technique:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

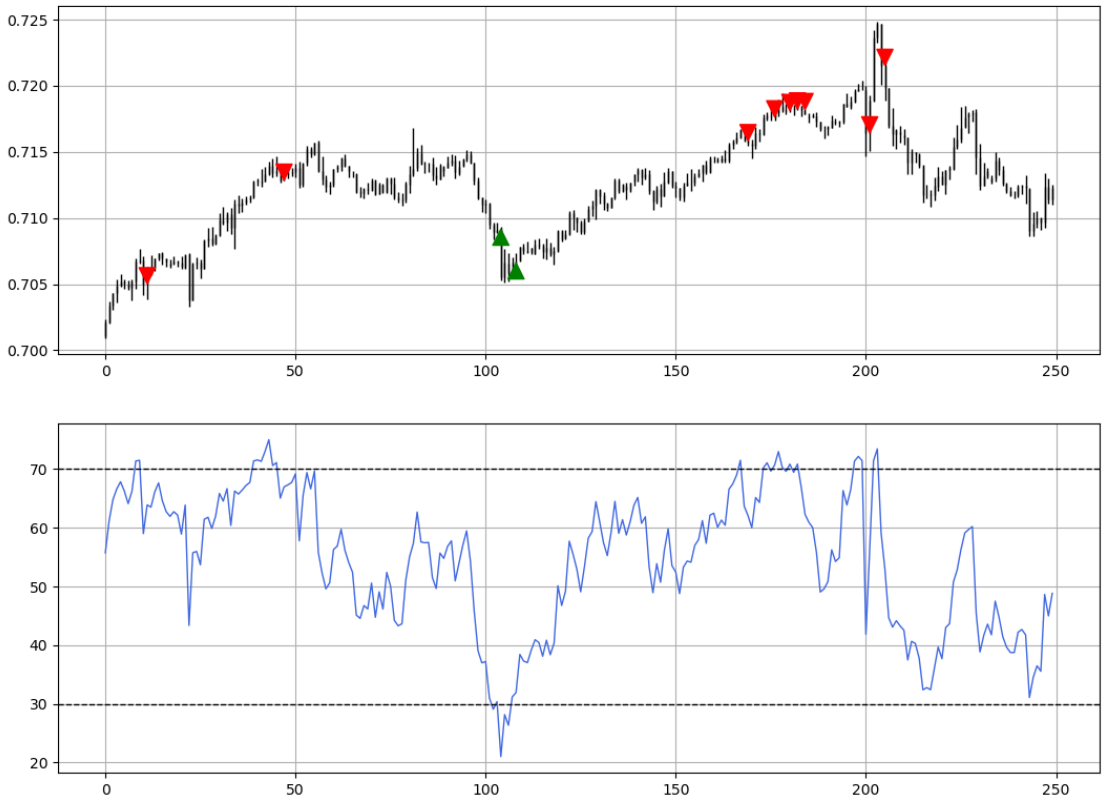


Figure 1-5 AUDUSD hourly values with the 14-period RSI.

SECTION 1.3 THE DIVERGENCE TECHNIQUE

Trends are finite regimes and divergences may hint to a possible exhaustion and return to a sideways market. This is a famous technique used by many traders to detect a change in trend and a possible reversal. Thus, the divergence technique has been designed to improve the conviction in the case of a trending market, but it is not necessarily effective in that market regime. The conditions for this technique are as follows:

- **A long signal is generated whenever a bullish divergence is spotted. This occurs when noticing lower market prices with higher indicator readings.**
- **A short signal is generated whenever a bearish divergence is spotted. This occurs when noticing higher market prices with lower indicator readings.**

The signal function of the divergence technique is lengthy which is why I have decided it to only put it in the GitHub repository. You can access it through the link[20] or through the below QR code.

[20] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

In detail, a bullish divergence occurs when there are two local market bottoms where the second one is lower than the first one. Simultaneously, two local bottoms on the RSI must appear in parallel to the ones seen on the market price. On the other hand, the second bottom on the RSI must be higher than the first one. The two bottoms on the RSI must be below 30 or a pre-determined oversold level.

Similarly, a bearish divergence occurs when there are two local market tops where the second one is higher than the first one. Simultaneously, two local tops on the RSI must appear in parallel to the ones seen on the market price. On the other hand, the second top on the RSI must be lower than the first one. The two tops on the RSI must be above 70 or a pre-determined oversold level. The next chart shows a buy signal generated following a bullish divergence.

The advantage of this technique is that it is heavily followed by traders. It is recommended in both ranging and trending markets but more confirmation from other techniques remains necessary. The main disadvantage of the technique is that it is very subjective as the interpretation of a divergence differs from one trader to another. You also need to be aware of the width concept. The width is the maximum number of bars between the two troughs (bullish divergence) or tops (bearish divergence).

In other words, the greater the width, the more and bigger divergences the algorithm will detect. I generally set this variable to no less than 40-60.

Figure 1-6 EURUSD hourly values with the 14-period RSI.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market. The signals may be infrequent.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are in your favor if you use buy and sell signals in a trending market with no trend filter.
- The odds are against you if you use buy signals in a bearish market and sell signals in a bullish market.

Figure 1-7 Gold hourly values with the 14-period RSI.

SECTION 1.4 THE NEUTRALITY REVERSAL TECHNIQUE

The RSI is a normalized calculation bounded between 0 and 100, therefore, values below 50 hint to an on-going bearish momentum while values above 50 hint to an on-going bullish momentum. Hence, you can assume that 50 is a pivot level from where you can expect reactions.

Generally, this level is used in tandem with oversold and overbought levels (as a target). Its objectivity is its most powerful trait since it is the only fixed level as opposed to the oversold and overbought levels which can change due to the trader's preferences.

The trading conditions for this technique are as follows:

- **A long signal is generated whenever the RSI approaches 50 from the above. The technique assumes that the market will find support.**
- **A short signal is generated whenever the RSI approaches 50 from the below. The technique assumes that the market will find resistance.**

The signal function for this technique is as follows:

```
def signal(data, rsi_column, tolerance, buy, sell):
        data = add_column(data, 10)
        for i in range(len(data)):
                # Bullish signal
                if data[i, rsi_column] >= 50 and data[i - 1, rsi_column] <= 50 + tolerance and \
                  data[i - 1, rsi_column] >= data[i, rsi_column]:
                        data[i + 1, buy] = 1
                # Bearish signal
                if data[i, rsi_column] <= 50 and data[i - 1, rsi_column] >= 50 - tolerance and \
                  data[i - 1, rsi_column] <= data[i, rsi_column]:
                        data[i + 1, sell] = -1
        return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, the variable tolerance refers to the margin allowed between neutrality and the current RSI reading, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 1-8 shows the signals generated on EURUSD.

Figure 1-8 EURUSD hourly values with the 14-period RSI.

The advantage of this technique is that it is objective as it takes the middle line of the RSI as a trigger. The main disadvantage of the technique is that it is not used by traders generally and hence its predictability is hindered. Also, it does not give enough margin to play a trade since the RSI is squeezed between the middle line and the extremes in case the exit is set to the RSI's values.

The signals may be too frequent and noisy. The technique has the following biases that impact its predictability:

- The odds are in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are against you if you use buy and sell signals in a ranging market due to the frequent signals with no help from the trend.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market due to the frequency of signals and the trend obstacle.

Figure 1-9 GBPUSD hourly values with the 14-period RSI.

SECTION 1.5 THE EXTREME DURATION TECHNIQUE

The aggressive technique has a flaw in that there is a risk of a sticky RSI around oversold and overbought levels. This means that the RSI can remain oversold or overbought for a long time without seeing any reaction from the market. This is where the extreme duration technique enters, it is simply the exit from the oversold or overbought areas after a minimum number of time periods spent on the extremes.

The default number of time periods is 5, meaning that an RSI that surpasses 30 after having been below it for at least 5 consecutive time periods will give a buy signal. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the RSI stays oversold for at least 5 time periods before exiting on the upside.**
- **A short signal is generated whenever the RSI stays overbought for at least 5 time periods before exiting on the downside.**

Even though the default minimum time periods number is set to 5, it should not prevent you from experimenting with other variables. For example, a 14-period RSI with a minimum number of 8 time periods might give out extended signals but they can be quite rare.

Of course, to remedy this infrequency problem, you can always adjust the oversold and overbought levels the way you want them. It is important not to fall into the trap of overfitting by tweaking the parameters to a certain point that the back-tests outperform in such a way that it is impossible to reproduce in the future.

The signal function for this technique is as follows:

```
def signal(data, rsi_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, rsi_column] > lower_barrier and data[i - 1, rsi_column] < lower_barrier and \
          data[i - 2, rsi_column] < lower_barrier and data[i - 3, rsi_column] < lower_barrier and \
          data[i - 4, rsi_column] < lower_barrier and data[i - 5, rsi_column] < lower_barrier:
            data[i + 1, buy] = 1
        # Bearish signal
        if data[i, rsi_column] < upper_barrier and data[i - 1, rsi_column] > upper_barrier and \
          data[i - 2, rsi_column] > upper_barrier and data[i - 3, rsi_column] > upper_barrier and \
          data[i - 4, rsi_column] > upper_barrier and data[i - 5, rsi_column] > upper_barrier:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

The advantage of this technique is that it takes into account extended oversold and overbought conditions thus lowering the risk of stickiness. It is recommended in a ranging market but discouraged in a trending market. The main disadvantage of the technique is that oversold and overbought conditions may persist while the market continues its initial move.

Figure 1-10 EURUSD hourly values with the 14-period RSI.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

Figure 1-11 S&P500 hourly values with the 14-period RSI.

SECTION 1.6 THE CROSS TECHNIQUE

To understand this technique, you must refer to the concept of moving averages that you have previously seen. When a certain variable is above its moving average, it means that its current momentum is stronger than average and when it is below its moving average, it means that its current momentum is weaker than average. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the RSI surpasses its moving average while being below 50.**
- **A short signal is generated whenever the RSI breaks its moving average while being above 50.**

Note that you are applying the moving average function on the values of the RSI and not on the market prices. Again, do not mind the code and try to understand it and not copy it as you will be able to easily fetch the full version from the GitHub and apply it.

The signal function for this technique is as follows:

```
def signal(data, rsi_column, ma_column, buy, sell):
        data = add_column(data, 10)
        for i in range(len(data)):
                # Bullish signal
                if data[i, rsi_column] > data[i, ma_column] and data[i - 1, rsi_column] < data[i - 1,
                  ma_column] and data[i, rsi_column] < 50:
                        data[i + 1, buy] = 1
                # Bearish signal
                elif data[i, rsi_column] < data[i, ma_column] and data[i - 1, rsi_column] > data[i - 1,
                    ma_column] and data[i, rsi_column] > 50:
                        data[i + 1, sell] = -1
        return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, the variable ma_column refers to the column of the moving average, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 1-12 USDCHF hourly values with the 14-period RSI.

The advantage of this technique is that it is used by some traders, and it takes into account the averaging technique which is regarded as a general value-adder. The main disadvantage of the technique is that the RSI can surpass and break its moving average many times before establishing a clear directional and it may not even establish a clear one. This unfortunately, gives out many false signals.

The technique has the following biases that impact its predictability:

- The odds are in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are harshly against you if you use buy and sell signals in a ranging market due to lag.
- The odds are harshly against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

Figure 1-13 GBPUSD hourly values with the 14-period RSI.

SECTION 1.7 THE V TECHNIQUE

Quick rejection is a sign of recognition of a certain level or area by the market. Therefore, you need to be vigilant and opportunistic when it comes to such events. An example of a quick rejection is a market dramatically reversing below its descending trend line thus confirming the bias of the bears[21] around the resistance area. The V technique uses a similar approach to the conservative technique seen in section 1.2; however, the only difference is that the V technique is paradoxically the aggressive version of the conservative technique. The trading conditions are as follows:

- **A long signal is generated whenever the RSI surpasses the oversold level after having been below it for only one period thus forming a V shape.**
- **A short signal is generated whenever the RSI breaks the overbought level after having been above it for only one period thus forming an inverted V shape.**

The bullish version of this technique resembles a small V letter while the bearish version resembles an inverted V letter. I generally prefer low lookback periods on the indicator when using this technique, for example, a 5-period RSI.

Important

The lookback periods used in the techniques presented are only for illustration purposes. You should find your own preferred lookback period that fits your profile and time horizon.

[21] Bears refer to short sellers. They are traders who bet that the value of the underlying will depreciate.

The signal function for this technique is as follows:

```
def signal(data, rsi_column, buy, sell):
        data = add_column(data, 10)
        for i in range(len(data)):
                # Bullish signal
                if data[i, rsi_column] > lower_barrier and data[i - 1, rsi_column] < lower_barrier and \
                  data[i - 2, rsi_column] > lower_barrier:
                        data[i + 1, buy] = 1
                # Bearish signal
                elif data[i, rsi_column] < upper_barrier and data[i - 1, rsi_column] > upper_barrier and \
                    data[i - 2, rsi_column] < upper_barrier:
                        data[i + 1, sell] = -1
        return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 1 – 14 shows the signals generated using the V technique with barriers at 30 and 70 as outlined in the chart through the dashed black lines.

Figure 1-14 EURUSD hourly values with the 5-period RSI.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

Figure 1-15 S&P500 hourly values with the 5-period RSI.

SECTION 1.8 THE M TECHNIQUE

Old school chartists are very familiar with double tops and bottoms which signal reversals and give the potential of said reversals. Despite their simplicity, they are very subjective, and some are not visible like the others. This hinders the ability to know whether they add value or not. Luckily, it is possible to simplify the conditions using algorithms. This technique will show a simplified version of double tops and bottoms that I call the M technique. In the last part of the book, I will present a more sophisticated algorithm that scans for real double tops and bottoms.

Figure 1-16 shows a theoretical example of a double top formation.

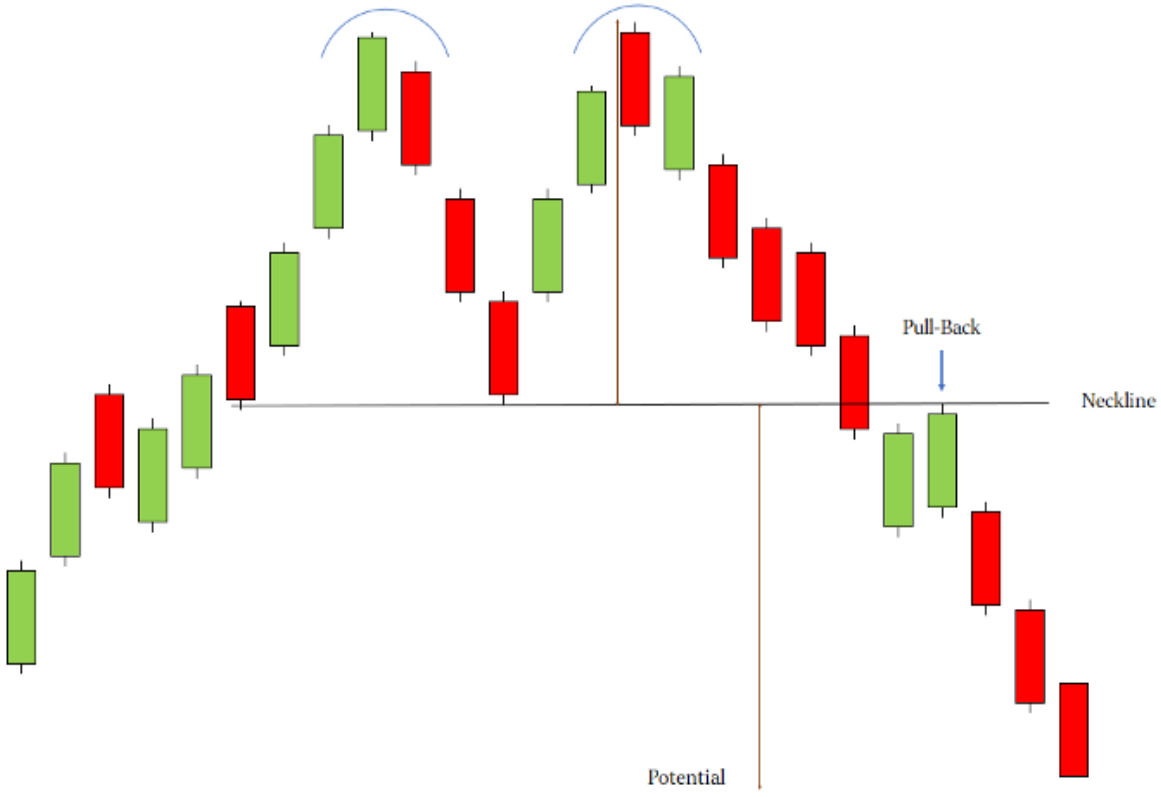


Figure 1-16 Double top configuration.

You can notice three important elements in a double top:

- **The neckline**: This is the line linking the lowest low between the two peaks and the beginning/end of the pattern. It serves to determine the pull-back.
- **The pull-back**: Having broken the neckline, the market should shape a desperate attempt towards the neckline but fails to continue higher as the sellers use this level as a re-entry to continue shorting. Therefore, the pull-back level is the optimal selling point after validating a double top.
- **The potential**: This is the target of the double top. It is measured as the same distance between the top of the pattern and the neckline projected lower and starting from the same neckline point as shown in the chart through the two arrows which have the same length.

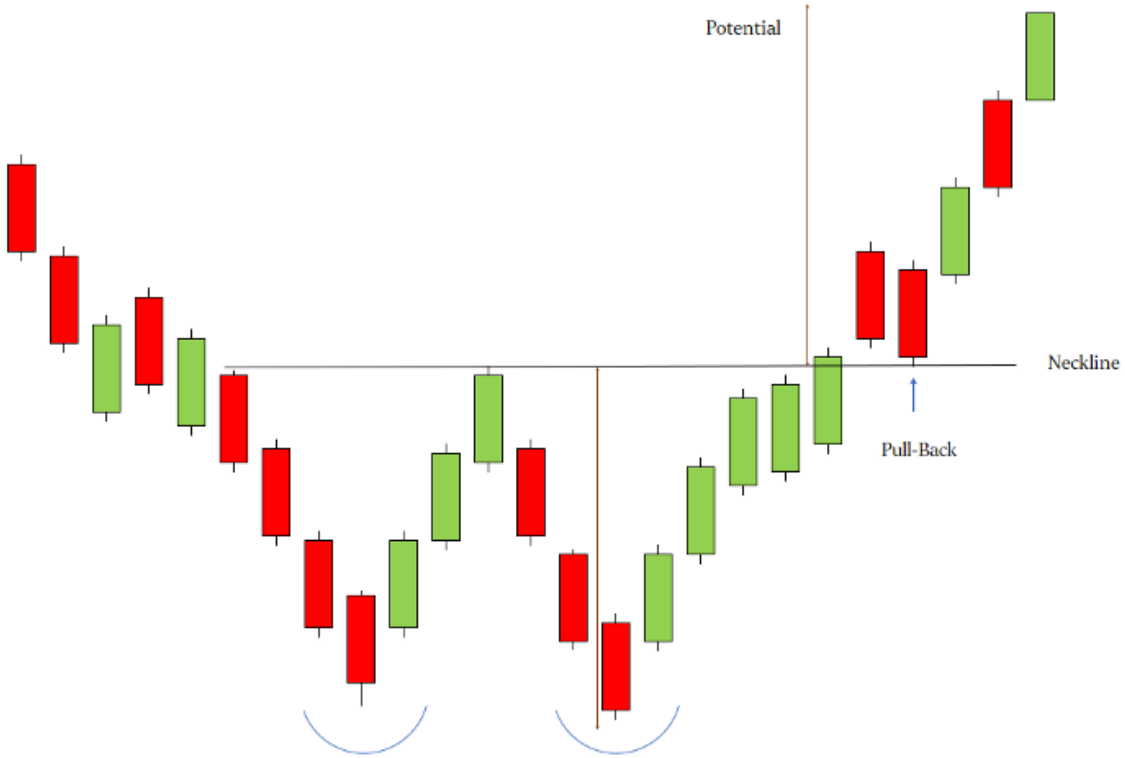


Figure 1-17 Double bottom configuration.

Similarly, the three elements present in a double bottom are:

- **The neckline**: This is the line linking the highest high between the two troughs and the beginning/end of the pattern. It serves to determine the pull-back.
- **The pull-back**: Having surpassed the neckline, the market should shape a desperate attempt towards the neckline but fails to continue lower as the buyers use this level as a re-entry to continue the move upwards.
- **The potential**: This is the target of the double bottom. It is measured as the same distance between the bottom of the pattern and the neckline projected higher and starting from the same neckline point as shown in the chart through the two arrows which have the same length.

Theoretically, the pattern's psychological explanation is that with the second top or bottom, the market has failed to push the prices beyond the first top/bottom and therefore is showing weakness which might be exploited by the other team[22]. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the RSI breaks the oversold level and surpasses it, then breaks it again while forming a bottom above or at the first bottom and then manages again to surpass the oversold level thus forming an inverted M shape.**
- **A short signal is generated whenever the RSI surpasses the overbought level and breaks it, then surpasses it again while forming a top below or at the first top and then manages again to break the overbought level thus forming an M shape.**

You might have noticed the terms break and surpass. I tend to use them as follows:

- A market **breaks** its support. This implies that breaking is a downside event.
- A market **surpasses** its resistance. This implies that surpassing is an upside event.

The important thing to keep in mind with the M technique is that the conditions are successive which means that there are no intermediary bars between them. Every condition must happen exactly one time period after the previous one. This is the difference between the M technique and the double top and bottom configurations.

[22] The other team in case of a bullish market are sellers. The other team in case of a bearish market are buyers.

The signal function for this technique is as follows:

```
def signal(data, rsi_column, buy, sell):
        data = add_column(data, 10)
        for i in range(len(data)):
                # Bullish signal
                if data[i, rsi_column] > lower_barrier and data[i - 1, rsi_column] <
                  lower_barrier and data[i - 2, rsi_column] > lower_barrier and
                  data[i - 3, rsi_column] < lower_barrier and data[i - 4, rsi_column] >
                  lower_barrier and data[i - 1, rsi_column] >= data[i - 3, rsi_column]:
                        data[i + 1, buy] = 1
                # Bearish signal
                elif data[i, rsi_column] < upper_barrier and data[i - 1, rsi_column] >
                     upper_barrier and data[i - 2, rsi_column] < upper_barrier and
                     data[i - 3, rsi_column] > upper_barrier and data[i - 4, rsi_column] <
                     upper_barrier and data[i - 1, rsi_column] <= data[i - 3, rsi_column]:
                        data[i + 1, sell] = -1
        return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

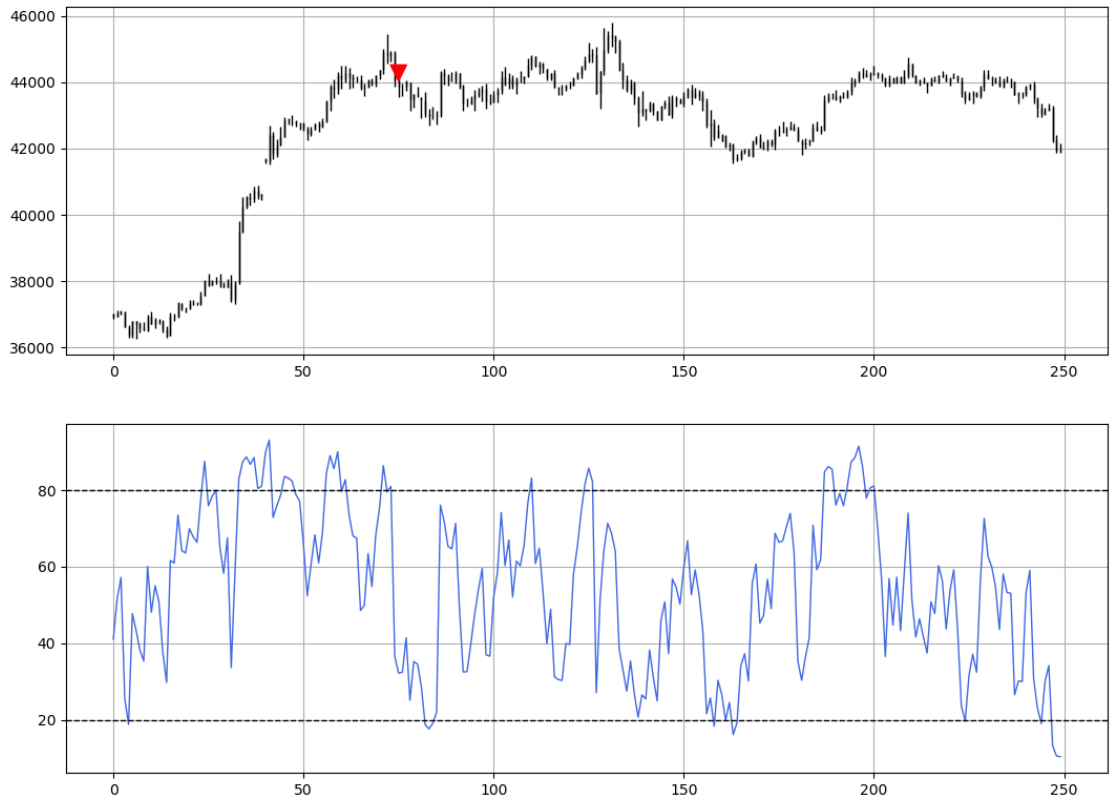


Figure 1-18 BTCUSD[23] hourly values with the 5-period RSI.

The technique has the following biases that impact its predictability:

- The odds are in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

[23] BTCUSD is the price of 1 Bitcoin in USD terms. Bitcoin is the most liquid cryptocurrency.

Figure 1-19 EURUSD hourly values with the 5-period RSI.

SECTION 1.9 THE PULL-BACK TECHNIQUE

Pull-backs are an underrated event in trading. Generally, whenever a market is trading within a range and manages to surpass the upper limit of the range, traders tend to say that there has been a bullish breakout and the market is likely to continue higher, therefore, they look to buy the breakout immediately. However, the pull-back technique states that as soon as the breakout happens, you should wait for the price to return to the broken resistance which gives the buyers an opportunity to buy at a better price. This technique will apply the pull-back method on the indicator rather on the market price.

The trading conditions for this technique are as follows:

- **A long signal is generated whenever the RSI surpasses the oversold level and pulls-back to it.**
- **A short signal is generated whenever the RSI breaks the overbought level and pulls-back to it.**

The conditions state that for a bullish signal, the RSI must emerge from its oversold level which is generally around 30 and then the algorithm must wait for a comeback towards the 30 zone before initiating a buy order. In parallel, for a bearish signal, the RSI must dive from its overbought level which is generally around 70 and then the algorithm must wait for a comeback towards the 70 zone before initiating a sell order.

Figure 1-20 Gold hourly values with the 14-period RSI.

The signal function for this technique is as follows:

```
def signal(data, rsi_column, tolerance, buy, sell):
        data = add_column(data, 10)
        for i in range(len(data)):
                if data[i, rsi_column] > lower_barrier and data[i - 1, rsi_column] < lower_barrier:
                        for a in range(i + 1, len(data)):
                                if data[a, rsi_column] >= lower_barrier and data[a, rsi_column] <
                                  lower_barrier + tolerance and data[a, rsi_column] < data[a - 1,
                                  rsi_column]:
                                        data[a + 1, buy] = 1
                                        break
                                elif data[a, rsi_column] > upper_barrier:
                                        break
                elif data[i, rsi_column] < upper_barrier and data[i - 1, rsi_column] > upper_barrier:
                        for a in range(i + 1, len(data)):
                                if data[a, rsi_column] <= upper_barrier and data[a, rsi_column] >
                                  upper_barrier - tolerance and data[a, rsi_column] > data[a - 1,
                                  rsi_column]:
                                        data[a + 1, sell] = -1
                                        break
                                elif data[a, rsi_column] < lower_barrier:
                                        break
        return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, the variable tolerance refers to the margin allowed in the pull-back, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Alternatively, I recommend you add a neutrality filter. What is the intuition of this? Let's review the conditions I have presented for a bullish pull-back:

- The market surpasses 30 after having been below it.
- The market reverts back to 30 while staying below 30 + the tolerance.

The neutrality filter I recommend adding will update the conditions to the following:

- The market surpasses 30 after having been below it.
- The market reverts back to 30 while staying below 30 + the tolerance.
- Simultaneously, all the previous conditions must happen with an RSI below 50.

The neutrality filter prevents the RSI from going too high before shaping a pull-back which is a reasonable thing to do. I have not made this filter as it is an enhancement, and it would be a good practice exercise if you add it yourself into the original signal function. The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

Take a look at Figure 1 – 21 and try to analyze why certain obvious signals have not been generated by the algorithm.

Figure 1-21 USDCAD hourly values with the 14-period RSI.

SECTION 1.10 THE EXTREME-TO-EXTREME TECHNIQUE

When a market goes from one extreme to another, it is likely an overreaction and it should return to normality soon. This is the hypothesis on which the technique is based. Naturally, due to the hypothesis, you must use a very short-term RSI. Unlike the default 14-period RSI which does not go from an oversold level to an overbought level and vice versa in one time period, I will be using a 2-period RSI which does this more frequently.

The conditions for this technique are as follows:

- **A long signal is generated whenever the RSI is below the oversold level with the previous value above the overbought level.**
- **A short signal is generated whenever the RSI is above the overbought level with the previous value below the oversold level.**

The signal function for this technique is as follows:

```
def signal(data, rsi_column, buy, sell):
        data = add_column(data, 10)
        for i in range(len(data)):
                # Bullish signal
                if data[i, rsi_column] < lower_barrier and data[i - 1, rsi_column] > upper_barrier:
                        data[i + 1, buy] = 1
                # Bearish signal
                elif data[i, rsi_column] > upper_barrier and data[i - 1, rsi_column] < lower_barrier:
                        data[i + 1, sell] = -1
        return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 1-22 EURUSD hourly values with the 2-period RSI.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

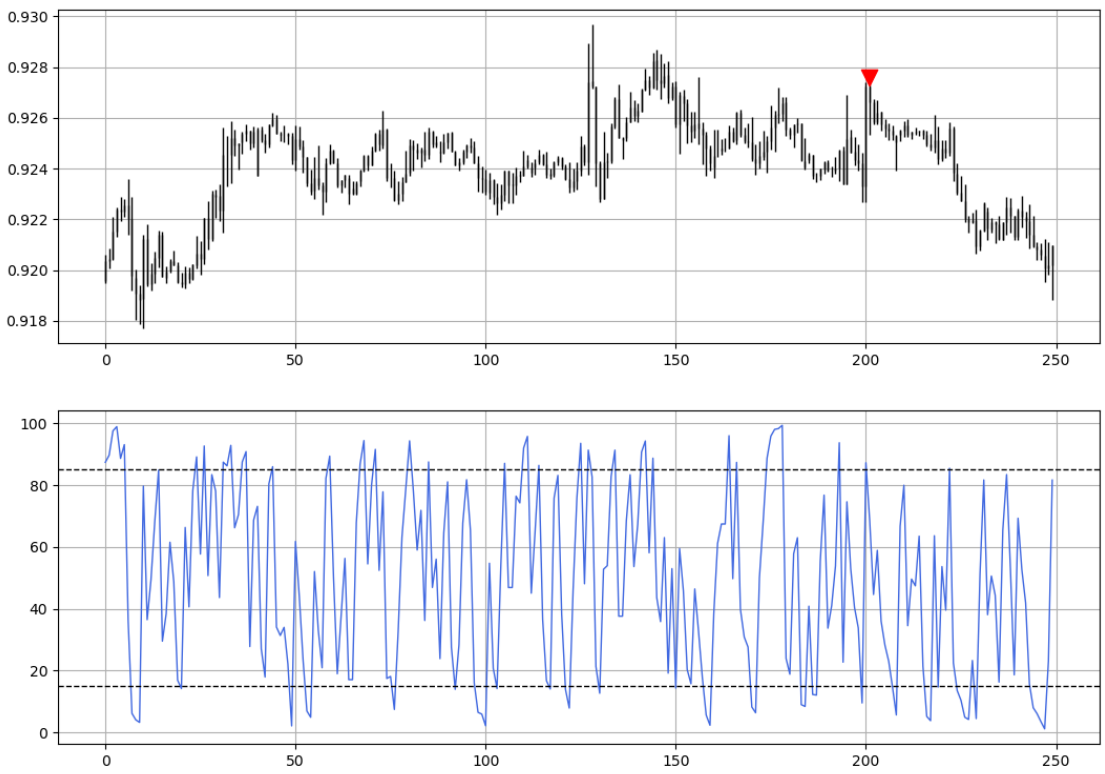


Figure 1-23 USDCHF hourly values with the 2-period RSI.

To summarize, the RSI is a versatile technical indicator that can be used in many ways. Traders mainly like the RSI for the following reasons:

- The RSI is a bounded indicator which facilitates interpretation.
- The RSI is monitored by the majority of the trading world which makes it more impactful.
- There are many techniques that can be used with the RSI.
- The RSI can be easily combined with other technical indicators to create strategies.

CHAPTER 2 THE STOCHASTIC OSCILLATOR

Any set of values can be normalized between two pre-specified barriers so that you get an idea on the scale of the changes. Take for example the following chronologically ordered list {12, 45, 33, 9, 21, 22, 8, 54, 15, 19, 4, 2, 1, 10, 11}. What if you want to normalize the list so that its values become stuck between 5 and 15? What would this imply and why would you do this? To answer the first question, this would imply that the lowest number in the original list becomes 5 while the highest number becomes 15. Consequently, the number in the middle would be around 10. The reason you want to do this is to understand when the values are around the extremes. Let's see the normalization formula that allows you to do this:

$$x_{normalized} = \frac{x_{original} - x_{low}}{x_{high} - x_{low}}(maximum - minimum) + minimum$$

The following bullet list illustrates the elements of the above formula:

- The normalized x is the output from using the formula.
- The original x is the original number of the list that will be transformed.
- The low x is the lowest value of the original x for a set lookback period
- The high x is the highest value of the original x for a set lookback period.
- The maximum is the upper bound you are seeking.
- The minimum is the lower bound you are seeking.

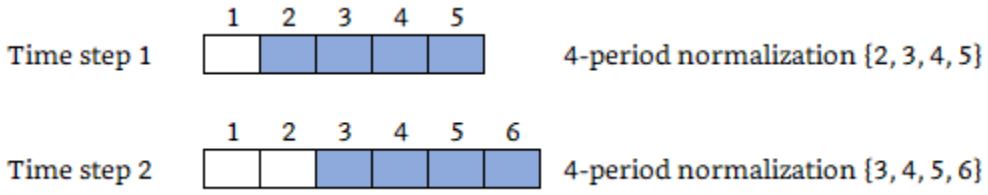
Therefore, given the previous list, when you want to normalize it between 5 and 15, you will find this new transformed list {7.08, 13.30, 11.01, 6.51, 8.77, 8.96, 6.32, **15.00**, 7.64, 8.40, 5.57, 5.19, **5.00**, 6.70, 6.89}. Note how the values in bold represent the extremes {1;5.00, 54;15.00}.

It is better to normalize the values to a known range that is understood by most and interpreted easily. This range I am talking about is between 0 and 1 with values close to 0 representing the lows and values close to 1 representing the highs. Something interesting happens when applying the normalization formula to the {0, 1} interval:

$$x_{normalized} = \frac{x_{original} - x_{low}}{x_{high} - x_{low}}(1 - 0) + 0$$

$$x_{normalized} = \frac{x_{original} - x_{low}}{x_{high} - x_{low}}$$

It becomes much simpler now to apply normalization between 0 and 1. The list from the first example becomes as follows when normalizing it between 0 and 1 {0.21, 0.83, 0.60, 0.15, 0.38, 0.40, 0.13, **1.00**, 0.26, 0.34, 0.06, 0.02, **0.00**, 0.17, 0.19}. With time series, you should apply normalization using a moving window. For example, when you want to normalize the last 20 periods, you will use the formula on the last 20 periods every time you advance one time step.



As the previous chart shows, each time you move forward in time, you will always calculate the latest bars of a specified number by dropping the first one and adding the most recent one. The stochastic oscillator normalizes the data so that it becomes bounded between 0 and 100 by incorporating the highs and lows into the formula.

Follow these steps to calculate the stochastic oscillator:

- Apply the below special normalization function[24] for the last specified period:

$$Stochastic\ Value_i = \frac{Close_i - Low_{i-n:i}}{High_{i-n:i} - Low_{i-n:i}}$$

- Slow the stochastic value using a 3-period moving average. This is called %K.
- Further smooth out the values with another 3-period moving average applied on the slowed values from the previous step. This is called %D.

The stochastic oscillator is therefore three calculations but only the two last calculations are retained. Most charting software show %K and %D.

[24] I use i-n:i to specify the range with i as the current index (time step) and n as the lookback period. Hence, i-n:i represents the range between the current index and n time steps to the past.

Figure 2-1 shows the values of the hourly AUDUSD values charted with the 14-period stochastic oscillator.

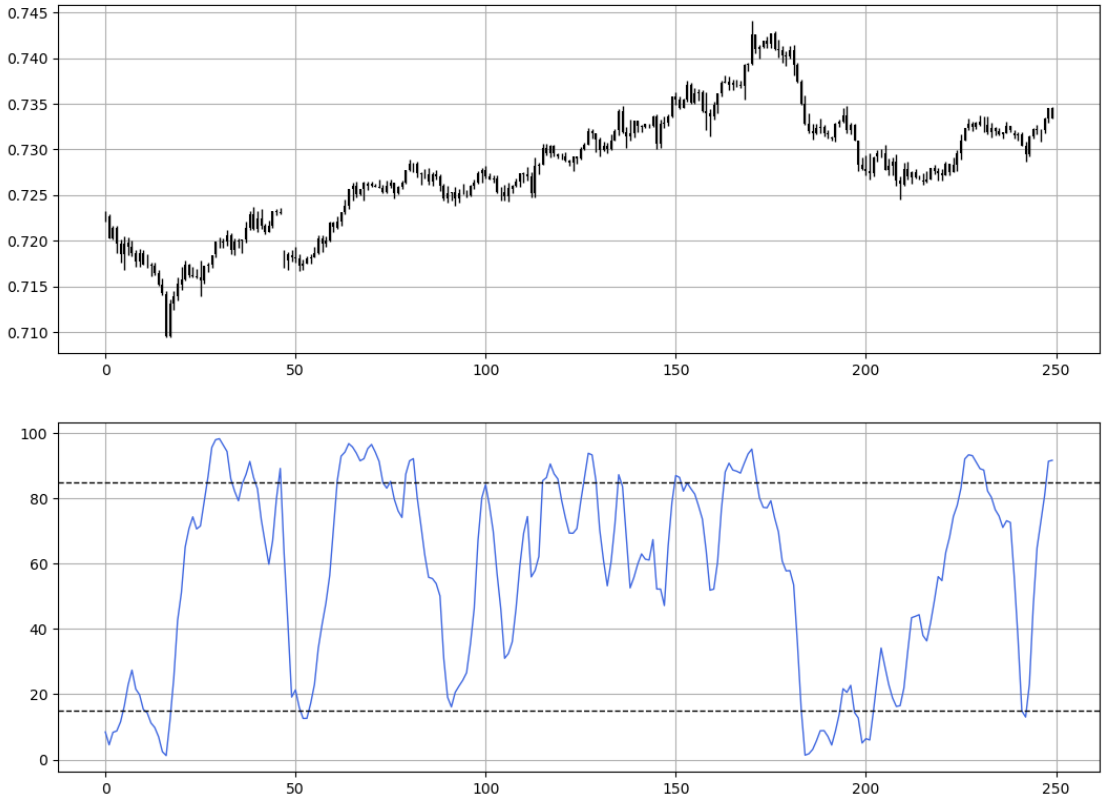


Figure 2-1 AUDUSD hourly values with the 14-period stochastic oscillator.

Note that a 14-period stochastic oscillator here implies that you use 14 as the lookback period in the first formula and then slow it down with a 3-period moving average. The next code snippet shows the full function of the stochastic oscillator.

```
def stochastic_oscillator(data, lookback, high, low, close, position, slowing = False,  smoothing = False,
                          slowing_period = 1, smoothing_period = 1):
    data = add_column(data, 1)
    for i in range(len(data)):
        try:
            data[i, position] = (data[i, close] - min(data[i - lookback + 1:i + 1, low])) /
                                (max(data[i - lookback + 1:i + 1, high]) - min(data[i - lookback +
                                1:i + 1, low]))
        except ValueError:
            pass
    data[:, position] = data[:, position] * 100
    if slowing == True and smoothing == False:
        data = ma(data, slowing_period, position, position + 1)
    if smoothing == True and slowing == False:
        data = ma(data, smoothing_period, position, position + 1)
    if smoothing == True and slowing == True:
        data = ma(data, slowing_period, position,   position + 1)
        data = ma(data, smoothing_period, position + 1, position + 2)
    data = delete_row(data, lookback)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable close refers to the column of the close price, the variable position refers to the column where you place the indicator, the variable slowing variable is a Boolean (meaning that it has a binary value of true or false) which activates a

condition in the function that calculates the first moving average, the variable smoothing is also a Boolean which activates a condition in the function that calculates the second moving average, the variable slowing_period refers to the number of time periods to consider in the calculation of the first moving average, and the variable smoothing_period refers to the number of time periods to consider in the calculation of the second moving average. Let's have a look at the techniques.

SECTION 2.1 THE AGGRESSIVE TECHNIQUE

The aggressive technique deals with the event where the indicator reaches[25] the extreme level. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the stochastic oscillator reaches the oversold level.**
- **A short signal is generated whenever the stochastic oscillator reaches the overbought level.**

```
def signal(data, stoch_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, stoch_column] < lower_barrier and data[i - 1, stoch_column] > lower_barrier:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, stoch_column] > upper_barrier and data[i - 1, stoch_column] < upper_barrier:
            data[i + 1, sell] = -1
    return data
```

[25] As opposed to the conservative technique which deals with the event where the indicator exits the extreme level.

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable stoch_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 2-2 EURUSD hourly values with the 14-period stochastic oscillator.

The advantage of this technique is that it is heavily used by traders as it is supposed to be the default way of using this indicator. It is recommended in a ranging market but highly discouraged in a trending market. The main disadvantage of the technique is that oversold and overbought conditions may persist while the market continues its initial move without any reaction.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter[26].
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

Figure 2-3 ETHUSD[27] hourly values with the 14-period stochastic oscillator.

[26] A trend filter refers to the first step where buy signals are only taken in a bullish market and sell signals are only taken in a bearish market.
[27] ETHUSD is the price of 1 Ethereum in USD terms. Ethereum is the second largest and popular cryptocurrency.

You can notice that the sell signals in Figure 2-3 triggered at the beginning of the bullish trend were not of high quality while both buy and sell signals in the ranging regime that occurred after are relatively of better quality.

SECTION 2.2 THE CONSERVATIVE TECHNIQUE

Traders typically prefer this method as oversold and overbought conditions may persist especially with the stochastic oscillator which is known to be sticky[28] in nature. Ideally, this should be the default way of using the stochastic. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the stochastic oscillator exits the oversold level.**
- **A short signal is generated whenever the stochastic oscillator exits the overbought level.**

The signal function for this technique is as follows:

```
def signal(data, stoch_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, stoch_column] > lower_barrier and data[i - 1, stoch_column] < lower_barrier:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, stoch_column] < upper_barrier and data[i - 1, stoch_column] > upper_barrier:
            data[i + 1, sell] = -1
    return data
```

[28] The Stochastic oscillator tends to stick around the extremes more than the other indicators due to the way it is calculated.

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable stoch_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 2-4 S&P500 hourly values with the 14-period stochastic oscillator.

The advantage of this technique is that it is also followed by traders. It is recommended in a ranging market but highly discouraged in a trending market. The main disadvantage of the technique is that oversold and overbought conditions may persist while the market continues its initial move even though the exit from the extreme levels may seem to remedy this, the stochastic oscillator might go back to the extreme levels directly. For instance, to find a buy signal, you are looking for a stochastic that is above 15 while the

previous value being below 15, however, upon entering, you notice that the oscillator dips back below 15 and stays there. The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

Figure 2-5 Gold hourly values with the 14-period stochastic oscillator.

SECTION 2.3 THE DIVERGENCE TECHNIQUE

Divergences can also be applied on the stochastic oscillator the same way as the RSI. The conditions for this technique are as follows:

- **A long signal is generated whenever a bullish divergence is spotted. This occurs when noticing lower market prices with higher indicator readings.**
- **A short signal is generated whenever a bearish divergence is spotted. This occurs when noticing higher market prices with lower indicator readings.**

The signal function of the divergence technique is lengthy which is why I have decided it to only put it in the GitHub repository. You can access it through the link[29] or through the below QR code.

[29] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

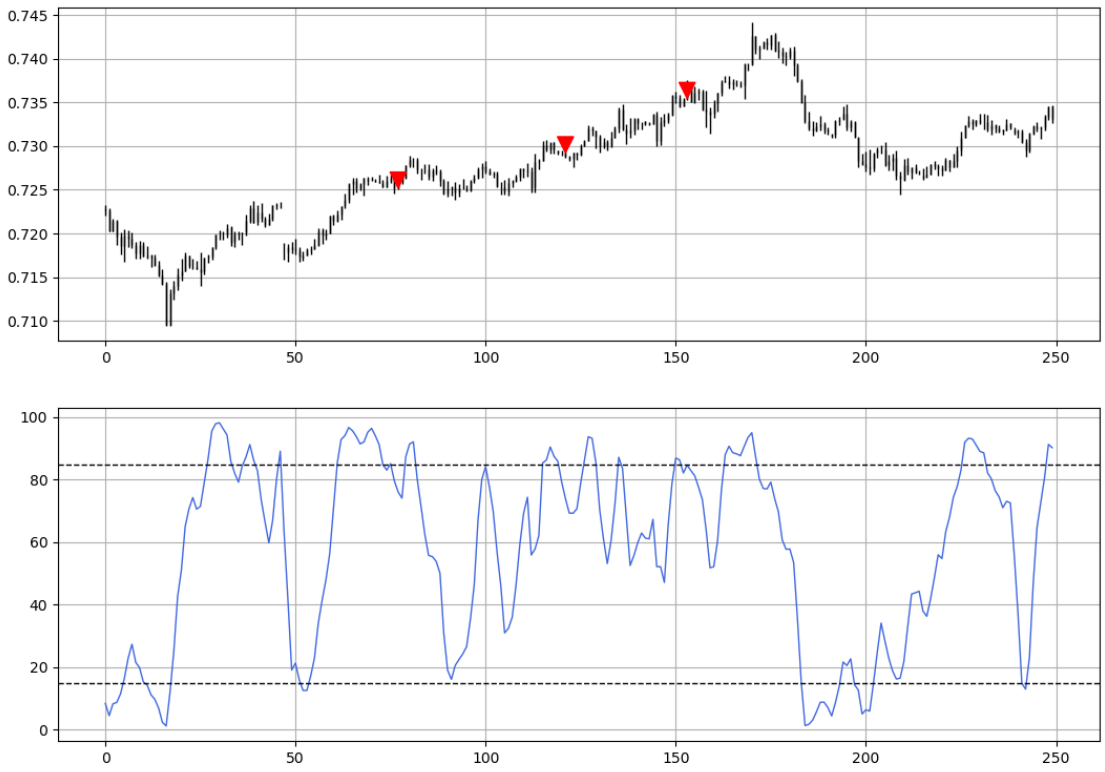


Figure 2-6 AUDUSD hourly values with the 14-period stochastic oscillator.

The advantage of this technique is that it is heavily followed by traders. It is recommended in both ranging and trending markets but more confirmation from other techniques remains necessary. The main disadvantage of the technique is that it is very subjective as the interpretation of a divergence differs from one trader to another. The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market. The signals may be infrequent.
- The odds are in your favor if you use buy and sell signals in a ranging market.

- The odds are in your favor if you use buy and sell signals in a trending market with no trend filter.
- The odds are against you if you use buy signals in a bearish market and sell signals in a bullish market.

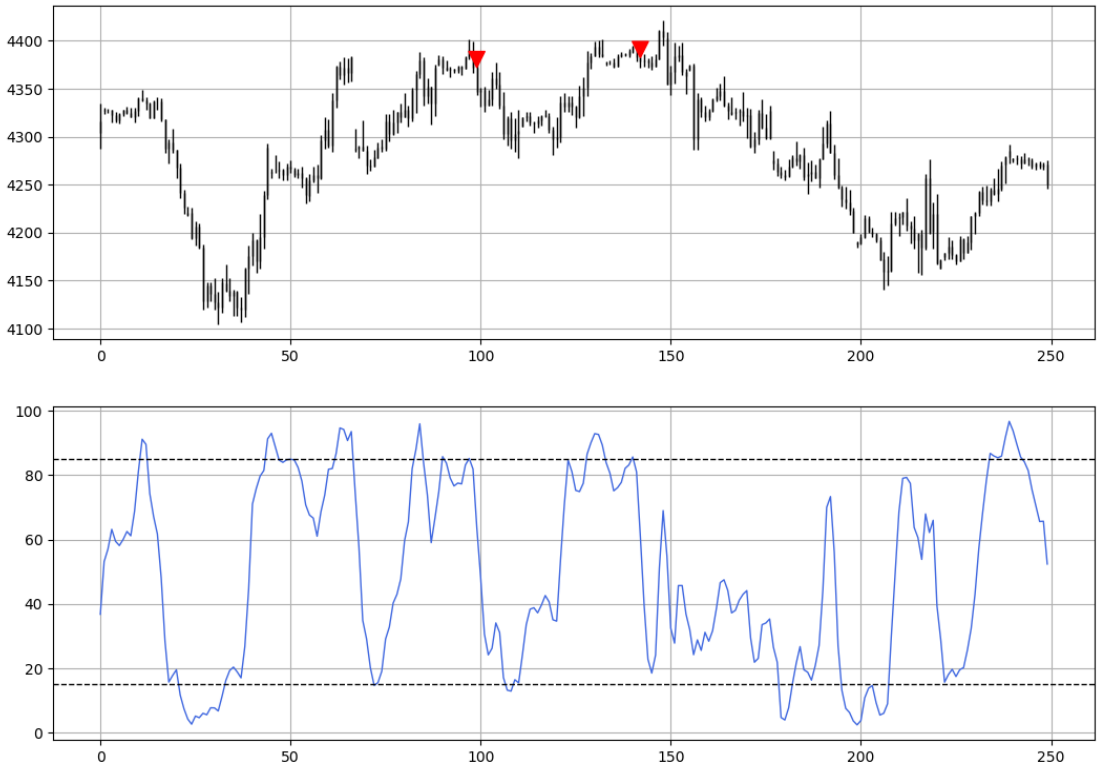


Figure 2-7 S&P500 hourly values with the 14-period stochastic oscillator.

SECTION 2.4 THE EXTREME DURATION TECHNIQUE

As previously mentioned, this technique is simply the exit of the extremes after a minimum number of time periods spent outside normality. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the stochastic oscillator stays oversold for at least 5 time periods before exiting on the upside.**
- **A short signal is generated whenever the stochastic oscillator stays overbought for at least 5 time periods before exiting on the downside.**

Even though the default minimum time periods number is set to 5, it should not prevent you from experimenting with other figures. The signal function for this technique is as follows:

```
def signal(data, stoch_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, stoch_column] > lower_barrier and data[i - 1, stoch_column] <
          lower_barrier and data[i - 2, stoch_column] < lower_barrier and
          data[i - 3, stoch_column] < lower_barrier and data[i - 4, stoch_column] <
          lower_barrier and data[i - 5, stoch_column] < lower_barrier:
              data[i + 1, buy] = 1
        # Bearish signal
        if data[i, stoch_column] < upper_barrier and data[i - 1, stoch_column] >
          upper_barrier and data[i - 2, stoch_column] > upper_barrier and
          data[i - 3, stoch_column] > upper_barrier and data[i - 4, stoch_column] >
          upper_barrier and data[i - 5, stoch_column] > upper_barrier:
              data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable stoch_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 2-8 shows the signals generated on USDCHF.

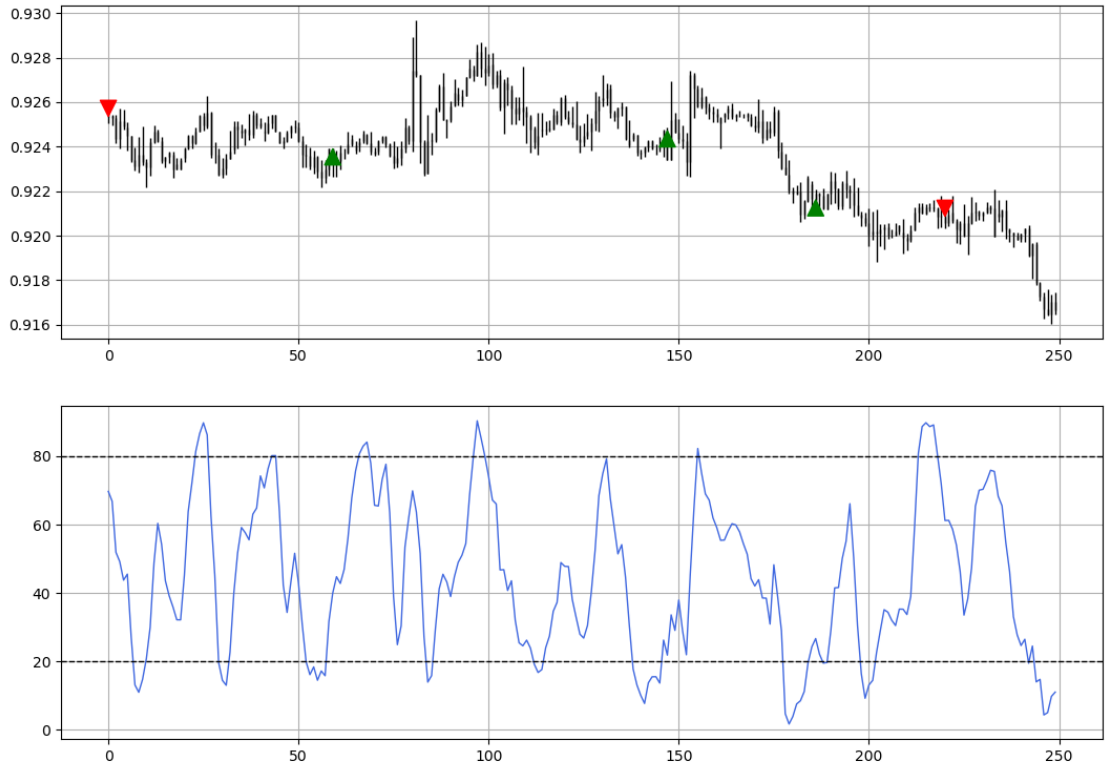


Figure 2-8 USDCHF hourly values with the 14-period stochastic oscillator.

The advantage of this technique is that it takes into account extended oversold and overbought conditions thus lowering the risk of stickiness. It is recommended in a ranging market but highly discouraged in a trending market. The main disadvantage of the technique is that oversold and overbought conditions may persist while the market continues its initial move.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

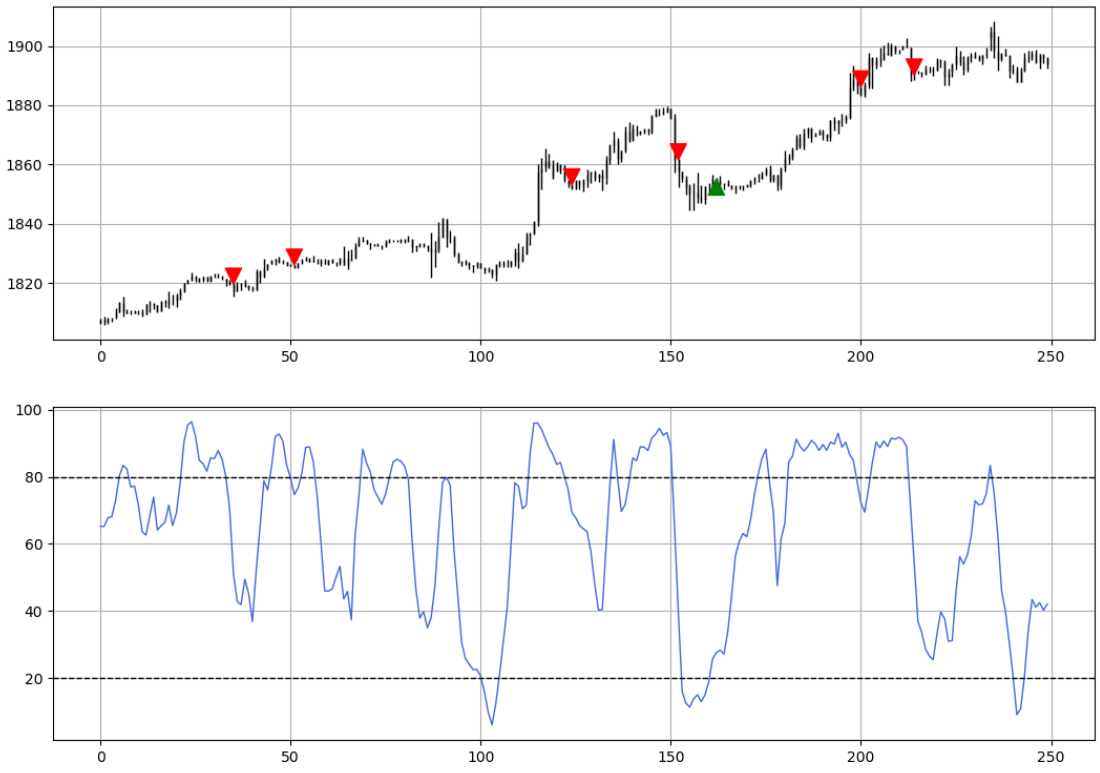


Figure 2-9 Gold hourly values with the 14-period stochastic oscillator.

SECTION 2.5 THE CROSS TECHNIQUE

The trading conditions for the cross technique are as follows:

- **A long signal is generated whenever the stochastic oscillator surpasses its moving average while being below 50.**
- **A short signal is generated whenever the stochastic oscillator breaks its moving average while being above 50.**

With the stochastic oscillator, the moving average applied onto it is called a smoothing factor. This means that generally, when you calculate a stochastic oscillator, you slow it down with a moving average a first time and then smooth it again with another moving average, thus keeping the last two calculations (the slowed and smoothed version).

```
def signal(data, stoch_column, ma_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, stoch_column] > data[i, ma_column] and data[i - 1, stoch_column] < data[i - 1,
          ma_column] and data[i, stoch_column] < 50:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, stoch_column] < data[i, ma_column] and data[i - 1, stoch_column] > data[i - 1,
            ma_column] and data[i, stoch_column] > 50:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable stoch_column refers to the column of the indicator, the variable ma_column

refers to the column of the moving average, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 2-10 EURUSD hourly values with the 14-period stochastic oscillator.

The advantage of this technique is that it is used by some traders, and it takes into account the averaging technique. The main disadvantage of the technique is that the oscillator can surpass and break its moving average many times before establishing a clear directional.

The technique has the following biases that impact its predictability:

- The odds are in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are harshly against you if you use buy and sell signals in a ranging market due to lag.
- The odds are harshly against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

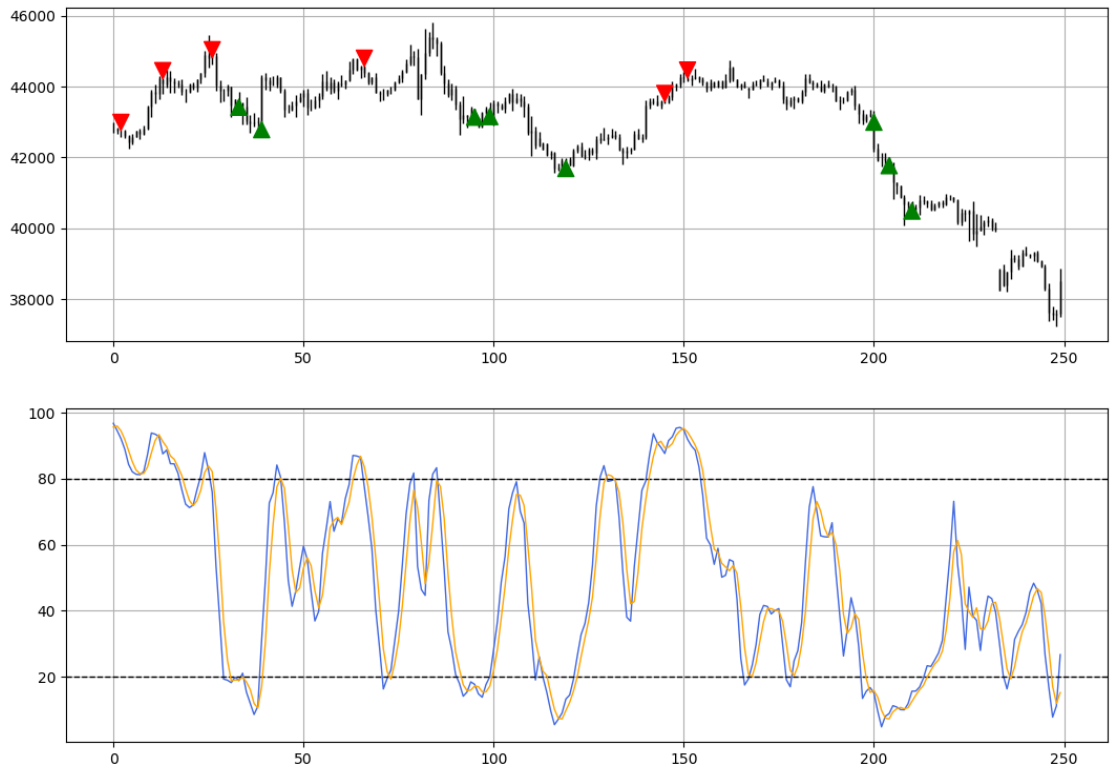


Figure 2-11 BTCUSD hourly values with the 14-period stochastic oscillator.

SECTION 2.6 THE M TECHNIQUE

As discussed previously, the M technique is a simplistic version of the double top and bottom patterns. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the stochastic oscillator breaks the oversold level and surpasses it, then breaks it again while forming a bottom above or at the first bottom and then manages again to surpass the oversold level thus forming an inverted M shape.**
- **A short signal is generated whenever the stochastic oscillator surpasses the overbought level and breaks it, then surpasses it again while forming a top below or at the first top and then manages again to break the overbought level thus forming an M shape.**

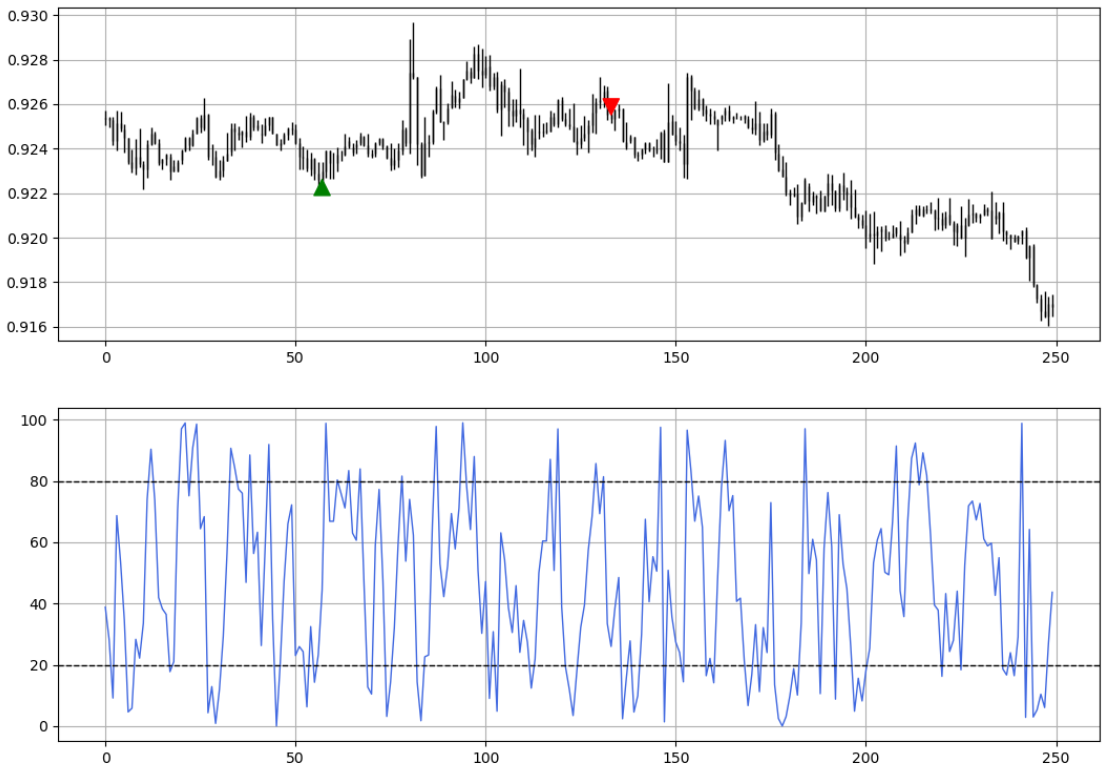


Figure 2-12 USDCHF hourly values with the 5-period stochastic oscillator.

The signal function for this technique is as follows:

```
def signal(data, stoch_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, stoch_column] > lower_barrier and data[i - 1, stoch_column] < lower_barrier and \
          data[i - 2, stoch_column] > lower_barrier and data[i - 3, stoch_column] < lower_barrier
          and data[i - 4, stoch_column] > lower_barrier and data[i - 1, stoch_column] >= data[i - 3,
          stoch_column]:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, stoch_column] < upper_barrier and data[i - 1, stoch_column] > upper_barrier and
            data[i - 2, stoch_column] < upper_barrier and data[i - 3, stoch_column] > upper_barrier
            and data[i - 4, stoch_column] < upper_barrier and data[i - 1, stoch_column] <= data[i - 3,
           stoch_column]:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable stoch_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

The technique has the following biases that impact its predictability:

- The odds are in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.

- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

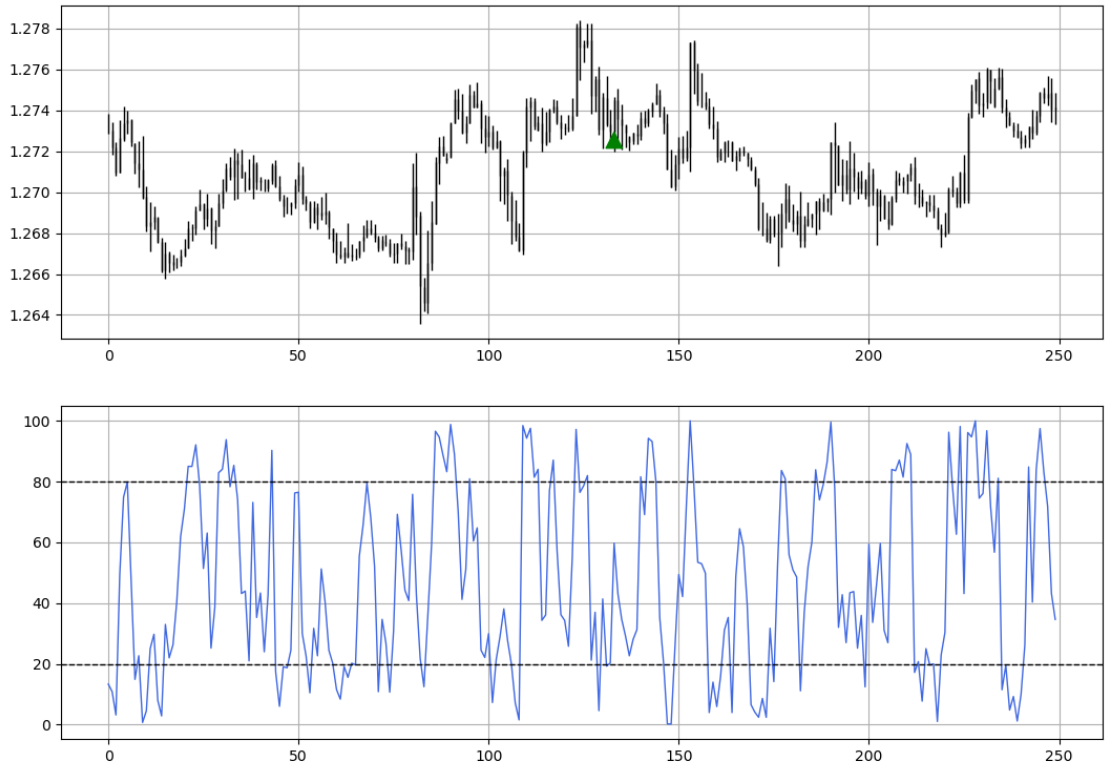


Figure 2-13 USDCAD hourly values with the 5-period stochastic oscillator.

SECTION 2.7 THE PULL-BACK TECHNIQUE

The trading conditions for this technique are as follows:

- **A long signal is generated whenever the stochastic oscillator surpasses the oversold level and pulls-back to it.**
- **A short signal is generated whenever the stochastic oscillator breaks the overbought level and pulls-back to it.**

The conditions state that for a bullish signal, the stochastic oscillator must emerge from its oversold level which is in this case around 30 and then the algorithm must wait for a comeback towards the 30 zone before initiating a buy order. In parallel, for a bearish signal, the stochastic oscillator must dive from its overbought level which is in this case around 70 and then the algorithm must wait for a comeback towards the 70 zone before initiating a sell order.

Figure 2-14 EURUSD hourly values with the 14-period stochastic oscillator.

The signal function for this technique is as follows:

```
def signal(data, stoch_column, tolerance, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        if data[i, stoch_column] > lower_barrier and data[i - 1, stoch_column] < lower_barrier:
            for a in range(i + 1, len(data)):
                if data[a, stoch_column] >= lower_barrier and data[a, stoch_column] <
                  lower_barrier + tolerance and data[a, stoch_column] < data[a - 1,
                  stoch_column]:
                    data[a + 1, buy] = 1
                    break
                elif data[a, stoch_column] > upper_barrier:
                    break
        elif data[i, stoch_column] < upper_barrier and data[i - 1, stoch_column] > upper_barrier:
            for a in range(i + 1, len(data)):
                if data[a, stoch_column] <= upper_barrier and data[a, stoch_column] >
                  upper_barrier - tolerance and data[a, stoch_column] > data[a - 1,
                  stoch_column]:
                    data[a + 1, sell] = -1
                    break
                elif data[a, stoch_column] < lower_barrier:
                    break
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable stoch_column refers to the column of the indicator, the variable tolerance refers to the margin allowed in the pull-back, and the variables buy and sell refer to the columns containing bullish and bearish signals.

As mentioned previously, I recommend adding a neutrality filter. The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

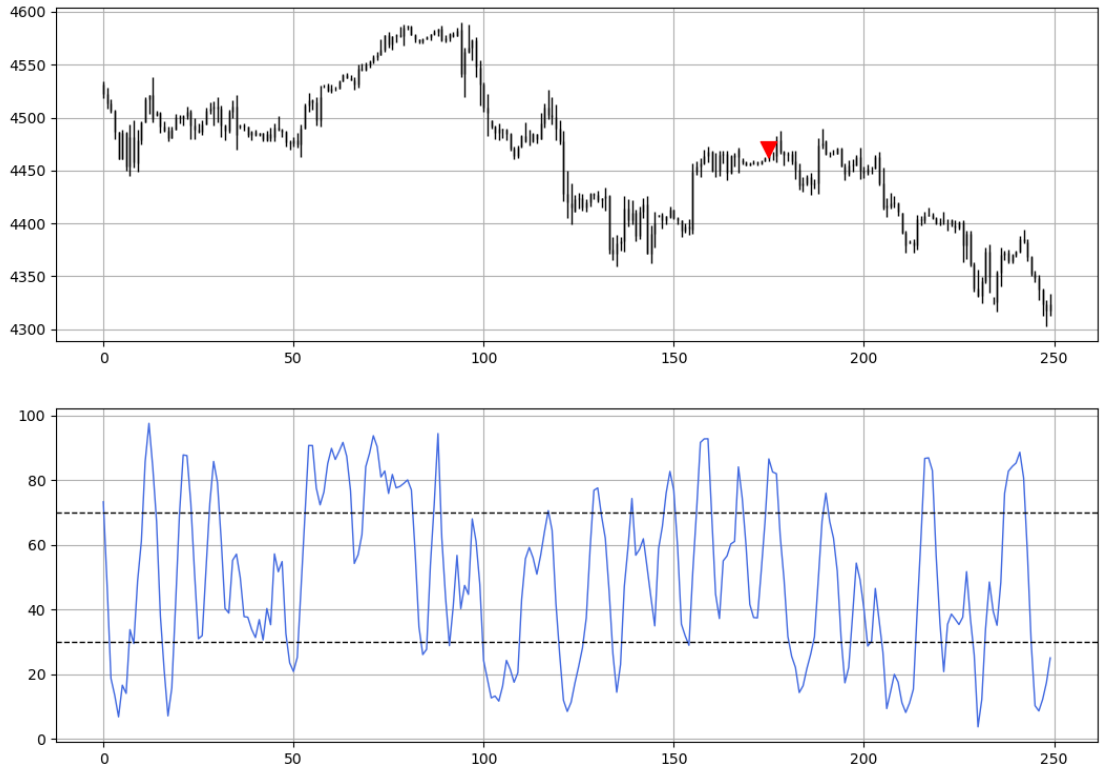


Figure 2-15 S&P500 hourly values with the 14-period stochastic oscillator.

SECTION 2.8 THE EXTREME-TO-EXTREME TECHNIQUE

The conditions for this technique are as follows:

- **A long signal is generated whenever the stochastic oscillator is below the oversold level with the previous value above the overbought level.**
- **A short signal is generated whenever the stochastic oscillator is above the overbought level with the previous value below the oversold level.**

The signal function for this technique is as follows:

```
def signal(data, stoch_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, stoch_column] < lower_barrier and data[i - 1, stoch_column] > upper_barrier:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, stoch_column] > upper_barrier and data[i - 1, stoch_column] < lower_barrier:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable stoch_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

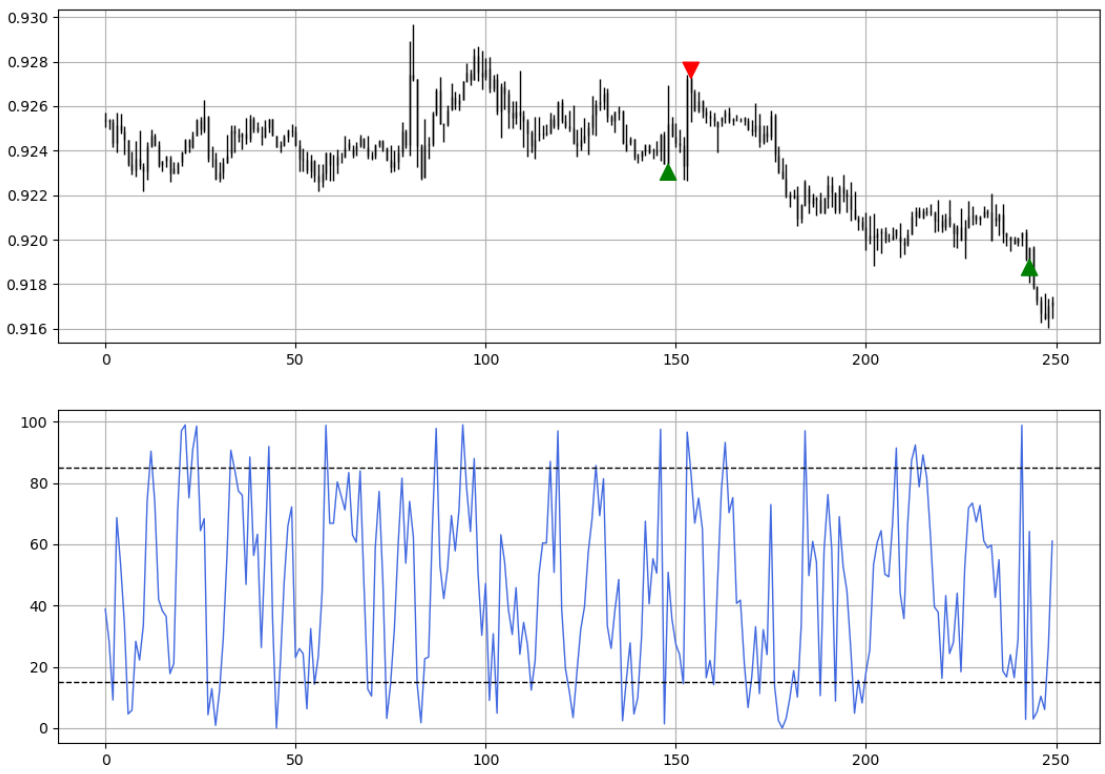


Figure 2-16 USDCHF hourly values with the 2-period stochastic oscillator.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

Figure 2.17 shows the signals generated on S&P500.

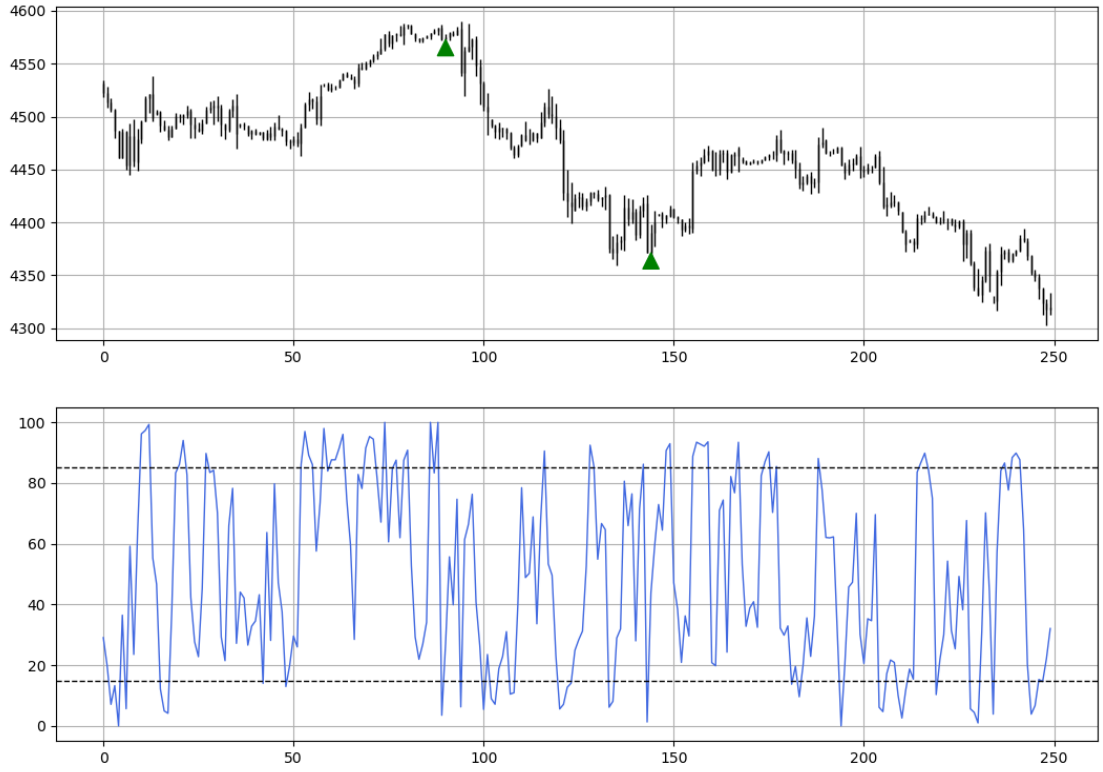


Figure 2-17 S&P500 hourly values with the 5-period stochastic oscillator.

To summarize, the stochastic oscillator is a versatile indicator and traders like it for the following reasons:

- The stochastic oscillator is a bounded indicator which facilitates interpretation.
- The stochastic oscillator is monitored by the majority of the trading world which makes it more impactful.
- There are many techniques that can be used with the stochastic oscillator.
- The stochastic oscillator can be easily combined with other technical indicators to create strategies.

CHAPTER 3 BOLLINGER BANDS

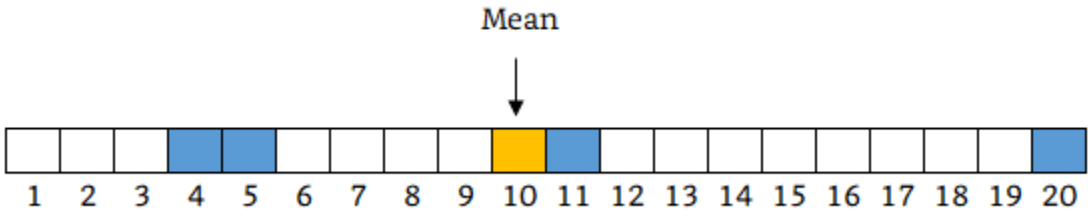
Created by John Bollinger, the bands are a powerful type of analysis that are more statistical in nature than technical. To understand Bollinger bands, I will first start with an introduction to descriptive statistics. Consider the following list {11, 4, 5, 20}. Given the four values, how would you describe the elements? Generally, the best measure that describes the elements in a list is the mean. It is also the best estimate of the next expected value (if you are to add a new element). To calculate the mean of a list, follow the below formula:

$$\bar{x} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right) = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Simply put, the formula states that to find a mean of a list of values, you must sum them and divide the result by their quantity:

$$\bar{x} = \frac{11 + 4 + 5 + 20}{4} = 10$$

The mean of the list is 10. If you plot a horizontal line with the values, you will find that 10 lies approximately on the middle as shown in the next figure:



You must have noticed that I have presented moving averages in the previous chapters. Moving averages are simply a moving mean. The next concept you need to understand is volatility which is a measure of dispersion of values around the mean. In other words, volatility is how much the individual values

fluctuate around the mean value. Generally, volatility is measured using standard deviation and this is what I will do with Bollinger bands. To calculate standard deviation, use this formula:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})}$$

Following the previous formula, to calculate the standard deviation of the list, follow these steps:

- $(4 - 10)^2 = 36$
- $(5 - 10)^2 = 25$
- $(11 - 10)^2 = 1$
- $(20 - 10)^2 = 100$

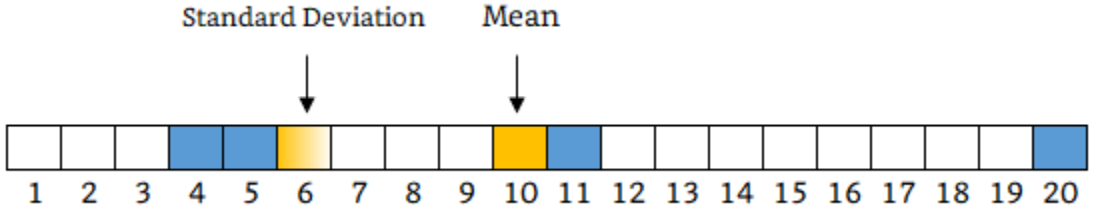
Then, calculate the average of the results found from the previous steps:

$$\bar{x} = \frac{36 + 25 + 1 + 100}{4} = 40.5$$

Finally, take the square root of the mean found in the previous step:

$$\sigma = \sqrt{40.5} = 6.36$$

And voilà, the standard deviation of the list is 6.36 with a mean of 10. The interpretation of this is that on average, the elements of the list are at a distance of 6.36 from the mean.



To code a moving standard deviation (volatility) measure, use the following function:

```
def volatility(data, lookback, close, position):
    data = add_column(data, 1)
    for i in range(len(data)):
        try:
            data[i, position] = (data[i - lookback + 1:i + 1, close].std())
        except IndexError:
            pass
    data = delete_row(data, lookback)
    return data
```

The function defines volatility (standard deviation) where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable close refers to the column of the close price, and the variable position refers to the column where you place the output.

Normal distribution is a probabilistic model which states that 68% of the data lies within 1 standard deviation, 95% of the data lies within 2 standard deviations, and 99.7% of the data lies within 3 standard deviations. This is an assumption even though financial data is generally not considered to be normal but close to normal. This brings us closer to an important point; Bollinger bands are an envelopment technique where the market lies most of the time above what the lower Bollinger band and below what the upper Bollinger band.

How to calculate these bands?

$$Lower\ Bollinger\ Band = Moving\ average - (x.Standard\ deviation)$$

$$Upper\ Bollinger\ Band = Moving\ average + (x.Standard\ deviation)$$

The moving average is generally calculated on the last 20 periods and the x is a variable that is generally 2 which is multiplied by the 20-period standard deviation as well. It is important to calculate the moving average and the standard deviation using the same lookback period.

The next Figure shows the values of USDCAD with 20-period Bollinger bands applied on it.

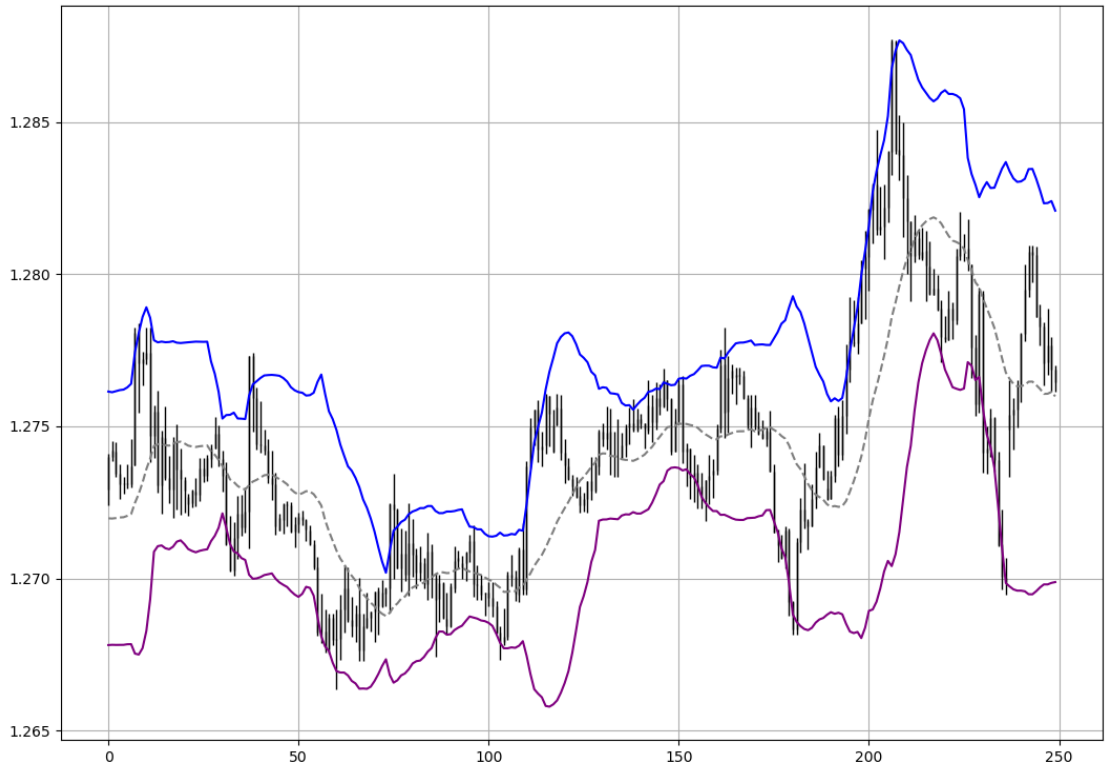


Figure 3-1 USDCAD hourly values with the Bollinger bands.

You can see that most of the time, the price is enveloped within a lower band in purple and an upper band in blue. Naturally, the two act as support and resistance levels respectively. I will later show in the subsequent sections how to generate signals using Bollinger bands. I will also revisit them in the last part of the book that presents more complex strategies.

Generally, Bollinger bands also have the 20-period moving average line appearing in the chart as shown in the dashed grey line in Figure 3-1.

It is clear that Bollinger bands are better suited for a ranging (sideways) market and not a trending market since the change of paradigm makes the market get out of its comfort zone (normality) and therefore, it tends to go above upper bands or below lower bands. The code you can use to calculate Bollinger bands on OHLC data can be as follows:

```
def bollinger_bands(data, lookback, standard_deviation, close, position):
        data = add_column(data, 2)
        data = ma(data, lookback, close, position)
        data = volatility(data, lookback, close, position + 1)
        data[:, position + 2] = data[:, position] + (standard_ deviation * data[:, position + 1])
        data[:, position + 3] = data[:, position] - (standard_ deviation * data[:, position + 1])
        data = delete_row(data, lookback)
        data = delete_column(data, position + 1, 1)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable standard_deviation is the volatility multiplier, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

SECTION 3.1 THE AGGRESSIVE TECHNIQUE

The first technique taught when it comes to the Bollinger bands is to initiate trades the moment the market price reaches the implied extremes which are also known as lower and upper bands. Generally, with 20-period Bollinger bands, you should use 2 standard deviations. The trading conditions for this technique are as follows:

- **A long signal is generated whenever the market price closes below the lower band.**
- **A short signal is generated whenever the market price closes above the upper band.**

The signal function can be coded as follows:

```
def signal(data, close, upper_band, lower_band, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, close] < data[i, lower_band] and data[i - 1, close] > data[i - 1, lower_band]:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, close] > data[i, upper_band] and data[i - 1, close] < data[i - 1, upper_band]:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price , the variable upper_band refers to the column of the upper band, the variable lower_band refers to the column of the lower band, and the variables buy and sell refer to the columns containing bullish and bearish signals.

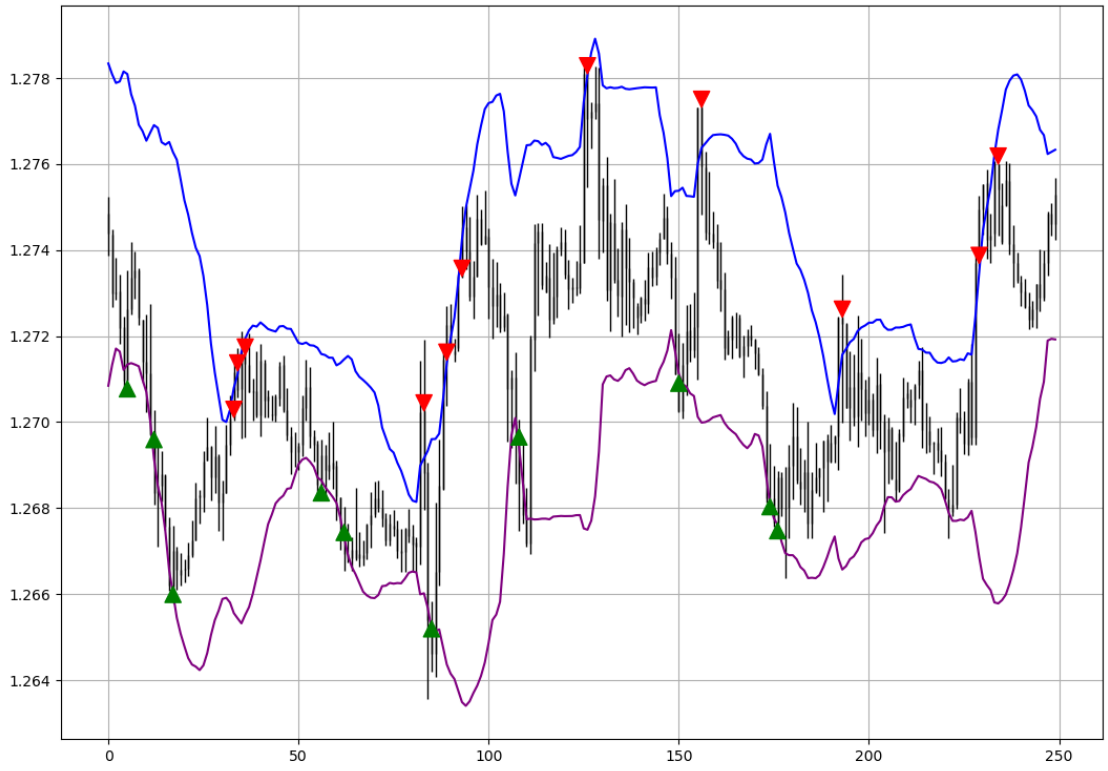


Figure 3-2 USDCAD hourly values with the Bollinger bands.

The advantage of this technique is that it is heavily used by traders as it is supposed to be the default way of using this indicator. It is recommended in a ranging market but highly discouraged in a trending market. The main disadvantage of the technique is that in a trending market, the quality of the signals heavily deteriorates.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.

- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

The next signal chart shows a number of short signals in a bullish market. Notice the low quality of the signals.

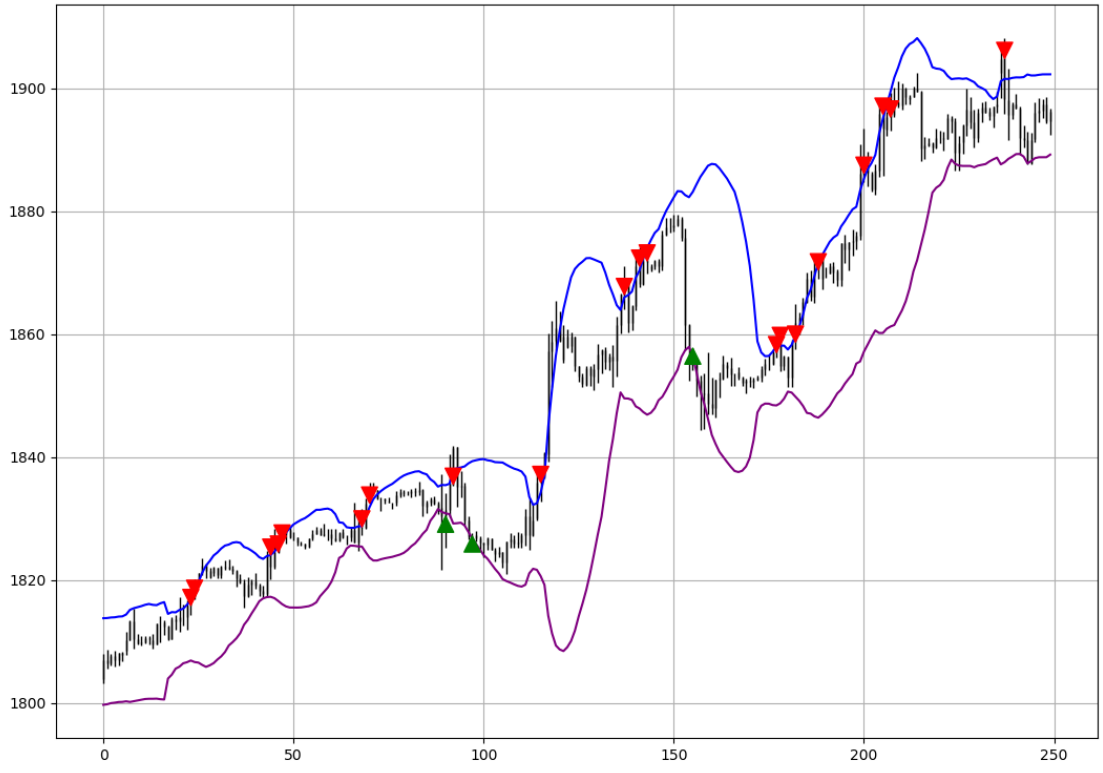


Figure 3-3 Gold hourly values with the Bollinger bands.

SECTION 3.2 THE CONSERVATIVE TECHNIQUE

The second technique resembles the one seen on the RSI and the stochastic oscillator. It deals with the reintegration of a broken band and has the following trading rules:

- **A long signal is generated whenever the market price closes above the lower band after having been below it. It should also be below the middle line.**
- **A short signal is generated whenever the market price below the upper band after having been above it. It should also be above the middle line.**

The signal function can be coded as follows:

```
def signal(data, close, middle_boll, upper_boll, lower_boll, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, close] > data[i, lower_boll] and data[i - 1, close] < data[i - 1,
          lower_boll] and data[i, close] < data[i, middle_boll]:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, close] < data[i, upper_boll] and data[i - 1, close] > data[i –
            1, upper_boll]  and data[i, close] > data[i, middle_boll]:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price , the variable middle_boll refers to the column of the moving average, the variable upper_boll refers to the column of the

upper band, the variable lower_boll refers to the column of the lower band, and the variables buy and sell refer to the columns containing bullish and bearish signals. Take a look at Figure 3-4 to see the signals generated on USDCAD.

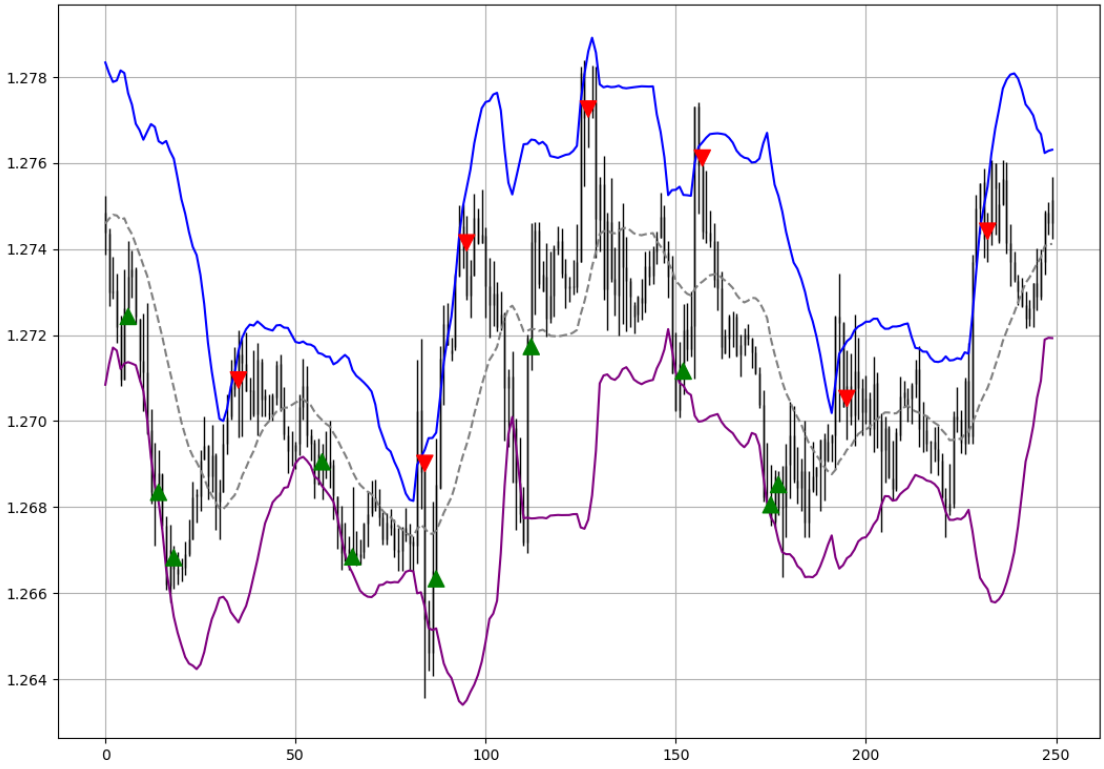


Figure 3-4 USDCAD hourly values with the Bollinger bands.

The advantage of this technique is that it awaits reintegration which is a form of confirmation that the market wants to return to normality. The main disadvantage of the technique is that in a trending market, the quality of the signals heavily deteriorates.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are against you if you use buy and sell signals in a trending market with no trend filter.
- The odds are harshly against you if you use buy signals in a bearish market and sell signals in a bullish market.

The next signal chart shows a number of signals on USDCHF. Notice that I have also kept the moving average line in this section and that is to use it as a filter. When the market closes above the lower band, there is a risk that this close is also near the upper band or even higher. In this case, the algorithm will not be able to differentiate a good signal from a bad one.

This is why I have added a moving average filter which states that in a bullish signal, the close price must be above the lower band but also below the moving average line. In parallel, a bearish signal is only validated with a close price below the upper band but above the moving average line.

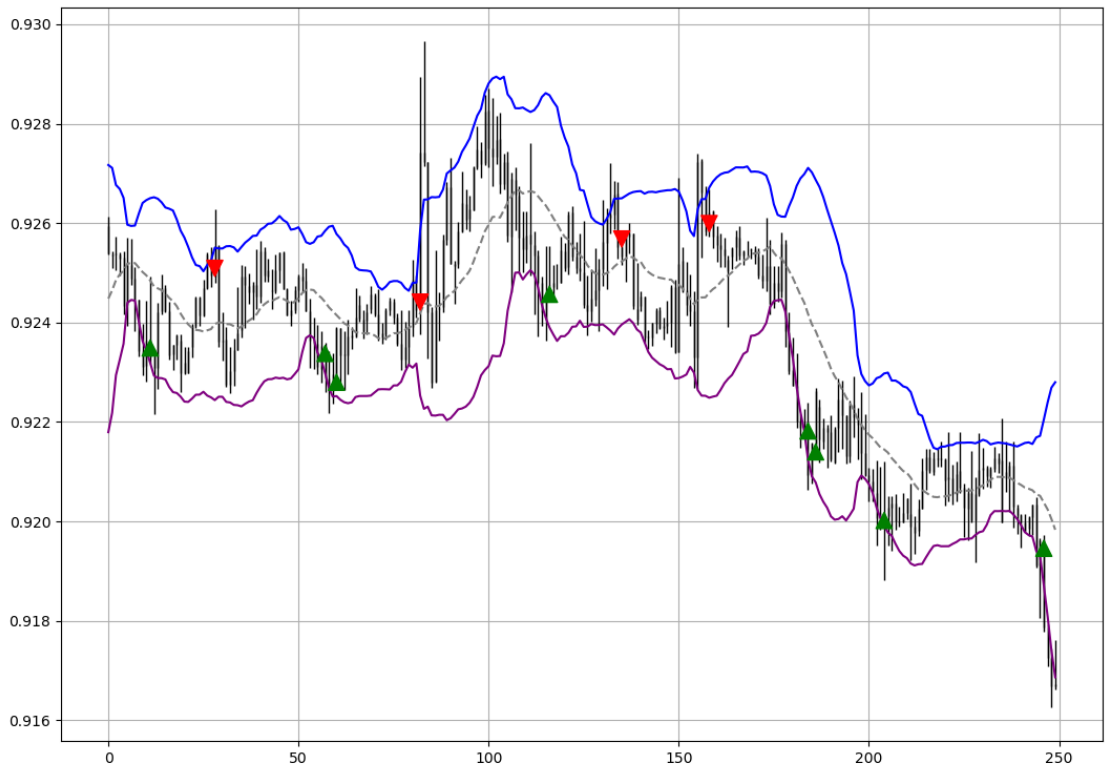


Figure 3-5 USDCHF hourly values with the Bollinger bands.

Generally, the reintegration technique can be considered the equivalent of the conservative technique seen on the RSI and the stochastic oscillator. By awaiting the confirmation from the market, you may be able to stay away from unnecessary risks.

SECTION 3.3 THE PERCENTAGE BANDS TECHNIQUE

The percentage bands indicator %B is a normalization technique used to create an oscillator that shows the market's position relative to the Bollinger bands. It is mostly useful in detecting divergences. To calculate %B, use the following formula:

$$\%B_i = \frac{Close_i - Lower\ band_i}{Upper\ band_i - Lower\ band_i}$$

This should remind you of the stochastic oscillator's calculation method. It is important to understand the functioning of this indicator. The rules of thumb to remember are the following:

- A value above 1 means that the market is above its upper Bollinger band.
- A value at 1 means that the market is exactly at its upper Bollinger band.
- A value at -1 means that the market is exactly at its lower Bollinger band.
- A value below -1 means that the market is below its lower Bollinger band.

You should also remember what a divergence is and therefore, you will be able to understand the trading conditions for the %B technique:

- **A long signal is generated whenever a bullish divergence is spotted. This occurs when noticing lower market prices with higher indicator readings.**
- **A short signal is generated whenever a bearish divergence is spotted. This occurs when noticing higher market prices with lower indicator readings.**

As you know, the signal function of the divergence technique is lengthy which is why I have decided it to only put it in the GitHub repository. You can access it through the link[30] or through the below QR code.

However, let's check out the code that allows you to calculate the %B from the Bollinger bands.

```
def bollinger_percentage_bands(data, close, upper_boll, lower_boll, position):
    data = add_column(data, 1)
    # Calculating the Bollinger percentage bands
    for i in range(len(data)):
        data[i, position] = (data[i, close] - data[i, lower_boll]) / (data[i,
                            upper_boll] –data[i, lower_boll])
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price, the variable upper_boll refers to the column of the upper band, the variable lower_boll refers to the column of the lower band and the variable position refers to the column where you place the output.

[30] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

The next Figure shows the signals generated on USDCAD's hourly values from the %B.

Figure 3-6 USDCAD hourly values with the percent Bollinger indicator.

The advantage of this technique is that it uses the well-known divergence technique on a well-known indicator called Bollinger bands. It is recommended in both ranging and trending markets but more confirmation from other techniques remains necessary. The main disadvantage of the technique is that it is very subjective as the interpretation of a divergence differs from one trader to another.

The technique has the following biases that impact its predictability:

- The odds are greatly in your favor if you use buy signals in a bullish market and sell signals in a bearish market.
- The odds are in your favor if you use buy and sell signals in a ranging market.
- The odds are in your favor if you use buy and sell signals in a trending market with no trend filter.
- The odds are against you if you use buy signals in a bearish market and sell signals in a bullish market.

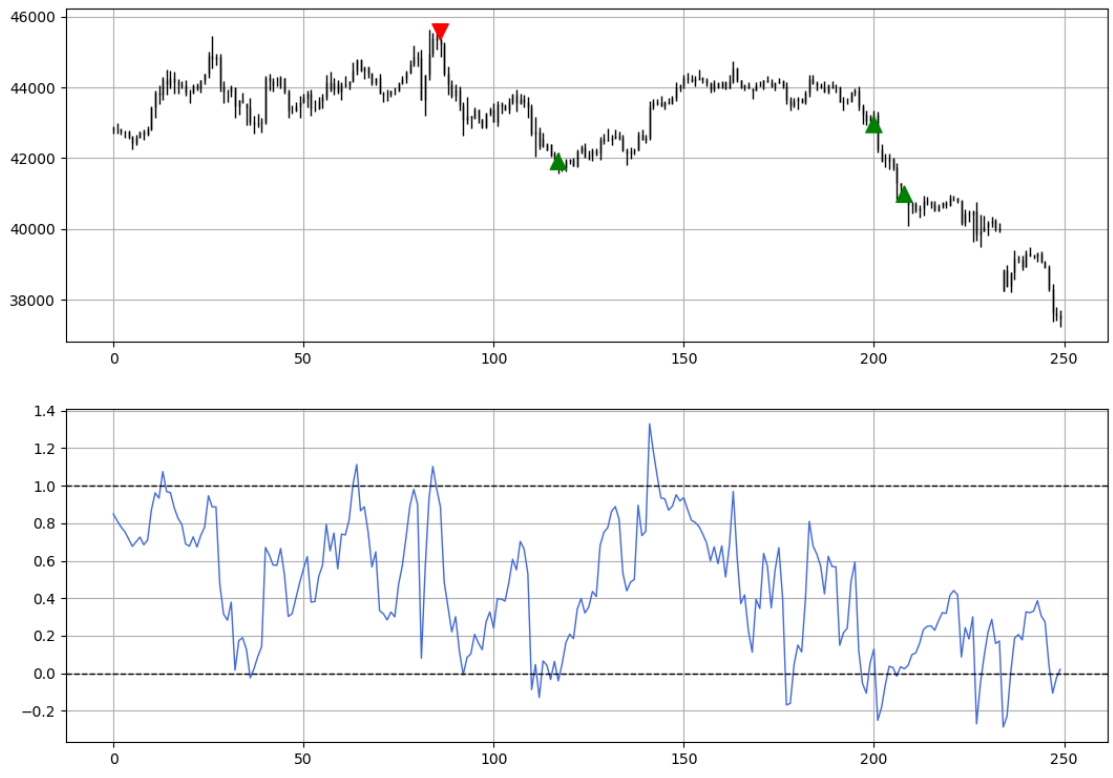


Figure 3-7 BTCUSD hourly values with the percent Bollinger indicator.

To summarize, Bollinger bands are extremely popular for the following reasons:

- They have clear and interpretable trading rules.
- They are monitored by the majority of the trading world which makes them more impactful.
- Bollinger bands can be easily combined with other technical indicators to create strategies.

You are now ready to start learning about secondary indicators. Make sure to master the concepts from this part as primary indicators are paramount to understanding the pillars of technical analysis. They are also key ingredients in many successful trading strategies.

PART 2 SECONDARY CONTRARIAN INDICATORS

Secondary indicators are less notorious indicators that aim for the same thing as primary contrarian indicators. This part will present a number of the mentioned indicators. The main idea of this part is not to generate signals but to understand how indicators are created. I have chosen the indicators based on their differences so that you get more exposure to different ways of calculating indicators. The key takeaway of the part is to create your own indicator based on your own idea.

Generally, secondary indicators are unlikely to provide significant predictive signals in their default forms and this is normal because even primary indicators rarely add value when used in their default forms. This is why you should focus more on the how-to-create than the how-to-use. the secondary contrarian indicators I will present are the following:

- The momentum indicator.
- The detrended price oscillator.
- The modified Fisher transform.
- The relative vigor index.
- The Demarker.
- The stochastic-RSI indicator.
- The RSI-ATR indicator.
- The directional probability index.
- The parabolic relative strength.
- The Chande momentum oscillator.

CHAPTER 4 THE MOMENTUM INDICATOR

The momentum indicator is one of the simplest bounded indicators that calculates the relative strength of the current close price to the close price from a pre-specified number of periods ago, generally 14. The formula is as follows:

$$Momentum\ indicator_i = \frac{Close\ price_i}{Close\ price_{i-n}}\ x\ 100$$

Understanding the notations in mathematical formulas is key to mastering the basics of how indicators are created. The variable i refers to the current index[31] and the variable n refers to the lookback period. Therefore, by default i – n is the index of the close 14 periods ago. The function that you can use to code the momentum indicator is as follows:

```
def momentum_indicator(data, lookback, close, position):
    data = add_column(data, 1)
    for i in range(len(data)):
        data[i, position] = data[i, close] / data[i - lookback, close] * 100
    data = delete_row(data, lookback)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

[31] The current time step which is the current close price. This is also the row in the case of arrays.

The next Figure shows the hourly EURUSD values versus the 14-period momentum indicator.

Figure 4-1 EURUSD hourly values with the 14-period momentum indicator.

Since the indicator is unbounded, classical techniques such as the aggressive and conservative techniques cannot be used. However, other techniques such as the cross technique can be valid.

The main advantage of the indicator is its simplicity, and the main disadvantage is its unboundedness and lack of notoriety among traders.

To summarize, the momentum indicator has no oversold nor overbought levels and the alternative solutions are the cross technique, the divergence technique, and tracing manual support and resistance lines.

CHAPTER 5 THE DETRENDED PRICE OSCILLATOR

The detrended price oscillator (DPO) is a contrarian indicator calculated using moving averages. The DPO is a price transformation that uses the price from a certain period ago in relation to the current moving average. It is a contrarian indicator that aims to remove the trend in order to have a better vision on troughs and peaks. It is calculated the following way:

- Select a lookback period on which the calculations will be based. For example, if you select 14, then the moving average will be calculated on the last 14 observations counting the current one. Let's start with the standard 14 periods.
- Calculate the variable k which will be based on the lookback you choose. The formula used to calculate the variable is as follows:

$$k = \frac{Lookback}{2} + 1$$

- Assuming that the lookback period is n, use the following formula to calculate the DPO:

$$DPO_i = SMA_{i-n:i} - Close\ price_k$$

The issue with the DPO is that there is no oversold nor overbought levels and the solutions are limited such as the cross technique, the divergence technique, and tracing manual support and resistance lines.

The function that you can use to code the DPO is as follows:

```
def detrended_price_oscillator(data, lookback, close, position):
    # Calculating the moving average
    data = ma(data, lookback, close, position)
    data = add_column(data, 1)
    # Calculating the detrended price oscillator
    for i in range(len(data)):
        x = int((lookback / 2) + 1)
        data[i, position + 1] = (data[i - x, close] - data[i, position]) * 100
    data = delete_column(data, position, 1)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator. Figure 5-1 shows the hourly GBPUSD values versus the 14-period DPO.

You can notice that I have traced two dashed black lines around 0.25 and -0.25. This is to show that one way of using this indicator is by tracing implied support and resistance horizontal lines. It is a form of forcing oversold and overbought levels by looking at the historical reactions of the DPO. This technique is risky because the volatility and the dynamic of the DPO is bound to change from time to time thus rendering the levels unjustified.

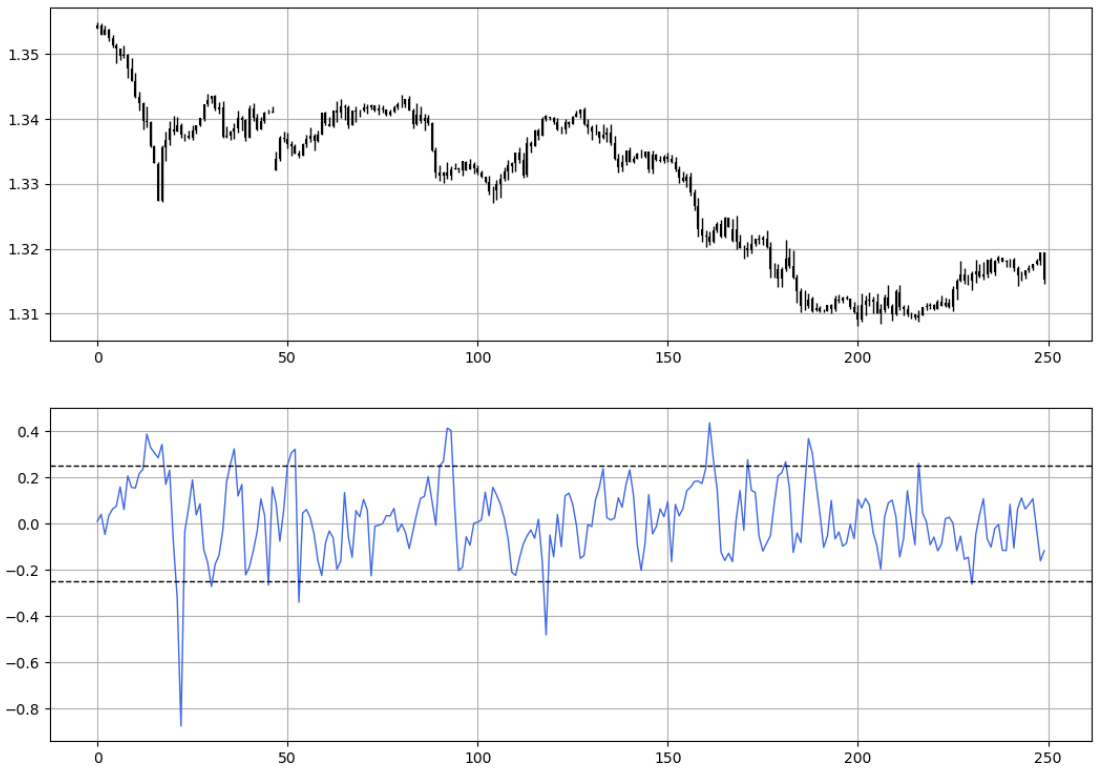


Figure 5-1 GBPUSD hourly values with the 14-period DPO.

To summarize, the DPO is a contrarian indicator that uses a simple formula and relies on a moving average. I recommend you use it with other more known indicators like the RSI. You can apply the divergence technique using this indicator, but you have to continuously update the oversold and overbought levels from where you expect the divergence to occur. Otherwise, you can apply the special divergence strategy which I will present in the last part of the book.

CHAPTER 6 THE MODIFIED FISHER TRANSFORM

Created by John F. Ehlers, the Fisher transform seeks to transform the price into a normal Gaussian (normal) distribution. In this chapter, I will present a modified version of the original Fisher transform that I have found to perform slightly better[32].

The steps to create the modified Fisher transformation (MFT) are somewhat similar to the original Fisher Transformation. Here is how to do it:

- Select a lookback period on which the calculations will be based on.
- Calculate the stochastic oscillator using the formula I have previously presented. This stochastic should not have any slowing nor smoothing.
- Divide the results from the last step by 100 and then apply the following formula to trap the values between -1 and 1:

$$Value_i = (2\ x\ Value_i) - 1$$

- Impose a condition to eliminate the extremes (-1's and 1's) and transform them into -0.999's and 0.999's. This is done so that you get no infinite values which makes the whole calculations invalid. Also, by doing this, you will have a bounded indicator.
- Finally, apply the following formula to get the MFT:

$$MFT_i = \frac{1}{2}\ x \ln\left(\frac{1 + Value_i}{1 - Value_i}\right)$$

[32] This is based on my experience and in no way states that the modification is better than Ehlers's original indicator.

To code the MFT, you can follow this syntax:

```
def fisher_transform(data, lookback, high, low, close, position):
        data = add_column(data, 1)
        data = stochastic_oscillator(data, lookback, high, low, close, position)
        data[:, position] = data[:, position] / 100
        data[:, position] = (2 * data[:, position]) - 1
        for i in range(len(data)):
                if data[i, position] == 1:
                        data[i, position] = 0.999
                if data[i, position] == -1:
                        data[i, position] = -0.999
        for i in range(len(data)):
                data[i, position + 1] = 0.5 * (np.log((1 + data[i, position]) / (1 - data[i,
                                        position])))
        data = delete_column(data, position, 1)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

Naturally, with the small modification that I have added over the original formula, the maximum values should be 3.80 and the minimum values should be -3.80.

The next Figure shows the hourly ETHUSD values versus the 10-period MFT.

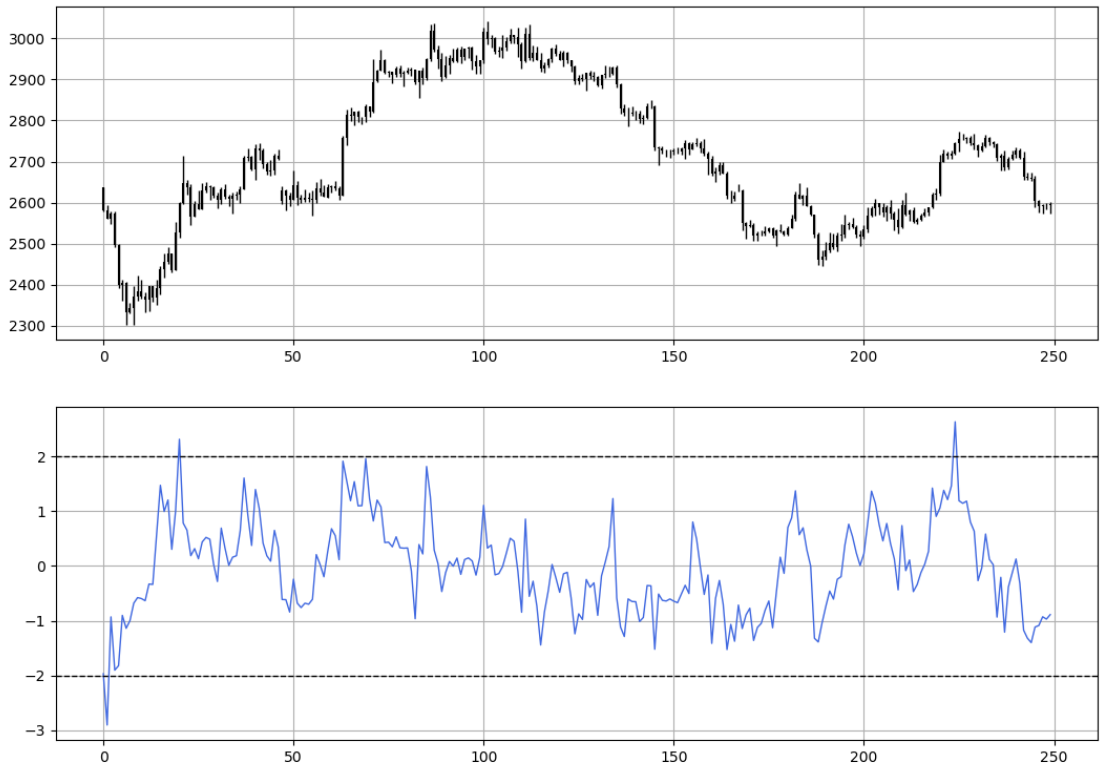


Figure 6-1 ETHUSD hourly values with the 14-period MFT.

There are many ways to use the MFT. The following are the conditions of the simplest technique:

- **A long signal is generated whenever the MFT reaches the oversold level which is generally set at -2.**
- **A short signal is generated whenever the MFT reaches the overbought level which is generally set at 2.**

A more conservative version is to use -3.80 and 3.80 as oversold and overbought levels respectively, but the signals tend to be rare.

CHAPTER 7 THE RELATIVE VIGOR INDEX

Also created by John F. Ehlers, the relative vigor index (RVI) is a contrarian indicator calculated through a number of interesting formulas. The RVI is composed of many building blocks, namely the totality of OHLC data and moving averages. It seeks to measure the strength of the trend by comparing the ranges between the open/close and high/low values.

The RVI is unbounded, and this is a weakness that decreases the number of techniques that can be applied. The numerator line of the RVI can be calculated this way:

- Calculate the difference between the close and open price. Let's call this variable A.
- Calculate the difference between the close price and the open price from one period ago. Let's call this variable B. Multiply the result by 2.
- Calculate the difference between the close price and the open price two periods ago. Let's call this variable C. Multiply the result by 2.
- Calculate the difference between the close price and the open price three periods ago. Let's call this variable D.

The denominator line of the RVI can be calculated this way:

- Calculate the difference between the high and low price. Let's call this variable E.
- Calculate the difference between the high price and the low price one period ago. Let's call this variable F. Multiply the result by 2.
- Calculate the difference between the high price and the low price two periods ago. Let's call this variable G. Multiply the result by 2.
- Calculate the difference between the high price and the low price three periods ago. Let's call this variable H.

Apply the formulas below:

$$Numerator = \frac{A + B + C + D}{6}$$

$$Denominator = \frac{E + F + G + H}{6}$$

And finally, the RVI is calculated using this formula below:

$$RVI_i = \frac{SMA_{i-n:i}(Numerator)}{SMA_{i-n:i}(Denominator)}$$

The RVI is mostly used with the cross technique which means that you need a signal line[33]. Remember that the cross technique is composed of the indicator and a signal line that is generally a moving average of the indicator. With the RVI, the signal line is calculated using the below formula:

$$Signal\ line_i = \frac{RVI_i + (2\ x\ RVI_{i-1}) + (2\ x\ RVI_{i-2}) + RVI_{i-3}}{6}$$

Obviously, the RVI is unusual and highly prone to overfitting bias due to the specific weighting it has.

However, this should not stop you from back-testing this indicator. The next Figure illustrates a 14-period RVI with the hourly values of EURUSD.

[33] The signal line is another name for the moving average that is applied on an indicator. A third name that can also be used is the smoothing line like I have shown you with the stochastic oscillator.

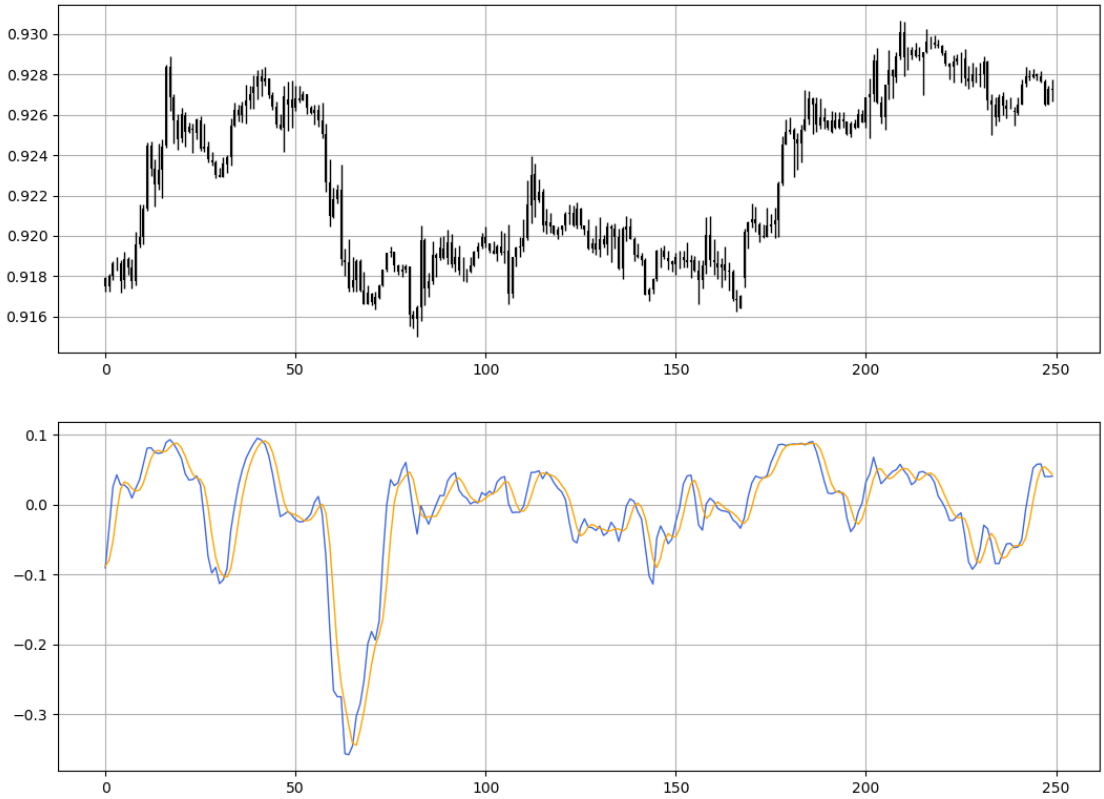


Figure 7-1 USDCHF hourly values with the 14-period RVI.

The main issue is that crosses happen all the time and therefore, I recommend you use the neutrality filter which makes the trading conditions as follows:

- **A long signal is generated whenever the RVI surpasses its moving average while below 0.**
- **A short signal is generated whenever the RVI breaks its moving average while above 0.**

The function that you can use to code the RVI is as follows:

```
def rvi(data, lookback, open_column, high, low, close, position):
        data = add_column(data, 4)
        # Numerator
        for i in range(len(data)):
                data[i, position] = (data[i, close] - data[i, open_column]) +
                                    (2 * (data[i, close] - data[i - 1, open_column])) +
                                    (2 * (data[i, close] - data[i - 2, open_column])) +
                                    (data[i, close] – data[i - 3, open_column])
        data[:, position] = data[:, position] / 6
        data = ma(data, lookback, position, position + 1)
        # Denominator
        for i in range(len(data)):
                data[i, position + 2] = (data[i, high] - data[i, low]) +
                                        (2 * (data[i, high] - data[i - 1, low])) +
                                        (2 * (data[i, high] - data[i - 2, low])) +
                                        (data[i, high] - data[i - 3, low])
        data[:, position] = data[:, position] / 6
        data = ma(data, lookback, position + 2, position + 3)
        # RVI
        data[:, position + 4] = data[:, position + 1] / data[:, position + 3]
        # Signal
        for i in range(len(data)):
                data[i, position + 5] = ((data[i, position + 4]) + (2 * (data[i - 1, position + 4]))
                                        + (2 * (data[i –2, position + 4])) + (data[i - 3, position + 4])) / 6
        data = delete_column(data, position, 4)
        data = delete_row(data, lookback + 10)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable open_price refers to the column of the open price[34], the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

To summarize, the RVI is an indicator better suited with the cross technique since it has no boundaries. The calculation method makes it less flexible as specific weightings are unlikely to work in every market.

[34] I have added the word column to the variable open because it is already a built-in statement in Python and thus the need to specify it so that no error occurs.

CHAPTER 8 THE DEMARKER

Tom Demark, one of the greatest technical analysts has created many indicators and reshaped the world of technical analysis. Among the indicators he created is the Demarker which is obviously named after him. The Demarker resembles the RSI in some sort which is why I recommend using them both to confirm each other. To calculate the Demarker, follow these steps:

- Select a lookback period on which the calculations will be based on. For example, if you select 14, then the calculations will be done on the last 14 observations counting the current one. Let's start with the standard 14 periods.
- Calculate the two main variables in the indicator which are referred to as the DeMAX and the DeMIN. Their formulas are as follows:

$$DeMAX = \begin{cases} High - Previous\ high\ if > 0 \\ 0\ if\ High - Previous\ high < 0 \end{cases}$$

$$DeMIN = \begin{cases} Previous\ low - Low\ if > 0 \\ 0\ if\ Previous\ low - Low < 0 \end{cases}$$

The first formula states that the DeMAX value equals the current high minus the previous high if it is greater than zero. Otherwise, it equals zero. Similarly, the DeMIN value equals the previous low minus the current low if it is greater than zero.

- Finally, to calculate the Demarker, divide the 14-period moving average of the DeMAX by the summation of the DeMAX and DeMIN averaged values. This is illustrated by the formula below:

$$Demarker = \frac{MA_i(DeMAX)}{MA_i(DeMAX) + MA_i(DeMIN)}$$

The next Figure illustrates a 14-period Demarker with the hourly values of EURUSD.

Figure 8-1 EURUSD hourly values with the 14-period Demarker.

Unlike the RSI, it is better to set the default oversold and overbought levels to 0.20 and 0.80 on the Demarker as opposed to 30 and 70 (which can be the parallels of 0.30 and 0.70). The trading conditions by default are:

- **A long signal is generated whenever the Demarker reaches the oversold level generally set at 0.20.**
- **A short signal is generated whenever the Demarker reaches the overbought level generally set at 0.80.**

The function that you can use to code the Demarker is as follows:

```
def demarker(data, lookback, high, low, position):
    # Adding columns
    data = add_column(data, 3)
    # Calculating DeMAX
    for i in range(len(data)):
        if data[i, high] > data[i - 1, high]:
            data[i, position] = data[i, high] - data[i - 1, high]
        else:
            data[i, position] = 0
    # Calculating the moving average on DeMAX
    data = ma(data, lookback, position, position + 1)
    # Calculating DeMIN
    for i in range(len(data)):
        if data[i - 1, low] > data[i, low]:
            data[i, position + 2] = data[i - 1, low] - data[i, low]
        else:
            data[i, position + 2] = 0
    # Calculating the moving average on DeMIN
    data = ma(data, lookback, position + 2, position + 3)
    # Calculating Demarker
    for i in range(len(data)):
        data[i, position + 4] = data[i, position + 1] / (data[i, position + 1] +
                                data[i, position + 3])
    # Cleaning
    data = delete_column(data, position, 4)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable high refers to the column of the high price, the variable low refers to the column of the low price, and the variable position refers to the column where you place the indicator.

Figure 8-2 S&P500 hourly values with the 14-period Demarker.

To summarize, the Demarker is a contrarian indicator that resembles the RSI. I recommend you use both indicators together to confirm each other. Also, techniques that can be applied on the RSI are generally applicable on the Demarker as well.

CHAPTER 9 THE STOCHASTIC-RSI INDICATOR

A fusion of the two most known and most used technical indicators? Yes, please. The stochastic-RSI incorporates both indicators to create a new one that is used exactly the same way as the stochastic oscillator[35]. This implies that the techniques seen in chapter 2 can be applied on the stochastic-RSI.

Basically, the indicator is simply a stochastic oscillator applied on the values of the RSI. To calculate the stochastic-RSI, follow these steps:

- Select a lookback period on which the calculations will be based on. With this indicator, you need to select more than one lookback period as there is one for the stochastic oscillator and one for the RSI. Also, keep in mind that you will use the slowing and smoothing techniques of the stochastic. By default, you should use 14 periods for both the RSI and the stochastic oscillator. The slowing and smoothing periods are by default 3 as I have previously shown in chapter 2.
- Apply the below formula to find the stochastic-RSI:

$$StochasticRSI_i = \frac{RSI_i - Lowest\ RSI_{i-n:i}}{Highest\ RSI_{i-n:i} - Lowest\ RSI_{i-n:i}}$$

The next Figure illustrates a 14-period stochastic-RSI with the hourly values of EURUSD.

[35] Remember, there is some differences in the way you use the RSI and the stochastic oscillator due to their volatility and interpretation.

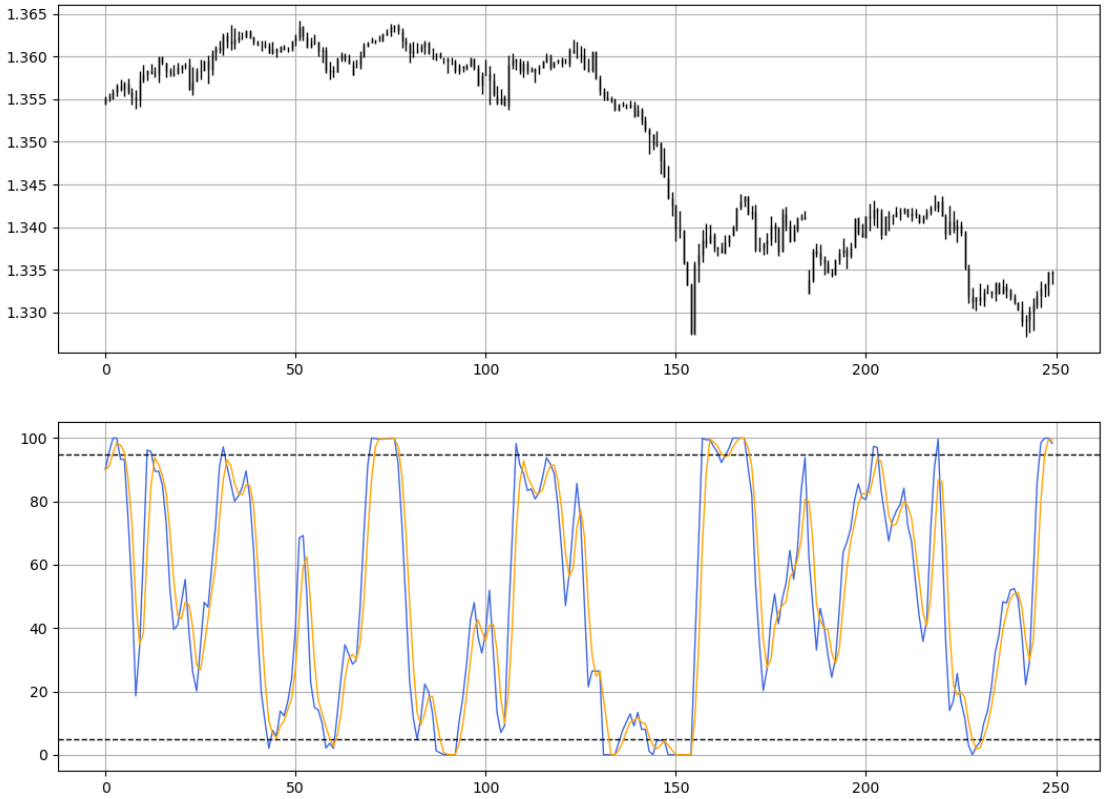


Figure 9-1 GBPUSD hourly values with the 14-period stochastic-RSI.

The indicator may be more extreme in its fluctuations and therefore, it is better to have a wide normality interval. In this example, the oversold level is at 5 and the overbought level is at 95. The trading conditions by default are:

- **A long signal is generated whenever the stochastic-RSI reaches the oversold level generally set at 5.**
- **A short signal is generated whenever the stochastic-RSI reaches the overbought level generally set at 95.**

The function that you can use to code the stochastic-RSI is as follows:

```
def stochastic_rsi(data, lookback_rsi, lookback_stoch, close, position,
                   slowing_period = 1, smoothing_period = 1):
    data = rsi(data, lookback_rsi, close, position)
    data = add_column(data, 1)
    for i in range(len(data)):
        try:
            data[i, position + 1] = (data[i, position] - min(data[i –
                                     lookback_stoch + 1:i + 1, position])) /
                                    (max(data[i - lookback_stoch + 1:i + 1,
                                    position]) - min(data[i - lookback_stoch +
                                    1:i + 1, position]))
        except ValueError:
            pass
    data = delete_column(data, position, 1)
    data[:, position] = data[:, position] * 100
    data = ma(data, slowing_period, position, position + 1)
    data = ma(data, smoothing_period, position + 1, position + 2)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback_rsi refers to the number of past time periods to consider in the calculation of the RSI including the current one, the variable lookback_stoch refers to the number of past time periods to consider in the calculation of the stochastic oscillator including the current one, the variable close refers to the column of the close price, the variable position refers to the column where you place the indicator, the variable slowing_period refers to the number of time periods to consider in the calculation of the first moving average, and the

variable smoothing_period refers to the number of time periods to consider in the calculation of the second moving average.

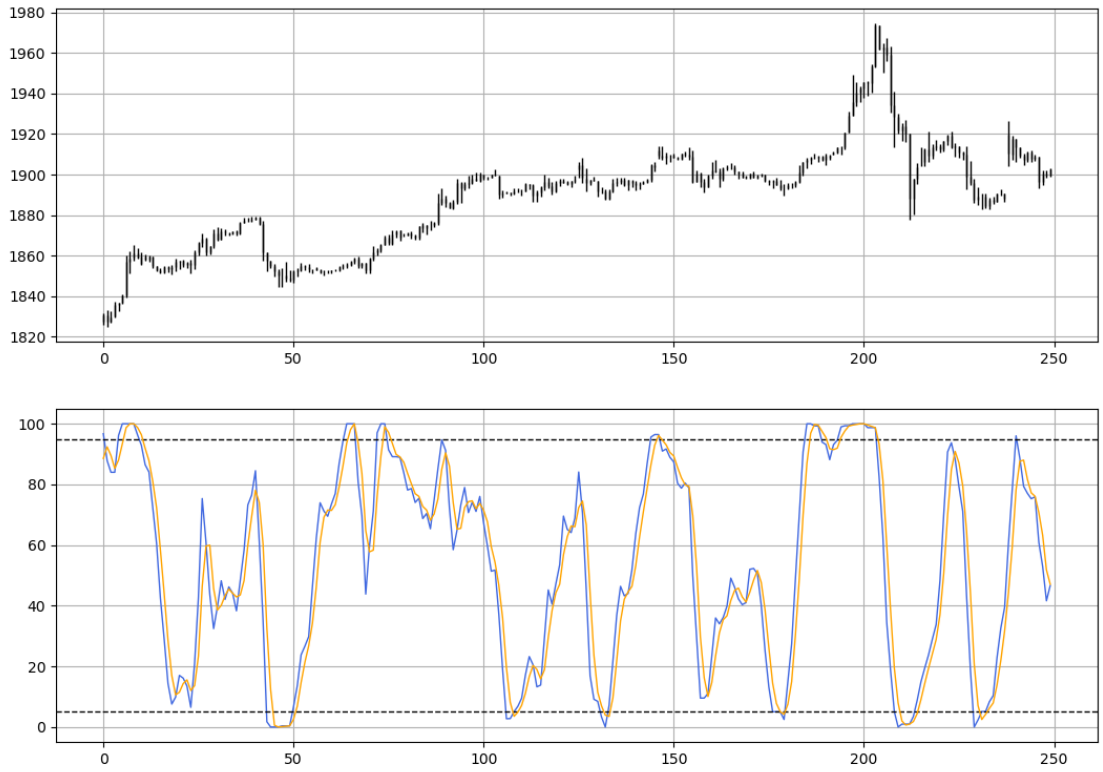


Figure 9-2 Gold hourly values with the 14-period stochastic-RSI.

To summarize, the stochastic-RSI is a contrarian indicator that combines the RSI and the stochastic oscillator. I recommend you use it alongside the two components independently.

CHAPTER 10 THE RSI-ATR INDICATOR

What if you take volatility into account when you calculate the RSI? Generally, including endogenous and exogenous variables within the trading framework improves the picture marginally.

There is no real proof that the volatility adjustment performs better than the default RSI but, in my experience, it delivers less false signals and adds some interesting hidden signals that were not visible when using only the regular RSI. Before you define this new volatility-weighted RSI, you need to define the average true range (ATR). The ATR can be used as a gauge for historical volatility.

First, you should understand how the true range is calculated (the ATR is just the smoothed average of that calculation). Consider an OHLC data. For each time period (bar), the true range is simply the greatest of the three absolute price differences:

$$True\ range\ A_i = High_i - Low_i$$

$$True\ range\ B_i = |High_i - Close\ price_{i-1}|$$

$$True\ range\ C_i = |Close\ price_{i-1} - Low_i|$$

$$True\ range_i = max\ (True\ range\ A_i,\ True\ range\ B_i,\ True\ range\ C_i)$$

Once you get the maximum out of the above three, you simply take a smoothed average[36] of n periods of the true ranges to get the ATR.

[36] Make sure to understand that this is the smoothed moving average and not the simple moving average.

To code the ATR, use the below syntax.

```
def atr(data, lookback, high, low, close, position):
        data = add_column(data, 1)
        for i in range(len(data)):
                try:
                        data[i, position] = max(data[i, high] - data[i, low], abs(data[i, high] –
                                                data[i - 1, close]), abs(data[i, low]  - data[i - 1,
                                                close]))
                except ValueError:
                        pass
        data[0, position] = 0
        data = smoothed_ma(data, 2, lookback, position, position + 1)
        data = delete_column(data, position, 1)
        data = delete_row(data, lookback)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

Take a look at the next Figure which illustrates the GBPUSD versus the 14-period ATR.

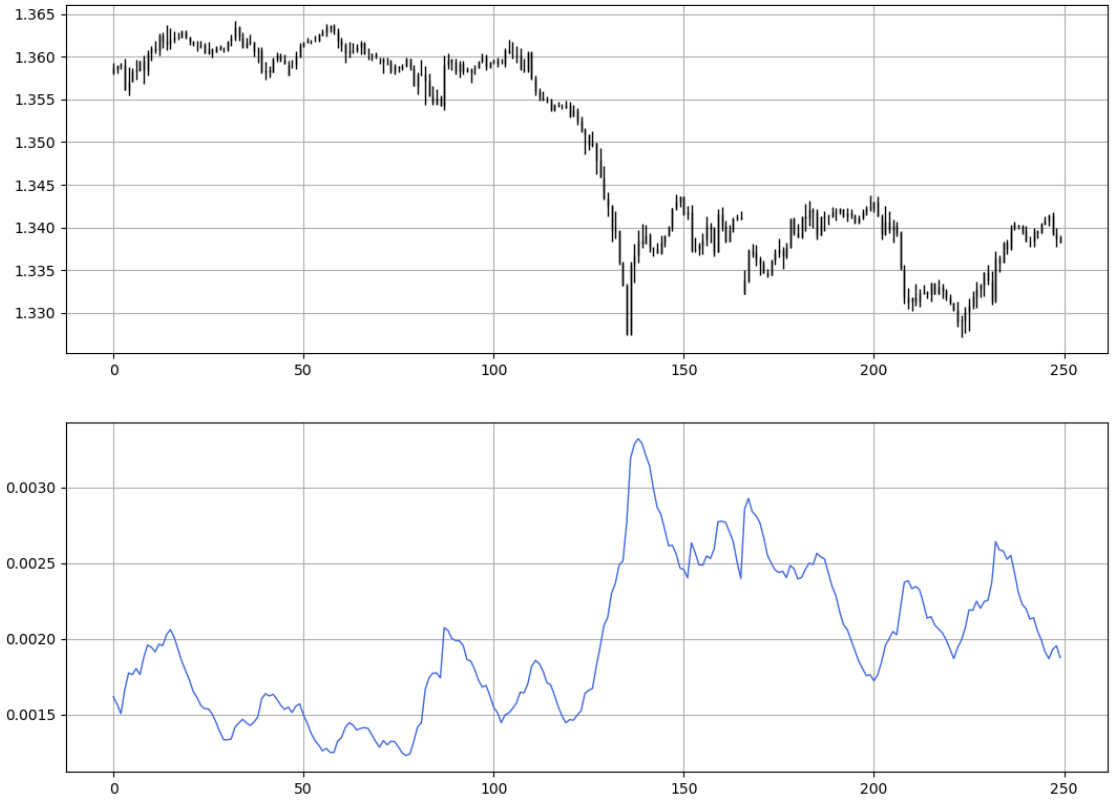


Figure 10-1 GBPUSD hourly values with the 14-period RSI-ATR.

To calculate the RSI-ATR, follow these steps:

- Calculate a 14-period RSI on the close price.
- Calculate a 14-period ATR as shown previously.
- Divide the RSI from the first step by the ATR in the second step.
- Calculate a 14-period RSI on the result from the third step.

Be careful as the last step takes the result from the third step and normalizes it into an RSI calculation. You generally calculate the RSI on the close price but

this time you will calculate it on the ratio between the RSI (based on close prices) and the ATR. The next Figure shows the EURUSD with the 14-period RSI-ATR with signals generated using the usual aggressive technique.

Figure 10-2 EURUSD hourly values with the 14-period RSI-ATR.

The indicator may be more extreme in its fluctuations due to its dependence on volatility. The trading conditions by default are:

- **A long signal is generated whenever the RSI-ATR reaches the oversold level generally set at 30.**
- **A short signal is generated whenever the RSI-ATR reaches the overbought level generally set at 70.**

The function that you can use to code the RSI-ATR is as follows:

```
def rsi_atr(data, lookback_rsi, lookback_atr, lookback_rsi_atr, high, low, close, position):
        data = rsi(data, lookback_rsi, close, position)
        data = atr(data, lookback_atr, high, low, close, position + 1)
        data = add_column(data, 1)
        data[:, position + 2] = data[:, position] / data[:, position + 1]
        data = rsi(data, lookback_rsi_atr, position + 2, position + 3)
        data = delete_column(data, position, 3)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback_rsi refers to the number of past time periods to consider in the calculation of the RSI including the current one, the variable lookback_atr refers to the number of past time periods to consider in the calculation of the ATR including the current one, the variable lookback_rsi_atr refers to the number of past time periods to consider in the calculation of the RSI-ATR including the current one, the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

I have taken the liberty to present signal charts directly as opposed to just the indicator because I believe that the RSI-ATR is a high potential indicator and needs to be studied thoroughly. The signals are adaptive which means they may work in more market regimes especially when combined with the current trend.

Figure 10-3 GBPUSD hourly values with the 14-period RSI-ATR.

To summarize, the RSI-ATR is a contrarian indicator that combines the RSI and the ATR to deliver a volatility-weighted RSI adapted to more market regimes. I recommend you use it carefully as volatility can be tricky. Note that you can use the divergence technique but be careful as volatility changes may impact the predictability since between the two tops or troughs, there can be a huge fluctuation.

CHAPTER 11 THE DIRECTIONAL PROBABILITY INDEX

The directional probability index (DPI) is a simple two-in-one calculation which has the following goals:

- **Inflection point detection**: Important with regards to the book's topics.
- **Regime detection**: Less relevant for the book's main topic[37].

The first step is to calculate the percentage of times when the close price was higher than the open and divide it by a lookback period (n) which in this case is 14 as is the case in many other indicators.

$$DPI_i = \frac{Number\ of\ bullish\ price\ bars_{i-n:i}}{n}$$

The above formula will give you a probability of the on-going market regime. It is obvious that the indicator is very simple to calculate. Let's see how to use it. As the RSI and other indicators, you can simply use the extremes technique where values around 80% signal a bearish reaction and values around 20% signal a bullish reaction.

To detect the market regime, you can search for readings above 50% to interpret the current dynamic as being bullish and readings below 50% to interpret the current dynamic as being bearish.

[37] The book mainly discusses how to fade a trend or an initial move. Regime detection is mostly related to trend following techniques and strategies which I have presented in my previous book: Kaabar (2021) *Trend following strategies in Python*.

The function that you can use to code the DPI is as follows:

```
def directional_probability_index(data, lookback, open_column, close, position):
        data = add_column(data, 3)
        # Calculating the DPI
        for i in range(len(data)):
                if data[i, close] > data[i, open_column]:
                        data[i, position] = 1
        for i in range(len(data)):
                data[i, position + 1] = data[i - lookback + 1:i + 1, position].sum()
        data[:, position + 2] = (data[:, position + 1] / lookback) * 100
        data = delete_column(data, position, 2)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable open_column refers to the column of the open price, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

Figure 11-1 GBPUSD hourly values with the 14-period DPI.

To summarize, the DPI is an extremely simple indicator that not only gives trading signals but an idea on the strength of the current trend and a gauge for the position of bullish bars relative to the bearish and neutral bars.

I have found that the divergence technique works well using this indicator and hence, a real time divergence scanner may be of help in the overall trading framework.

CHAPTER 12 THE PARABOLIC RELATIVE STRENGTH

This indicator is a bit unusual, but I have chosen to present it because it is very interesting, and it is a good example of a structured indicator[38]. Before I present the indicator, I need to discuss the parabolic stop-and-reverse which is the basic ingredient of the parabolic relative strength (PRS).

The parabolic stop-and-reverse (PSAR) is an interesting trend-following overlay[39] indicator created by Welles Wilder Jr.[40] This indicator is mostly used as a trailing stop that tracks the trend as it develops. It is worth noting that it performs relatively well in steady trends but just as any other indicator, it has its weakness, in this case, ranging markets.

The basic understanding is that when the PSAR is under the current price, then the outlook is bullish and when it is above the current price, then the outlook is bearish. The PSAR function is lengthy which is why I have decided it to only put it in the GitHub repository. You can access it through the link[41] or through the below QR code.

[38] An indicator created by combining other indicators. Another example of a structured indicator would be the stochastic-RSI.
[39] Charted alongside the market price as opposed to indicators that need their own panel due to a difference of axis.
[40] The creator of the RSI and the ATR.
[41] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

The PRS is simply an RSI applied on the values of the parabolic stop-and-reverse.

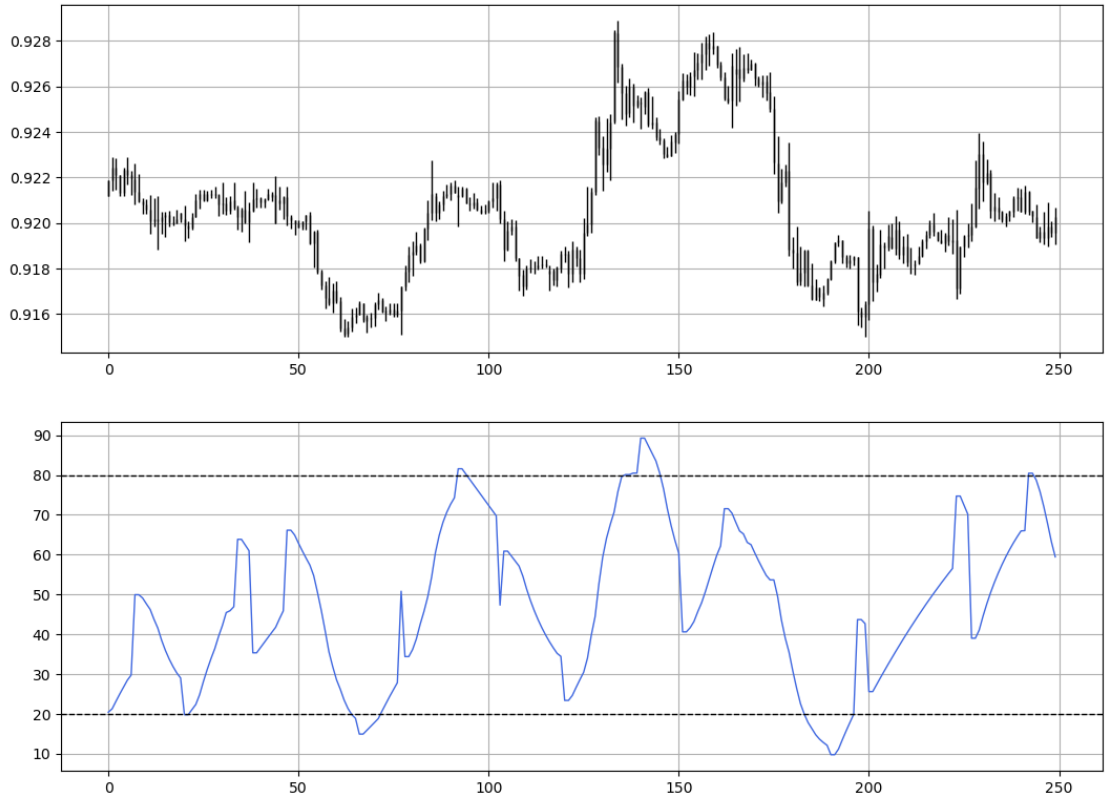


Figure 12-1 GBPUSD hourly values with the 14-period parabolic RSI.

You can notice the different way the PRS behaves from the default RSI however, the functioning remains the same with regards to techniques. I recommend using the conservative technique with the PRS. I also recommend widening the normality interval to 20 as the oversold level and 80 as the overbought level.

You can also notice that whenever the PRS changes course (shapes a U-turn), it tends to stick to the new direction. This can also be a technique that can be applied. As every other technique, it has its shortcomings.

The trading conditions of the U-turn technique are as follows:

- **A long signal is generated whenever the PSAR has a value greater than the previous one while the previous one being lower than the one prior to it.**
- **A short signal is generated whenever the PSAR has a value lower than the previous one while the previous one being greater than the one prior to it.**

I am sure that up to this point, you have managed to understand how I present the signal functions and how I structure the conditions, which is why I have decided to let you write your own signal function for the U-turn technique. Remember the following points:

- The signal function is a set of conditions between the current environment and the past environment.
- The indents in the code must be respected so that the code does what it is supposed to do.
- Remember to end with a return statement so that you can add columns in the signal function.

If you face difficulties with the signal function, you can contact me for the solution, but I am quite sure you will be able to write it and back-test it on your own. I am also sure that you will be able to create your own technical indicator and your own set of rules.

CHAPTER 13 THE CHANDE MOMENTUM OSCILLATOR

Created by Tushar Chande, the Chande momentum oscillator (CMO) is a contrarian indicator that uses a simple formula to show the market's momentum. To calculate the oscillator, use the following steps:

- Calculate the sum of the close prices that have closed higher during a lookback period, typically 14.
- Calculate the sum of the close prices that have closed lower during a lookback period, typically 14.
- Divide the difference between the results from step 1 and 2 by their sum and multiply by 100. Mathematically, this step is expressed using this formula:

$$CMO_i = \frac{Sum\ of\ higher\ closes_{i-n:i} - Sum\ of\ lower\ closes_{i-n:i}}{Sum\ of\ higher\ closes_{i-n:i} + Sum\ of\ lower\ closes_{i-n:i}}\ x\ 100$$

The next Figure illustrates a 14-period CMO with the hourly values of EURUSD.

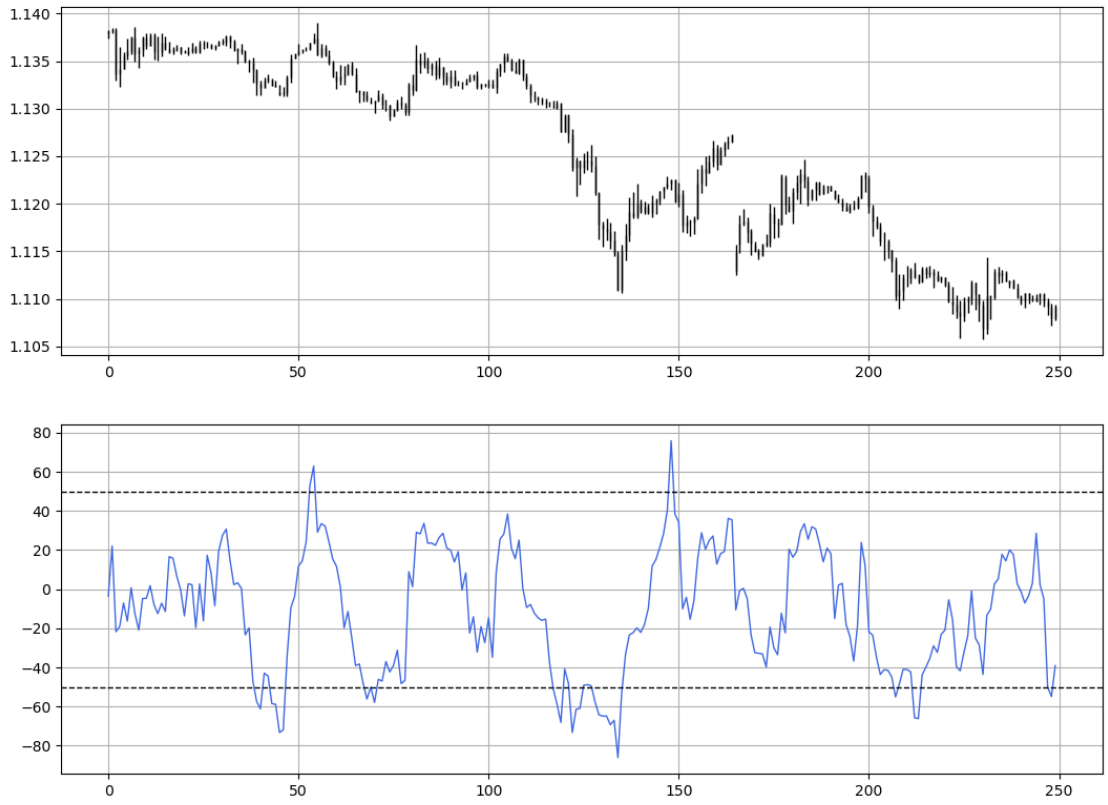


Figure 13-1 EURUSD hourly values with the 14-period CMO.

Generally, the indicator has implied boundaries at 50 and -50 but this can be personalized to each market and according to the current regime. For example, you can follow the trend of a bearish market by lowering the 50 upper barrier to 40 where you can add to your shorts whenever the market comes back around that area. The trading conditions by default are:

- **A long signal is generated whenever the CMO reaches the oversold level generally set at -50.**
- **A short signal is generated whenever the CMO reaches the overbought level generally set at 50.**

The function that you can use to code the CMO is as follows:

```
def chande_momentum_oscillator(data, lookback, close, position):
    data = add_column(data, 5)
    # Calculating the number of higher closes
    for i in range(len(data)):
        if data[i, close] > data[i - 1, close]:
            data[i, where] = data[i, close] - data[i - 1, close]
    # Calculating the number of lower closes
    for i in range(len(data)):
        if data[i, close] < data[i - 1, close]:
            data[i, where + 1] = abs(data[i, close] - data[i - 1, close])
    # Calculating the sum of higher closes
    for i in range(len(data)):
        data[i, where + 2] = data[i - lookback + 1:i + 1, where].sum()
    # Calculating the sum of lower closes
    for i in range(len(data)):
        data[i, where + 3] = data[i - lookback + 1:i + 1, where + 1].sum()
    # Calculating the CMO
    for i in range(len(data)):
        data[i, where + 4] = (data[i, where + 2] - data[i, where + 3]) / (data[i,
                                    where + 2] + data[i, where + 3]) * 100
    data = delete_column(data, 4, 4)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable close refers to the column of the close

price, and the variable position refers to the column where you place the indicator.

The next Figure shows the hourly ETHUSD values versus the 14-period CMO. Notice how the oscillator keeps breaking and surpassing the implied oversold and overbought levels by default. This market may be more suited with a wider interval, for instance -80 and 80.

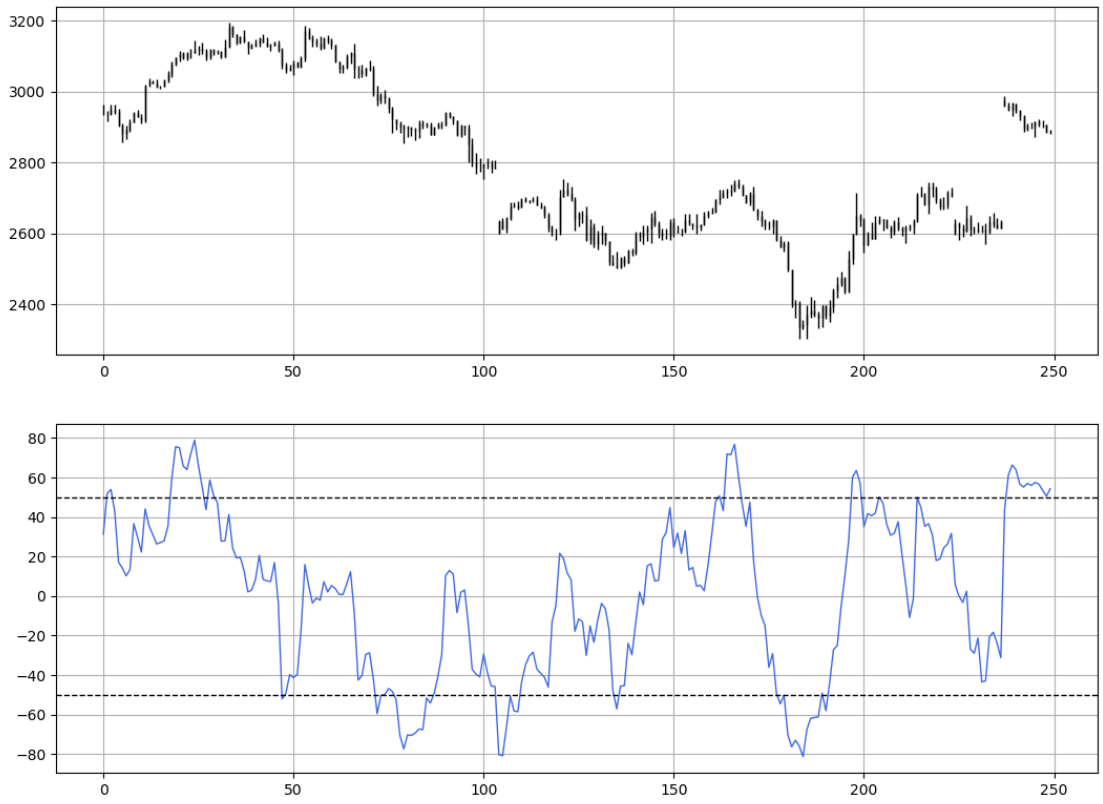


Figure 13-2 ETHUSD hourly values with the 14-period CMO.

To summarize, the CMO is an interesting contrarian indicator that uses a simple formula. I recommend you use in with other more known indicators like the RSI. You can apply the divergence technique using this indicator.

PART 3 K'S INDICATORS: NEW TECHNICAL HORIZONS

The K's indicators are a collection of my personal best indicators which I have created and optimized in order to add predictive value. Even though this book deals with contrarian indicators, I have chosen to include the full collection of K's indicators, be they contrarians or trend following.

Therefore, this part will deal with a collection of indicators that can be used together to form full trading systems that are generally uncorrelated to the primary and secondary indicators. Some of the K's indicators are a special combination of primary indicators while others are new calculations from scratch. Their utility is that they add a confirmation factor, but they can also be used as protagonists replacing the primary indicators.

Personally, I still use the RSI from time to time, but I have grown away from classical[42] indicators and have found that I have been principally using custom-made indicators such as the ones coming from the K's collection.

Combinations among the K's collection are also recommended as methods that seek to find levels such as **K's objective support and resistance levels** can be combined with indicators such as **K's envelopes** or **K's Fibonacci moving average**. Therefore, being creative and back-testing innovative ideas is the best you can do in your research and trading journey.

[42] Primary and secondary.

CHAPTER 14 K'S REVERSAL INDICATOR

This is a structured indicator composed of two famous classical indicators, Bollinger bands and the MACD which I have not presented in this book because it is a trend following indicator. I always say that using different indicators intelligently can help find superior trading setups which have above average predictability.

The K's reversal indicator is used to detect market tops and bottoms using a MACD cross technique and a Bollinger extreme. I have previously defined what Bollinger bands are, but I have not discussed the MACD yet.

An abbreviation for moving average convergence divergence, the MACD is one of the most common technical indicators and is one of the most known oscillators in technical analysis. Many people also consider it a trend-following indicator, but others use graphical analysis on it to find reversal points, making the MACD a versatile indicator. How is the MACD calculated?

It is the difference between the 26-period exponential moving average[43] applied to the close price and the 12-period exponential moving average also applied to the close price. The value found after taking the difference is called the MACD line. The 9-period exponential moving average of the MACD line is called the MACD signal (signal line). Therefore, the MACD oscillator is composed of two lines, the MACD line and the MACD signal.

[43] A type of weighted moving average. The exponential moving average uses a formula that gives more weight to the recent values so that the average is closer to the present environment. The code can be found in the Master_Function.py file in the GitHub repository.

The function to calculate the MACD is as follows:

```
def macd(data, close, long_ema, short_ema, signal_ema, position):
        data = add_column(data, 1)
        data = ema(data, 2, long_ema,  close, position)
        data = ema(data, 2, short_ema, close, position + 1)
        data[:, position + 2] = data[:, position + 1] - data[:, position]
        data = delete_row(data, long_ema)
        data = ema(data, 2, signal_ema, position + 2, position + 3)
        data = delete_column(data, position, 2)
        data = delete_row(data, signal_ema)
        return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price, the variable long_ema refers to the number of past time periods to consider in the calculation of the first exponential moving average, the variable short_ema refers to the number of past time periods to consider in the calculation of the second exponential moving average, the variable signal_ema refers to the number of past time periods to consider in the calculation of the signal line, and the variable position refers to the column where you place the indicator.

Reminder

Remember that a long position is synonymous to a buy position, but a short position is not really a sell order as it is merely a position on the other side where you borrow the asset from a party, sell it to a third party, and then buy it back at the market price to return it to the original party. This way, you can profit from a price decrease.

The way to use the MACD relies on a cross between the MACD line and its signal line and has the following conditions:

- **A long signal is generated but not confirmed whenever the MACD surpasses the signal line.**
- **A short signal is generated but not confirmed whenever the MACD breaks the signal line.**

Remember that the signal line is simply a 9-period exponential moving average applied on the MACD. K's reversal indicator is therefore composed of the cross technique I have just mentioned and the position of the price relative to a 100-period Bollinger bands with a standard deviation set to 2. The K's reversal indicator has the following trading conditions:

- **A long signal is generated whenever the MACD line surpasses its signal line while the current market price is below the lower 100-period Bollinger band.**
- **A short signal is generated whenever the MACD line breaks its signal line while the current market price is above the upper 100-period Bollinger band.**

Figure 14-1 illustrates the signals generated on the K's reversal indicator. Note that the indicator is superimposed on the price chart through its buy and sell signals.

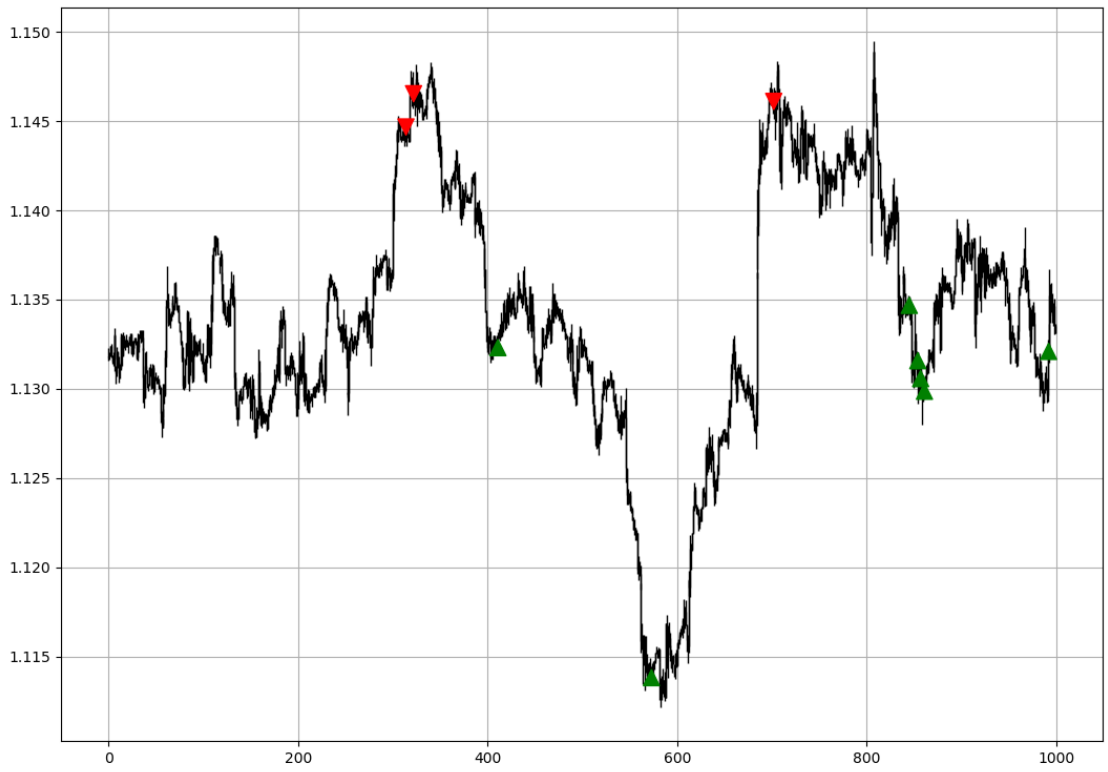


Figure 14-1 EURUSD hourly values with K's reversal indicator.

The previous chart shows the signals generated using the K's reversal indicator. For each signal, there was a price outside one of the Bollinger bands and at the same time its MACD is crossing its signal line.

I have found that quite often, the indicator can signal major tops and bottoms on the short-term horizon, but I generally use it in a ranging regime to confirm the trading ideas that are based on simple reactions.

The indicator seems to be useful with regards to the following points:

- The signals are based on the confirmation method where two powerful indicators are used to validate the signal. This gives it a double confirmation effect.
- The signals are not very successive and not very frequent, thus eliminating the need to filter them.
- The signals can be used in a trending market and a ranging market with the latter having a better success ratio at predicting tops and bottoms.
- It is uncorrelated with classical indicators such as the RSI.

The indicator is limited when it comes to the following points:

- There are no clear exit rules that work well on average across the markets. Even though K's reversal indicator gives contrarian signals, it does not show when to exit the positions.
- As with other indicators, it underperforms on some markets and is not to be used everywhere.
- False signals tend to occur during trending markets but there is no proven way to detect a false signal.

Of course, by time, the indicator shows some best practices. I have found that a signal must be invalidated whenever the following conditions are realized:

- **For a long signal to be invalidated, the market must break below double the difference between the high and low of the signal candle projected starting from the low of the signal candle.**
- **For a short signal to be invalidated, the market must surpass double the difference between the high and low of the signal candle projected starting from the high of the signal candle.**

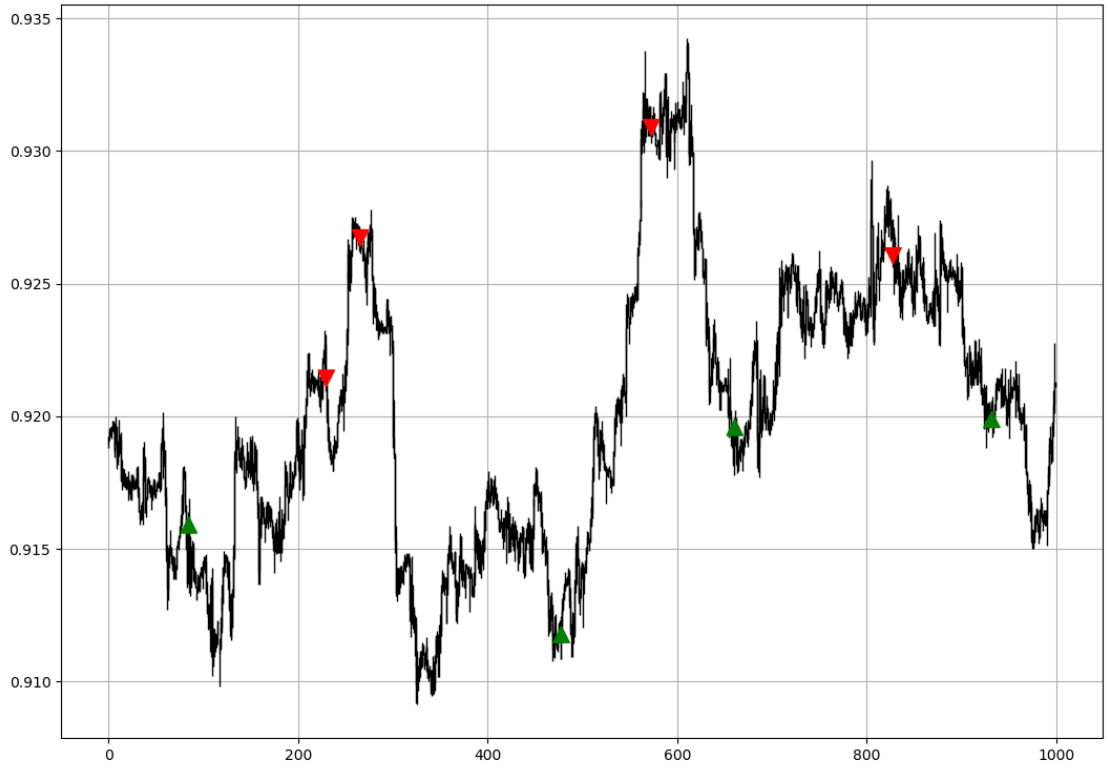


Figure 14-2 USDCHF hourly values with K's reversal indicator.

Here is an example to make things clearer. Consider that on the close of the candlestick, the signal is given. You have the following information upon the open of the next candlestick:

- Previous open: $1,000
- Previous high: $1,050
- Previous low: $995
- Previous close: $1,025
- Current open: $1,025

The first thing you do is to calculate the difference between the last high and low which is $55 and then double it which gives you $110. Therefore, if the market breaks $885, you can say that the signal has been invalidated. Note that

I have found $885 as the difference between $995 and $110. By default, you should have a 2.0 risk-reward ratio which implies that you can target two times the stop. Does this mean that by putting a stop at $885, you should target $1,305? Not really, because I have found that it is generally better to exit on your own terms as many target techniques that I have tried gave out mixed results, therefore, the target really depends on your preferences.

Why did I say that you should target $1,305? Remember that if you open a position on the open which is at $1,025 and place your stop at $885, then if you double the distance between the two to place your target, you will find that your target should be at $1,305.

The code for K's reversal indicator can be written down as follows:

```
def k_reversal_indicator(data, lookback_boll, standard_deviation, close, position):
    data = bollinger_bands(data, lookback_boll, standard_deviation, close, position)
    data = macd(data, close, 26, 12, 9, position + 3)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback_boll refers to the lookback period of the Bollinger bands, the variable standard_deviation refers to the multiplier of volatility, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

The signals generated from the indicator are therefore coded using this syntax:

```
def signal(data, open_price, close, upper_boll, lower_boll, macd_col, signal_col, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if min(data[i, open_price], data[i, close]) < data[i, lower_boll] and data[i,
          macd_col] > data[i, signal_col] and data[i - 1, macd_col] < data[i - 1,
          signal_col]:
            data[i + 1, buy] = 1
        # Bearish signal
        elif max(data[i, open_price], data[i, close]) > data[i, upper_boll] and data[i,
            macd_col] < data[i, signal_col] and data[i - 1, macd_col] > data[i - 1,
            signal_col]:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable open_price refers to the column of the open price, the variable close refers to the column of the close price, the variable upper_boll refers to the column of the upper Bollinger band, the variable lower_boll refers to the column of the lower Bollinger band, the variable macd_col refers to the column of the MACD line, the variable signal_col refers to the column of the signal line, and the variables buy and sell refer to the columns containing bullish and bearish signals.

To summarize, my recommendation is to use this indicator in conjunction with other indicators and techniques, preferably pure price action strategies such as graphical analysis and pivot point analysis (a technique I will present in the last part of the book).

CHAPTER 15 K'S OBJECTIVE SUPPORT & RESISTANCE LEVELS

Detecting support and resistance levels can be tricky and subjective. Technical analysis provides you with many ways to find levels such as simply tracing horizontal and trend lines on the market price. Similarly, other techniques can be used to find demand zones (support levels) and supply zones (resistance levels). In the most basic form:

- **A support level** is below the current market price, and it is supposed to be a floor from where the market bounces if it approaches it. It is recommended to buy close to a support level.
- **A resistance level** is above the current market price, and it is supposed to be a ceiling from where the market consolidates if it approaches it. It is recommended to sell close to a resistance level.

I have created this indicator to find objective support and resistance levels using the concept of maximum range intervals. Visually, K's support and resistance levels are a form of envelopment that look like a more stable version of Bollinger bands. To calculate the levels, follow these steps:

- Calculate the maximum range which is found through this formula:

$$Maximum\ range_i = max(High_{i-n:i}) - min\ (Low_{i-n:i})$$

- To find K's objective support, use this formula:

$$Support\ level_i = min(Low_{i-n:i}) + (Maximum\ range_i\ x\ 0.05)$$

- To find K's objective resistance, use this formula:

$$Resistance\ level_i = max(High_{i-n:i}) - (Maximum\ range_i\ x\ 0.05)$$

```
def k_objective_support_resistance(data, lookback, lower_range, high, low, position):
    data = add_column(data, 3)
    # Calculating the maximum range
    for i in range(len(data)):
        try:
            data[i, position] = (max(data[i - lookback + 1:i + 1, high]) –
                                 min(data[i - lookback + 1:i + 1, low]))
        except ValueError:
            pass
    # Calculating support
    for i in range(len(data)):
        try:
            data[i, position + 1] = min(data[i - lookback + 1:i + 1, low]) +
                                    (data[i, position] * lower_range)
        except ValueError:
            pass
    # Calculating resistance
    for i in range(len(data)):
        try:
            data[i, position + 2] = max(data[i - lookback + 1:i + 1, high]) –
                                    (data[i, position] * lower_range)
        except ValueError:
            pass
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical data, the variable lookback refers to the number of time periods to consider in the calculation of the maximum range, the variable lower_range refers to the percentage of inclusion, the variable high refers to the column of the highs, the variable low refers to the column of the

lows, and the variable position refers to the column where you place the indicator.

Figure 15-1 S&P500 hourly values with K's objective support and resistance levels.

The trading conditions are as follows:

- **A long signal is generated whenever the market reaches the lower part of the range.**
- **A short signal is generated whenever the market reaches the upper part of the range.**

The signal function can be written down as follows in the next code snippet.

```
def signal(data, close, k_support, k_resistance, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        try:
            # Bullish signal
            if data[i, close] > data[i, k_support] and data[i - 1, close] < data[i - 1, k_support]:
                data[i + 1, buy] = 1
            # Bearish signal
            elif data[i, close] < data[i, k_resistance] and data[i - 1, close] > data[i - 1,
                k_resistance]:
                data[i + 1, sell] = -1
        except IndexError:
            pass
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price, the variable k_support refers to the column of the support level, the variable k_resistance refers to the column of the resistance level, and the variables buy and sell refer to the columns containing bullish and bearish signals.

I have found that a signal must be invalidated whenever the following conditions are realized:

- **For a long signal to be invalidated, the market must print two successive close prices below the support that has generated the signal.**
- **For a short signal to be invalidated, the market must print two successive close prices above the resistance that has generated the signal.**

To summarize, K's objective support and resistance levels generally work great in sideways markets but underperform in trending markets.

I recommend using it and filtering the signals using a trend indicator so that you only take buy signals (approaching a support) in a bullish market and sell signals (approaching a resistance) in a bearish market. Also, they are better used when they are flat as this signals a flat market a better probability of reacting above the support or below the resistance.

CHAPTER 16 K'S FIBONACCI MOVING AVERAGE

The Fibonacci sequence is a famous mathematical series that uses a simple formula to calculate the next value. Originally discovered by Leonardo Bonacci also known as Leonardo Fibonacci, the sequence follows this distinct pattern:

$$F_n = F_{n-1} - F_{n-2}$$

This means that the current value is the sum of the previous two values. Therefore, it is an infinite series that keeps growing exponentially:

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, \ldots.$$

The Fibonacci sequence presents a deep field of technical studies that is outside the scope of this book. Typically, you use the Fibonacci sequence to get ratios and retracements, but I will show that in a later chapter. For the moment, I will stick to the simple sequence shown above.

At this stage, I have presented the concept of moving averages. The Fibonacci moving average is based on both simple and exponential moving averages and is constructed by the following these steps:

- Calculate exponential moving averages using the following lookback periods {5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181} on the highs and the lows of the underlying. This should give you 30 different calculations (columns).
- For each time step (row), take the simple moving average of all 15 exponential moving averages of the highs and all 15 exponential moving averages of the lows.

Therefore, you will basically be calculating a simple moving average on the exponential moving averages which are based on the Fibonacci sequence. Now, another particularity with the Fibonacci moving average is that it is calculated

as a zone. This means that the calculation is done on the highs and the lows instead of just the closing prices. By doing so, you will have two lines representing dynamic support and resistance zones. Take a look at the next chart of the EURUSD showing the Fibonacci moving average.

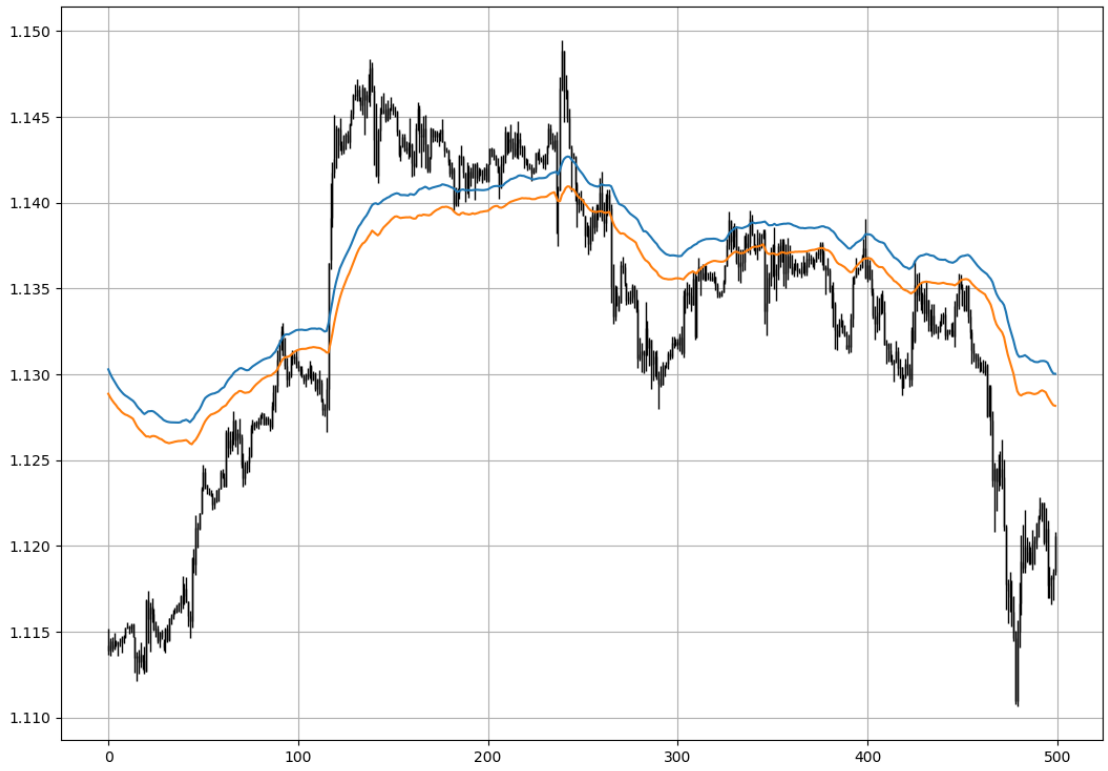


Figure 16-1 EURUSD hourly values with K's Fibonacci moving average.

The chart shows the market's reaction close to the Fibonacci moving average. The idea is to anticipate contrarian reactions every time the market enters the zone.

Generally, whenever the market enters the zone, you should anticipate a contrarian reaction. For example, a market in an uptrend that encounters the Fibonacci moving average above it should pause or even reverse course.

The trading conditions are as follows:

- **A long signal is generated whenever the market price enters the Fibonacci moving average zone from the above. This is in anticipation of finding a strong support level.**
- **A short signal is generated whenever the market price enters the Fibonacci moving average zone from the below. This is in anticipation of finding a strong resistance level.**

Since the code is a bit lengthy, I have decided not to add it to the book, but the full code is present in the official GitHub repository[44]. The signal function however has the following syntax. Remember that the indicator's function is used to calculate and call the indicator and the signal function gives the condition on how to use it.

```
def signal(data, high, low, close, upper_ma, lower_ma, buy, sell):
        data = add_column(data, 10)
        for i in range(len(data)):
        if data[i, low] < data[i, upper_ma] and data[i, low] > data[i, lower_ma] and
          data[i - 1, low] > data[i - 1, upper_ma] and data[i, close] < data[i, upper_ma]
          and data[i, buy] == 0:
                data[i + 1, buy] = 1
        elif data[i, high] < data[i, upper_ma] and data[i, high] > data[i, lower_ma] and
            data[i - 1, high] < data[i - 1, lower_ma] and data[i, close] > data[i, lower_ma]
            and data[i, sell] == 0:
                data[i + 1, sell] = -1
        return data
```

[44] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable high refers to the column of the highs, the variable low refers to the column of the lows, the variable close refers to the column of the close price, the variable upper_ma refers to the column of the upper Fibonacci moving average, the variable lower_ma refers to the column of the lower Fibonacci moving average, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 16-2 GBPUSD hourly values with K's Fibonacci moving average.

Notice how I am trying to generate signals each time the market approaches and enters the moving average zone, thus benefitting from a quick or extended reaction. The indicator seems to be useful with regards to the following points:

- The signals are based on an established and well-known indicator that is the moving average and therefore benefits from the greater impact effect.
- The signals are not very successive and not very frequent thus eliminating the need to filter them.

The indicator has limitations when it comes to the following points:

- There are no clear exit rules that work well on average across the markets. Even though the Fibonacci moving average gives contrarian signals, it does not show when to exit the positions.
- As with other indicators, it underperforms on some markets and is not to be used everywhere.
- False signals are abundant in sideways markets.

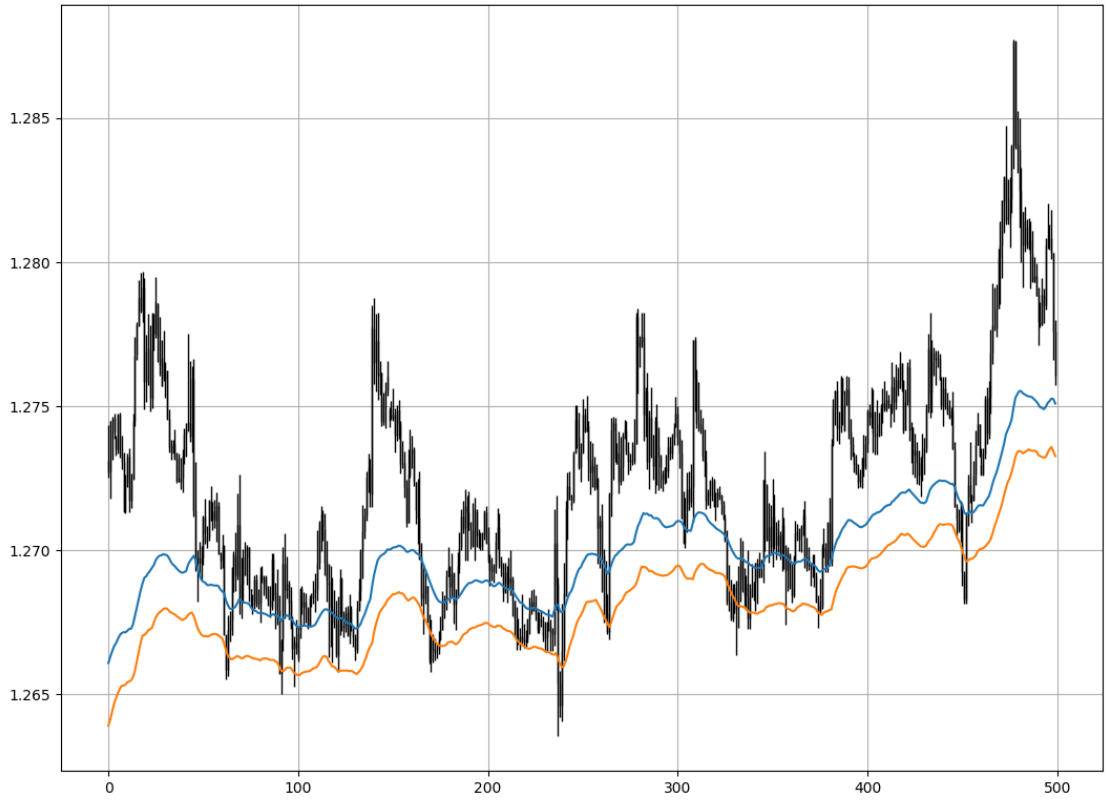


Figure 16-3 USDCAD hourly values with K's Fibonacci moving average.

I have found that a signal must be invalidated whenever the following conditions are realized:

- **For a long signal to be invalidated, the market must break below the distance between the upper and lower moving averages projected from the lower Fibonacci moving average.**
- **For a short signal to be invalidated, the market must break above the distance between the upper and lower moving averages projected from the upper Fibonacci moving average.**

Here is an example to make things clearer. Consider that on the close of the candlestick, the bullish signal is given. You have the following information upon the open of the candlestick:

- Previous upper Fibonacci moving average: $1,000
- Previous lower Fibonacci moving average: $995
- Current open: $997

The first thing to do is to calculate the difference between the upper and lower Fibonacci moving averages which in this case is $5, your stop therefore can be $995 - $5 = $990 and with an open at $997, you should target no less than $14 (target level at $1,011) to benefit from a superior risk-reward ratio.

Remember, by default you should have a 2.0 risk-reward ratio which implies that you can target two times your stop. Of course, this is just one example of a trade setting among many others.

To summarize, K's Fibonacci moving average is a moving zone that relies on the Fibonacci sequence. The aim is to generate contrarian reactions from the zone. I recommend using this indicator with other contrarian tools such as the RSI or K's reversal indicator.

CHAPTER 17 K'S ENVELOPES

K's envelopes are a set of 800-period simple moving averages where one is applied on the highs and the other is applied on the lows thus forming a moving zone around the market price. As simple as this indicator is, I consider it as one of my personal favorite ones. Even though it is made up of moving averages, K's envelopes are not a trend following indicator. The trading conditions are as follows:

- **A long signal is generated whenever the market price enters the envelope zone from the above. This is in anticipation of finding a strong support.**
- **A short signal is generated whenever the market price enters the envelope zone from the below. This is in anticipation of finding a strong resistance.**

To code K's envelopes, you can write down this very basic syntax on the condition that you have previously defined the moving average function:

```
def k_envelopes(data, lookback, high, low, position):
    # Calculating the upper moving average
    data = ma(data, lookback, high, position)
    # Calculating the lower moving average
    data = ma(data, lookback, low, position + 1)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable high refers to the column of the high

price, the variable low refers to the column of the low price, and the variable position refers to the column where you place the indicator.

The next chart shows the EURUSD hourly values with K's envelopes.

Figure 17-1 AUDUSD hourly values with K's envelopes.

The trading conditions of K's envelopes are the same as the ones of K's Fibonacci moving average, hence you can refer to them in the previous chapter.

Figure 17-2 EURUSD hourly values with K's envelopes.

The envelopes seem to be useful with regards to the following points:

- The signals are based on an established and well-known indicator that is the moving average and therefore benefits from the greater impact effect.
- The signals are not very successive and not very frequent thus eliminating the need to filter them.

The indicator has limitations when it comes to the following points:

- There are no clear exit rules that work well on average across the markets. Even though K's envelopes give contrarian signals, they do not show when to exit the positions if taken.

- As with other indicators, they underperform on some markets and are not to be used everywhere.
- False signals are abundant in sideways markets.

Figure 17-3 USDCHF hourly values with K's envelopes.

I have found that a signal must be invalidated whenever the following conditions are realized:

- **For a long signal to be invalidated, the market must break below the distance between the upper and lower moving averages projected from the lower moving average.**
- **For a short signal to be invalidated, the market must break above the distance between the upper and lower moving averages projected from the upper moving average.**

Here is an example to make things clearer. Consider that on the close of the candlestick, the bearish signal is given. You have the following information upon the open of the candlestick:

- Previous upper moving average: $92
- Previous lower moving average: $90
- Current open: $91

The first thing to do is to calculate the difference between the upper and lower moving averages which in your case is $2, your stop therefore can be $90 - $2 = $88 and with an open at $91, you should target no less than $6 (target level at $97) to benefit from a superior risk-reward ratio.

CHAPTER 18 K'S VOLATILITY BANDS

Volatility bands as you have seen previously with Bollinger bands are very interesting framing indicators. They give statistically defined levels that tell you that the market price should find support or resistance depending on its current position. K's volatility bands are simply a more reactive type of bands that use a different formula than the Bollinger bands.

To code K's volatility bands, you can follow the below steps:

- Calculate the maximum value of the highs using a lookback period of 20.
- Calculate the minimum value of the lows using a lookback period of 20.
- Calculate the mean between the two for every time step (row). Let's call this the middle line.
- Calculate the 20-period standard deviation of the close price.
- Calculate the maximum value of the volatility from the previous step. Let's call this the maximum volatility.
- To calculate K's volatility bands, use this formula:

$$K's\ Lower\ Band = Middle\ line - (x.Maximum\ volatility)$$

$$K's\ Upper\ Band = Middle\ line + (x.Maximum\ volatility)$$

Therefore, K's volatility bands use two key concepts which are the middle line and the maximum volatility. The former is calculated as being the mean between the highest highs and lowest lows for a certain lookback period while the latter is simply the highest standard deviation value (applied on the close price).

```
def k_volatility_band(data, lookback, multiplier, high, low, close, position):
    data = add_column(data, 6)
    # Calculating the middle line
    for i in range(len(data)):
        try:
            data[i, position] = max(data[i - lookback + 1:i + 1, high])
            data[i, position + 1] = min(data[i - lookback + 1:i + 1, low])
            data[i, position + 2] = (data[i, position] + data[i, position + 1]) / 2
        except ValueError:
            pass
    data = delete_column(data, position, 2)
    # Calculating maximum volatility
    data = volatility(data, lookback, close, position + 1)
    for i in range(len(data)):
        try:
            data[i, position + 2] = max(data[i - lookback + 1:i + 1, position + 1])
        except ValueError:
            pass
    data = delete_column(data, position + 1, 1)
    # Calculating the bands
    data[:, position + 2] = data[:, position] + (multiplier * data[:, position + 1])
    data[:, position + 3] = data[:, position] - (multiplier * data[:, position + 1])
    data = delete_column(data, position + 1, 1)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable multiplier is the volatility multiplier, the variable high refers to the column of the high price, the variable low refers

to the column of the low price, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

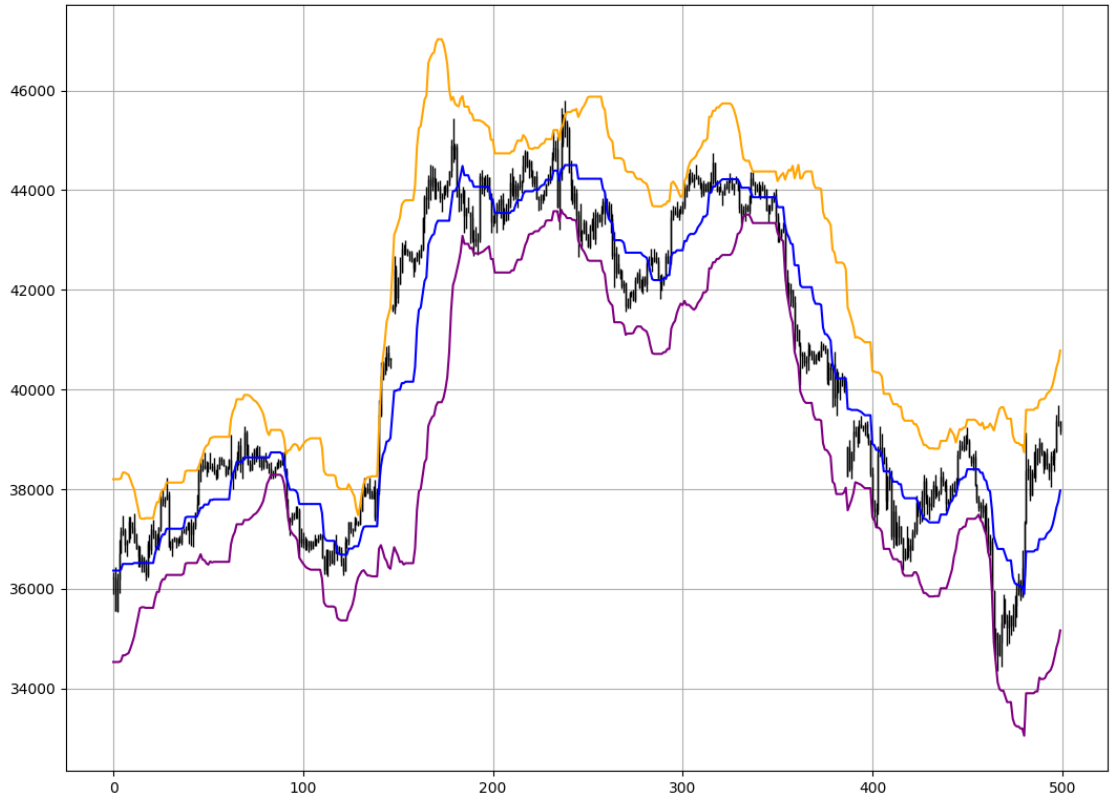


Figure 18-1 AUDUSD hourly values with K's volatility bands.

The basic technique of this indicator is as follows:

- **A long signal is generated whenever the market price reaches the lower volatility band.**
- **A short signal is generated whenever the market price reaches the upper volatility band.**

I recommend using the aggressive technique with K's volatility bands. The signal function is as follows:

```
def signal(data, close, upper_band, lower_band, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, close] < data[i, lower_band] and data[i - 1, close] > data[i - 1, lower_band]:
            data[i + 1, buy] = 1
        # Bearish signal
        elif data[i, close] > data[i, upper_band] and data[i - 1, close] < data[i - 1, upper_band]:
            data[i + 1, sell] = -1
    return data
```

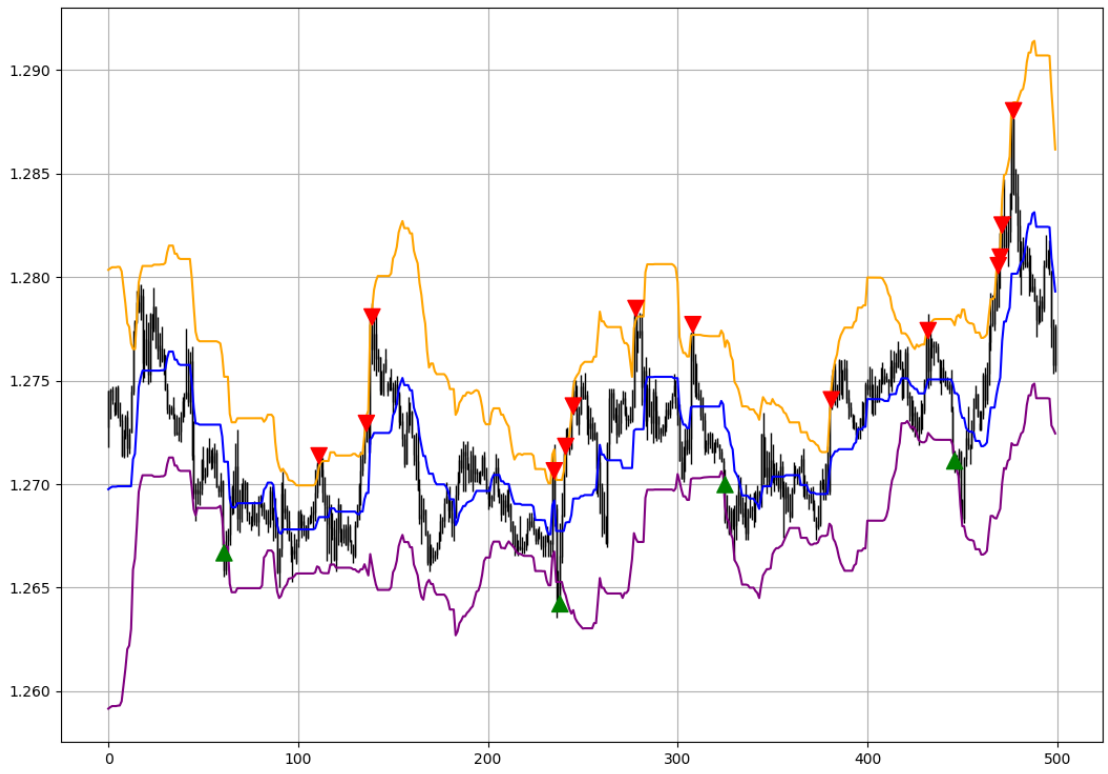


Figure 18-2 USDCAD hourly values with K's volatility bands.

K's volatility bands seem to be useful with regards to the following points:

- The signals are correlated to the well-known Bollinger bands which means it benefits partially from its impact.
- The signals are not very successive and not very frequent thus eliminating the need to filter them.
- Exit rules can be clearly established by exiting upon reaching the other band.

K's volatility bands have limitations when it comes to the following points:

- As with other indicators, they underperform on some markets and are not to be used everywhere.
- False signals are abundant in trending markets.

I have found that a signal must be invalidated whenever the following conditions are realized:

- **For a long signal to be invalidated, the market must print two successive close prices below the lower band.**
- **For a short signal to be invalidated, the market must print two successive close prices above the upper band.**

To summarize, K's volatility bands are an alternative to Bollinger bands and rely on the concept of the maximum range and the maximum volatility. I recommend using them in sideways markets only. They also seem to work well with classical indicators like the RSI and the stochastic oscillator.

CHAPTER 19 K'S FIBONACCI TIMING PATTERN

The Fibonacci sequence never ceases to impress me which is why it represents the basis of another of K's collection indicators. K's Fibonacci timing pattern combines the position of the market price relative to time in order to call a market top or bottom. The indicator is overlaid on the chart with upward pointing arrows signifying a buy pattern and downward pointing arrows signifying a sell pattern. The trading conditions are as follows:

- **A long signal is generated whenever the market prints 8 consecutive closes where each close is lower than the close from three periods ago and the close from five periods ago. In addition to this, the eighth bar must be lower than the seventh bar.**
- **A short signal is generated whenever the market prints 8 consecutive closes where each close is higher than the close from three periods ago and the close from five periods ago. In addition to this, the eighth bar must be higher than the seventh bar.**

The signal function of the divergence technique is lengthy which is why I have decided it to only put it in the GitHub repository. You can access it through the link[45] or through the below QR code.

[45] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

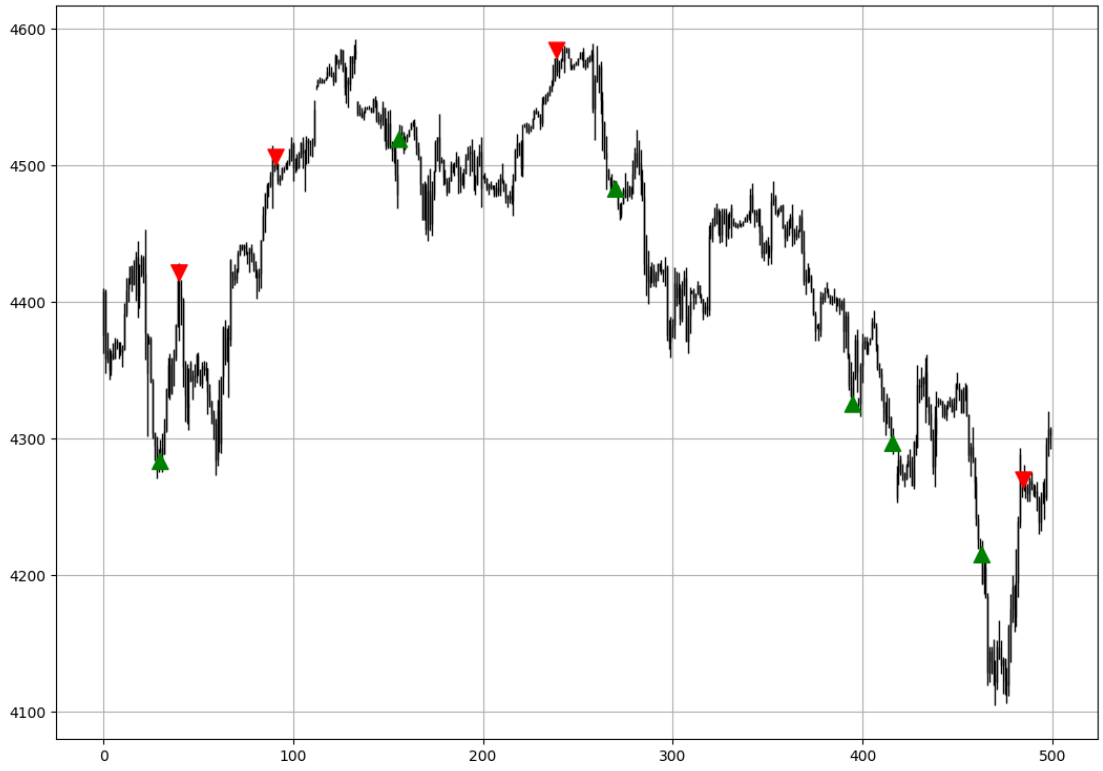


Figure 19-1 S&P500 hourly values with K's Fibonacci timing pattern.

K's Fibonacci timing pattern seems to be useful with regards to the following points:

- The signals are not very successive and not very frequent thus eliminating the need to filter them.
- The pattern works well during sideways market. It is mostly for short-term reactions.

K's Fibonacci timing pattern have limitations when it comes to the following points:

- As with other indicators, it underperforms on some markets.
- False signals are abundant in trending markets.

I have found that a signal must be invalidated whenever the following conditions are realized:

- **For a long signal to be invalidated, the market must print two successive close prices below the low of the signal bar.**
- **For a short signal to be invalidated, the market must print two successive close prices above the high of the signal bar.**

Figure 19-2 shows an example of signals generated on AUDUSD.

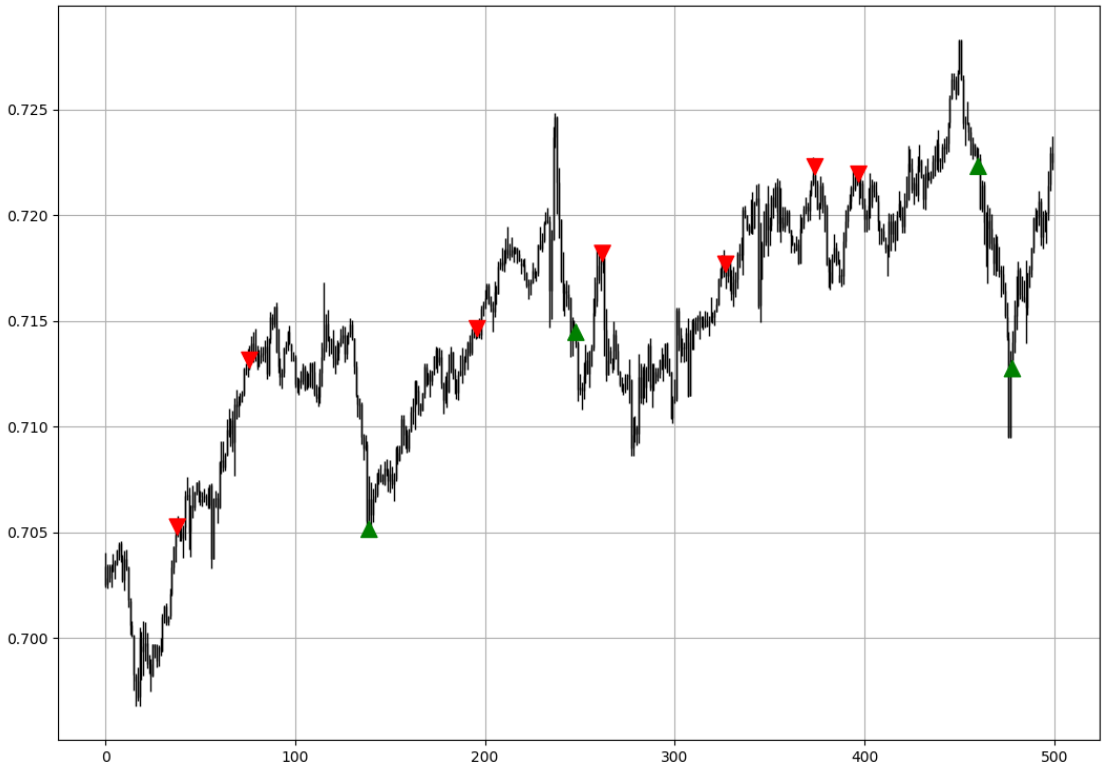


Figure 19-2 AUDUSD hourly values with K's Fibonacci timing pattern.

To summarize, K's Fibonacci timing pattern is as its name suggests, a timing pattern. I recommend using it with primary technical indicators such as the RSI and other indicators from K's collection such as the volatility bands.

PART 4 HANDS-ON TRADING STRATEGIES

You have finally reached the interesting part of the book, although the previous chapters have introduced a crude amount of knowledge which is defining, understanding, coding, and charting different types of indicators, you now need to see them in action by combining some of them in structured trading strategies that have the aim of filtering and improving the signals generated from individual indicators.

Before starting, let's take a look at how the strategies will be presented. First, I will not reintroduce the indicators all over again, therefore, I will just refer to them before presenting the trading rules.

It is important to understand that these are not ready-to-be-deployed strategies, they are merely raw suggestions on which you can choose to build your system on. A good trading framework based on one of the presented strategies will likely look completely different but will just have its roots coming from them. **Always do your own back-tests** because only you can know exactly what your risk and reward appetite is. Remember, applying strategies you find online or in books is unlikely to lead to sustainable profitable results but transforming these strategies by absorbing their main ideas will give you enough experience to maneuver whenever there is some turbulence in the markets. In other words, strategies do not work forever, and you must understand when to change your methods.

Before I move on to the strategies, I will show you how to evaluate them in a simple way. The following sections discuss very simple performance evaluation measures that will serve as general evaluators but of course, more detailed measures must be done to ensure the viability of the strategy.

The Holding Period

A variable holding period is based on the concept of exiting upon the next signal. This means that when you buy, you will hold the position until getting another signal whether bullish or bearish and only then you will close the position. Similarly, when you get a sell signal, you will remain bearish until getting a new signal whether bullish or bearish. Therefore, the result will be based on the difference between the price at exit and the entry price which is generally the open price after validating the signal on the previous close assuming normal execution.

Number of Trades

The number of trades reflects how frequent are the signals generated by the strategy. This gives us an idea on how many trades to expect which is a direct clue to transaction costs and the statistical significance of the back-testing results. There is no minimum or maximum number for this measure, it depends on the trader's appetite to take on trades.

The Hit Ratio

The hit ratio is simply the number of winning trades over the number of the trades taken in total. For example, if you have 1359 trades over the course of 5 years and you have been profitable in 711 of them, then your hit ratio is 711/1359 = 52.31%. This means that on average, for every 100 trades taken, 52 of them are expected to be winners. This measure is useless without risk management, and I will show why in the section dealing with the risk-reward ratio.

Profit Factor

This is a straightforward method that measures the total profit as a ratio of the total loss. Hence, you can say that the profit factor is how much you have gained compared to a $1 loss. If you consider that a strategy has earned you $50,000 this year in gross profits and caused you $20,000 in losses, then your profit factor is calculated as follows.

$$Profit\ Factor = \frac{50{,}000}{20{,}000} = 2.5$$

Therefore, for every $1 lost on the strategy, you have made $2.5 which is a sign of a good strategy in case you have adequate risk management.

Realized Risk-Reward Ratio

The realized risk-reward ratio measures how much risk you are taking for every trade realized. This means that you will divide the average gain by the average loss. A good strategy will have a realized risk-reward ratio of at least 1.80 – 2.00 so that you have enough margin with the hit ratio of the strategy.

$$Realized\ RiskReward\ Ratio = \frac{Average\ Gain}{Average\ Loss}$$

Trading is not about perfectly predicting the markets; success is dependent on how well you manage risk. Let's consider the below example:

- Average gain in 2020: $125.00
- Average loss in 2020: $75.00

$$Realized\ RiskReward\ Ratio = \frac{125}{75} = 1.67$$

The above results show that you are probably taking a little more risk than is generally acceptable. This does not mean that it is a bad strategy.

The hit ratio and the risk-reward ratio go hand in hand and successful strategies rely on a healthy trade-off between the two. Consider these two examples below:

- A strategy with a 90.00% hit ratio and a risk-reward ratio of 0.05 will unlikely result in any profits. This is because with a risk-reward ratio of 0.05, you would need a 95.23% hit ratio just to break-even assuming the same monetary size trades.
- A strategy with a 30.00% hit ratio and a risk-reward ratio of 4 will likely be profitable. This is because with a risk-reward ratio of 4, you will need a 20.00% hit ratio just to break-even assuming the same monetary size trades.

To calculate the break-even hit ratio, you can follow this below formula:

$$Breakeven\ Hit\ Ratio = \frac{1}{(1 + RiskReward\ Ratio)}$$

As weird as it looks, but sometimes winning 9 trades out 10 is less profitable than winning 3 trades out of 10 trades. This is one of the basics of risk management; knowing how much to risk and what to target.

The back-testing results that I present in this part measure the predictive nature until another signal is seen. Therefore, they do not reflect any complete trading framework as the latter must have more sophisticated entry and exit rules. I suggest you take the results with a lot of skepticism. In any way, the main takeaway of this part is to increase your horizons and to think outside the box.

STRATEGY #1 THE MACD SPECIAL DIVERGENCE

I have previously introduced the MACD oscillator and how to calculate it. This strategy uses the MACD to detect divergences to find trading signals. Naturally, when prices are going in a certain direction while a price-based indicator is going in the other, a weakening in the underlying trend is occurring and a possibility to change the bias becomes greater.

The special divergence technique uses the signal line on the MACD to validate the divergence thus removing the need for any lower and upper barriers. The trading conditions are:

- **A long signal is generated whenever the MACD surpasses its signal line, then breaks it while remaining higher than the first trough, and finally surpasses it. Of course, you must not forget about the most important condition, which is the market price making lower prices between the two troughs. Simultaneously, all the above conditions must occur while the MACD is below zero.**
- **A short signal is generated whenever the MACD breaks its signal line, then surpasses it while remaining lower than the first top, and finally breaks it. The market price must print higher prices between the two tops. Simultaneously, all the above conditions must occur while the MACD is above zero.**

As you know, the signal function of the divergence technique is lengthy which is why I have decided it to only put it in the GitHub repository.

Figure 1 shows an example of signals generated on USDCHF using the special divergence strategy.

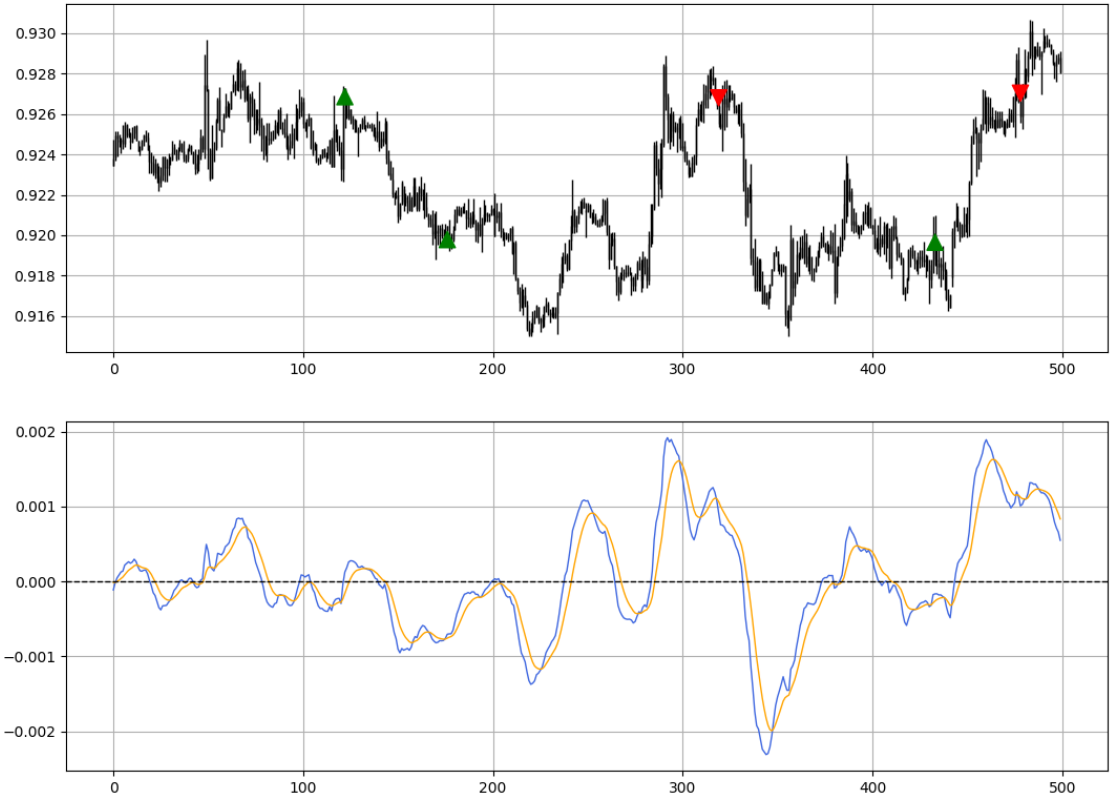


Figure 1 Signal chart on USDCHF.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020. The width is set to 60.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	52.54%	1.89	1.71	118
USDCHF	49.19%	1.10	1.14	124
GBPUSD	42.24%	0.93	1.28	116
AUDUSD	45.37%	1.45	1.75	108
USDCAD	34.00%	0.55	1.07	100
XAUUSD	53.06%	1.27	1.13	98
XAGUSD	49.01%	1.57	1.63	102
NIKKEI225	56.89%	2.62	1.99	58
S&P500	42.64%	1.75	2.35	68

The next Figure shows more signals generated on the S&P500 index.

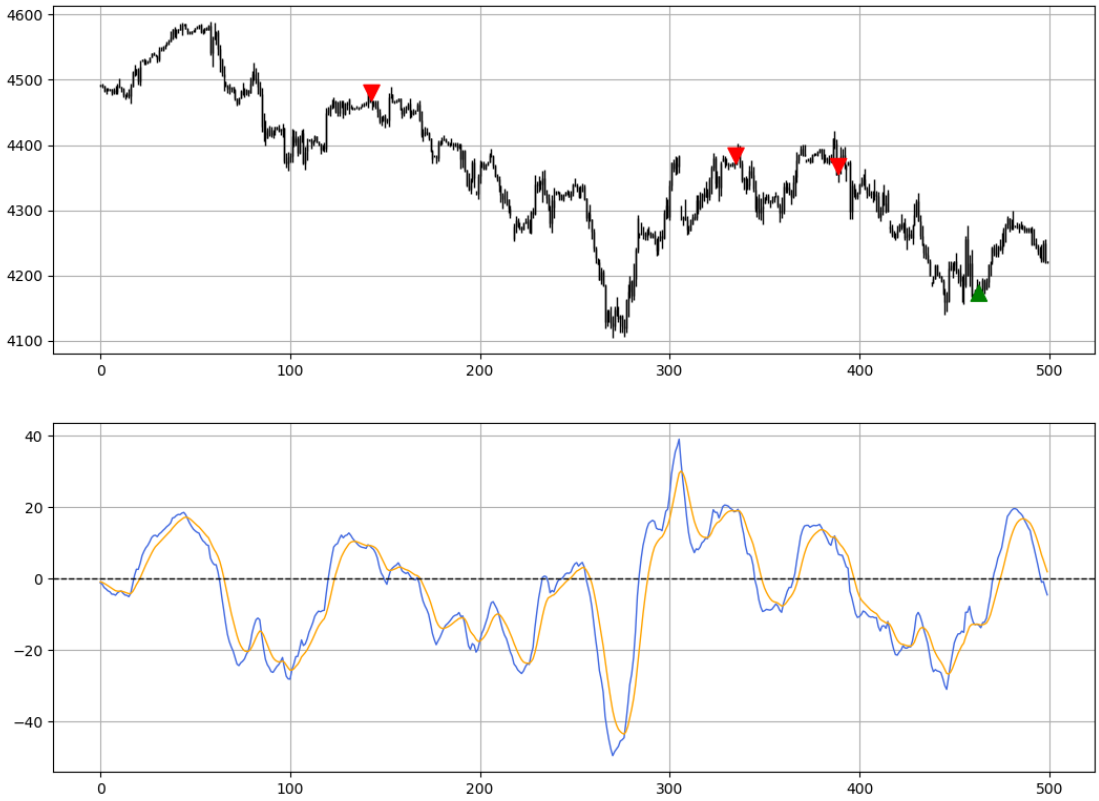


Figure 2 Signal chart on S&P500.

To summarize, the special divergence technique can be an alternative to the normal divergence technique when dealing with unbounded indicators such as the MACD. Furthermore, it can also be used with bounded indicators to increase the conviction by combining the technique with lower and upper barriers. For example, you can use the special divergence technique on the RSI while adding a condition that defines the obligation that the RSI should either be above the upper barrier or below the lower barrier. Make sure you always combine any divergence strategy with other tools.

STRATEGY #2 THE TIME'S UP INDICATOR

Before I present this strategy, I will introduce the time's up indicator. The main idea is based on counting the number of positive and negative time periods (hours) and deriving trading signals from them.

Hence, the indicator counts the consecutive up and down hours to see whether a pattern is formed on certain subjective barriers. For example, does having five consecutive up close prices consistently provide a reaction to the downside? It is not very likely in such complex markets, but it is worth seeing and back-testing.

As mentioned in the introduction, the time's up indicator will be a count of the consecutive positive and negative hours. For example, if you have three hours where the closing price is greater than the opening price, then the indicator will show a reading of 3. To create the time's up indicator, follow these steps:

- Calculate the price difference between the current market price and the one preceding it. This gives you the change in price.

$$Change\ in\ price_i = Close\ price_i - Close\ price_{i-1}$$

- Create the condition which states that if the change is positive, then the indicator will have a value of the previous one plus 1, otherwise, the value equals zero.
- Create the other condition which states that if the change is negative, then the indicator will have a value of the previous one minus 1, otherwise, the value equals zero.
- Sum the values from the two previous results to get the time's up indicator.

The code required for the indicator can be found in the following snippet:

```
def time_up(data, width, close, position):
    data = add_column(data, 4)
    # Calculating the difference in prices
    for i in range(len(data)):
        data[i, position] = data[i, close] - data[i - width, close]
        # Upward timing
    for i in range(len(data)):
        data[0, position + 1] = 1
        if data[i, position] > 0:
            data[i, position + 1] = data[i - width, position + 1] + 1
        else:
            data[i, position + 1] = 0
        # Downward timing
        for i in range(len(data)):
            data[0, position + 2] = 1
            if data[i, position] < 0:
                data[i, position + 2] = data[i - width, position + 2] + 1
            else:
                data[i, position + 2] = 0
    # Changing signs
    for i in range(len(data)):
        if data[i, position + 2] != 0:
            data[i, position + 2] = -1 * data[i, position + 2]
    # Time's up indicator
    data[:, position + 3] = data[:, position + 1] + data[:, position + 2]
    data = delete_column(data, position, 3)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable width refers to the length of the subtraction period (which I recommend setting it to 1[46]), the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

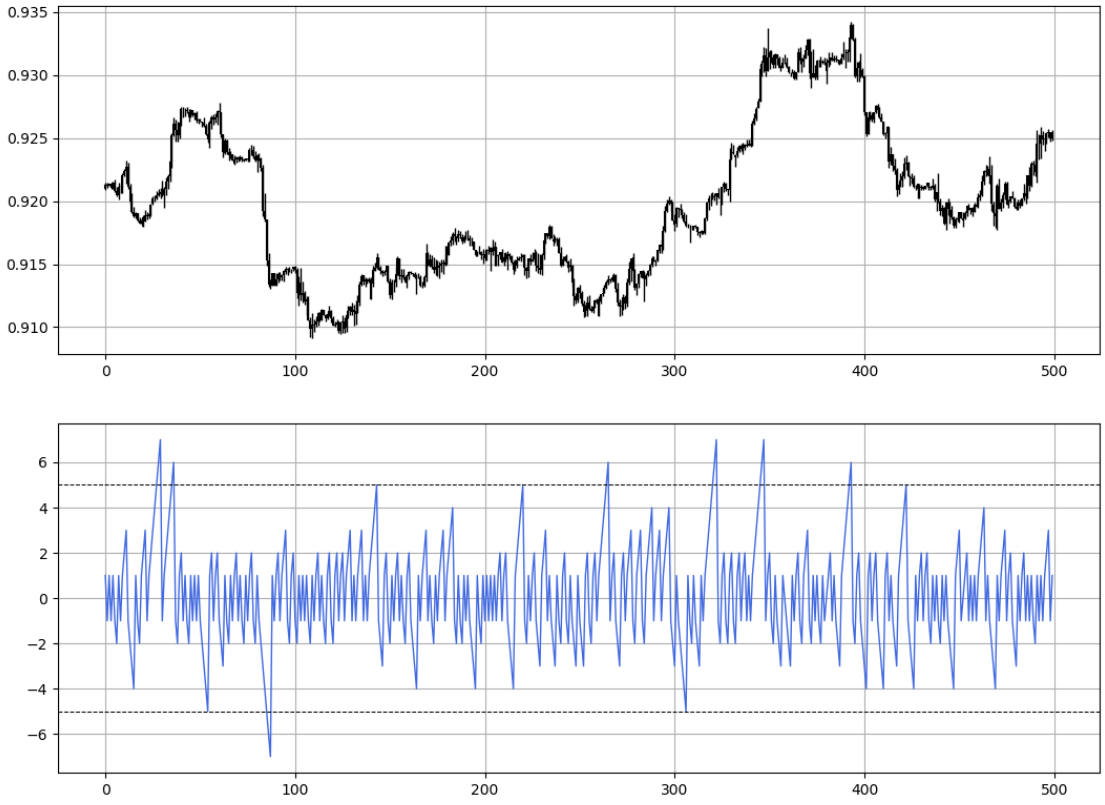


Figure 3 USDCHF with the time's up indicator.

[46] When set to 1, it means that you are subtracting the current close price by the one preceding it.

Let's set the barriers at 5 and -5 from where the following trading conditions will be activated:

- **A long signal is generated whenever the time's up indicator shapes a V form by re-integrating directly above -5 after having been below it.**
- **A short signal is generated whenever the time's up indicator shapes an inverted V form by re-integrating directly below 5 after having been above it.**

```
def signal(data, time_up_column, buy, sell):
    data = add_column(data, 5)
    for i in range(len(data)):
        try:
            # Bullish signal
            if data[i, time_up_column] >= lower_barrier and data[i - 1,
              time_up_column] < lower_barrier and data[i - 2,
              time_up_column] >= lower_barrier:
                data[i + 1, buy] = 1
            # Bearish signal
            elif data[i, time_up_column] <= upper_barrier and data[i - 1,
                time_up_column] > upper_barrier and data[i - 2,
                time_up_column] <= upper_barrier:
                    data[i + 1, sell] = -1
        except IndexError:
            pass
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable time_up_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

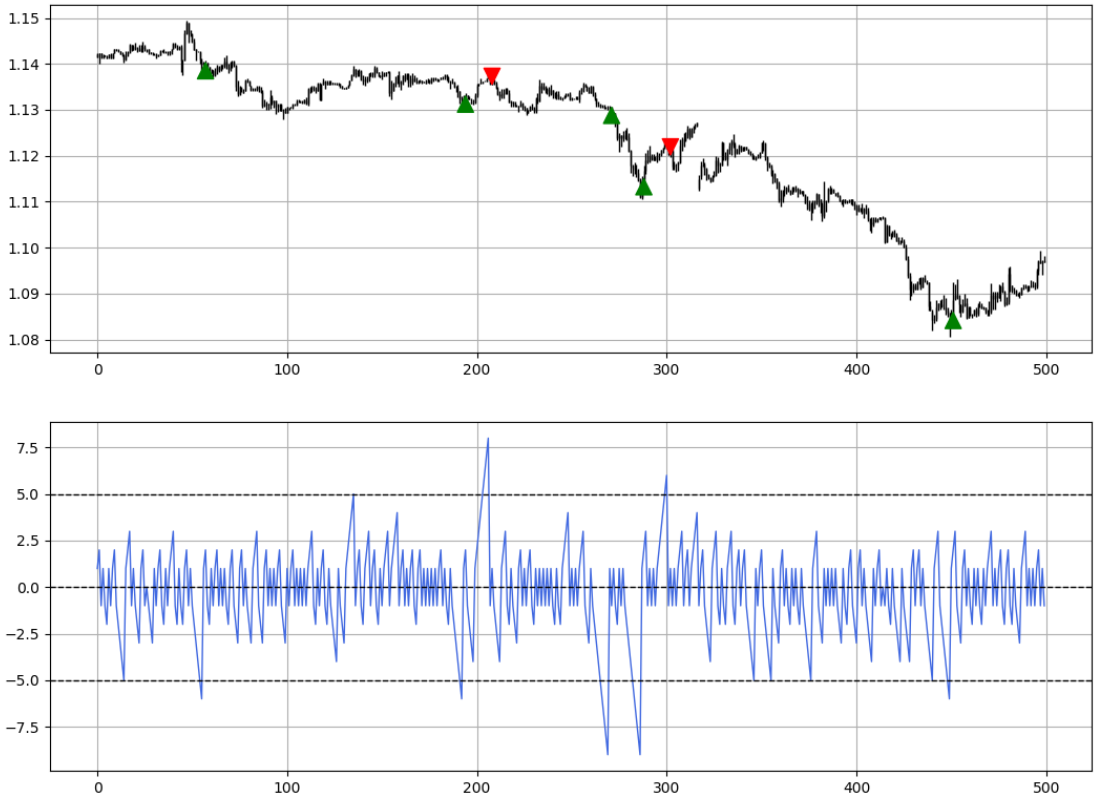


Figure 4 Signal chart on EURUSD.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	48.35%	1.15	1.22	91
USDCHF	42.26%	0.61	0.83	97
GBPUSD	48.91%	1.03	1.07	92
AUDUSD	55.00%	0.87	0.71	80
USDCAD	53.84%	0.93	0.80	91
XAUUSD	36.98%	0.75	1.27	73
XAGUSD	45.20%	0.51	0.62	73
NIKKEI225	56.45%	1.09	0.84	62
S&P500	54.09%	0.85	0.73	61

The next Figure shows an example of signals generated on S&P500.

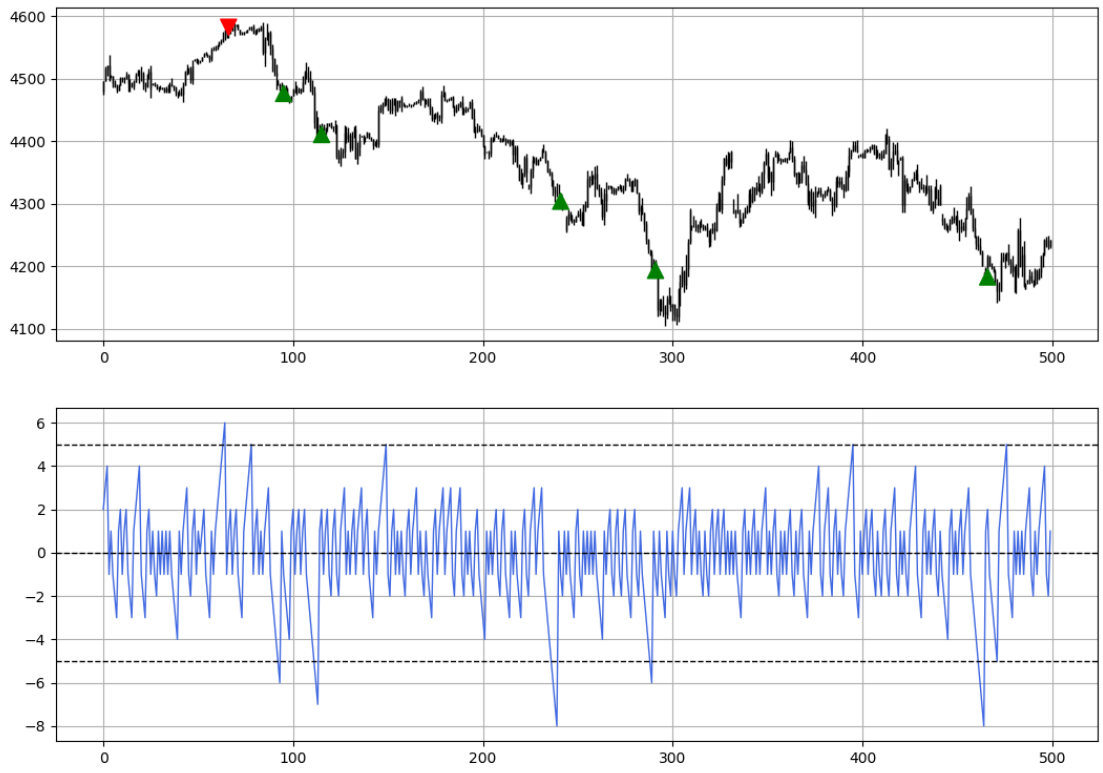


Figure 5 Signal chart on S&P500.

To summarize, the time's up indicator is an extremely simple calculation based on time and price, but this simplicity must be coupled with a more sophisticated technique to generate better signals. In any way, the indicator is useful in analysis and gives a quick glance at the strength of momentum timewise.

It is worth mentioning that the exit technique I am using must not be applied in real life because exiting upon the next signal is more likely to hurt your results than help them. It also lowers drastically the realized risk-reward ratio. I am only using it to show the absolute predictive power of the strategies but not to show how they perform in real life under other exit conditions.

STRATEGY #3 THE FISHER-RSI INDICATOR

You have previously seen the MFT and how to calculate it. Let's now transform it even further and create a trading strategy. The idea is to apply the RSI formula on the values of the MFT. What this does is normalize the MFT between 0 and 100. The strategy relies on the aggressive technique. Therefore, the trading conditions are:

- **A long signal is generated whenever the Fisher-RSI reaches the oversold level.**
- **A short signal is generated whenever the Fisher-RSI reaches the overbought level.**

```
def signal(data, fisher_rsi_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, fisher_rsi_column] < lower_barrier and data[i - 1,
          fisher_rsi_column] > lower_barrier:
            data[i + 1, buy] = 1
        # Bearish signal
        if data[i, fisher_rsi_column] > upper_barrier and data[i - 1,
          fisher_rsi_column] < upper_barrier:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable fisher_rsi_column refers to the column of the indicator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

The next Figure shows an example of signals generated on GBPUSD.

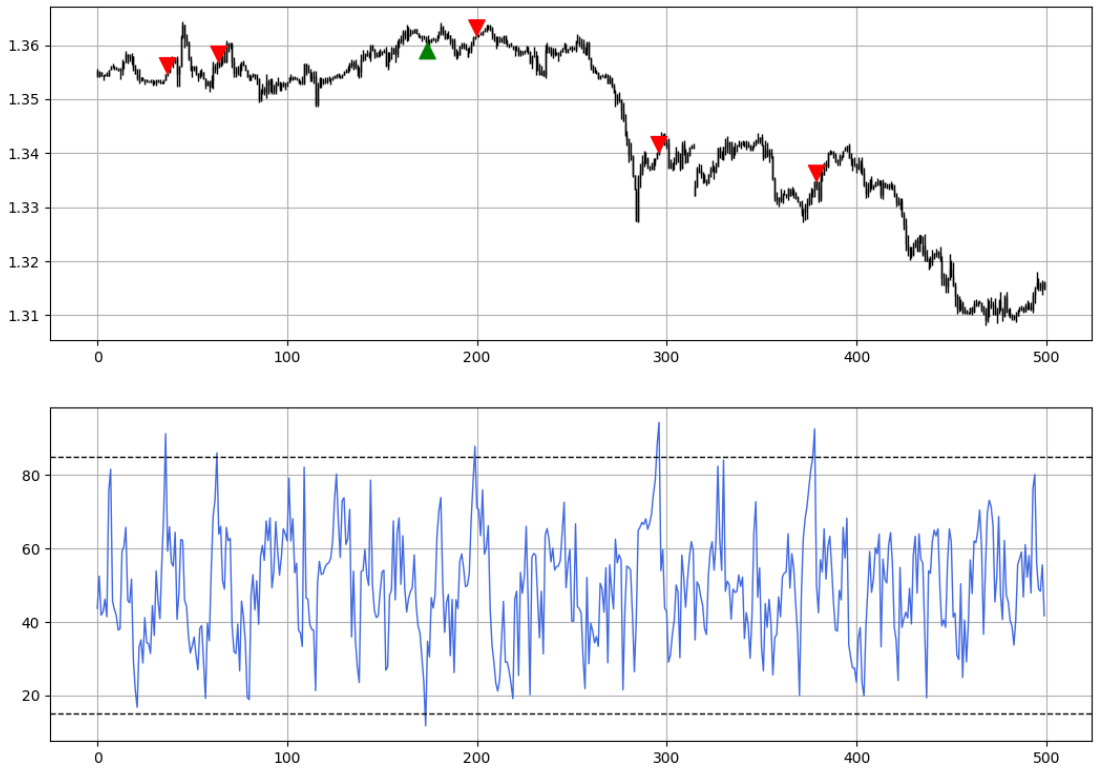


Figure 6 Signal chart on GBPUSD.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	50.17%	0.97	0.96	279
USDCHF	48.83%	0.98	1.03	258
GBPUSD	52.69%	0.98	0.88	260
AUDUSD	52.75%	1.07	0.96	254
USDCAD	53.87%	1.03	0.88	271
XAUUSD	51.02%	0.91	0.88	245
XAGUSD	52.40%	1.12	1.02	229
NIKKEI225	47.45%	0.84	0.93	177
S&P500	45.94%	0.97	1.14	148

The next Figure shows an example of signals generated on S&P500.

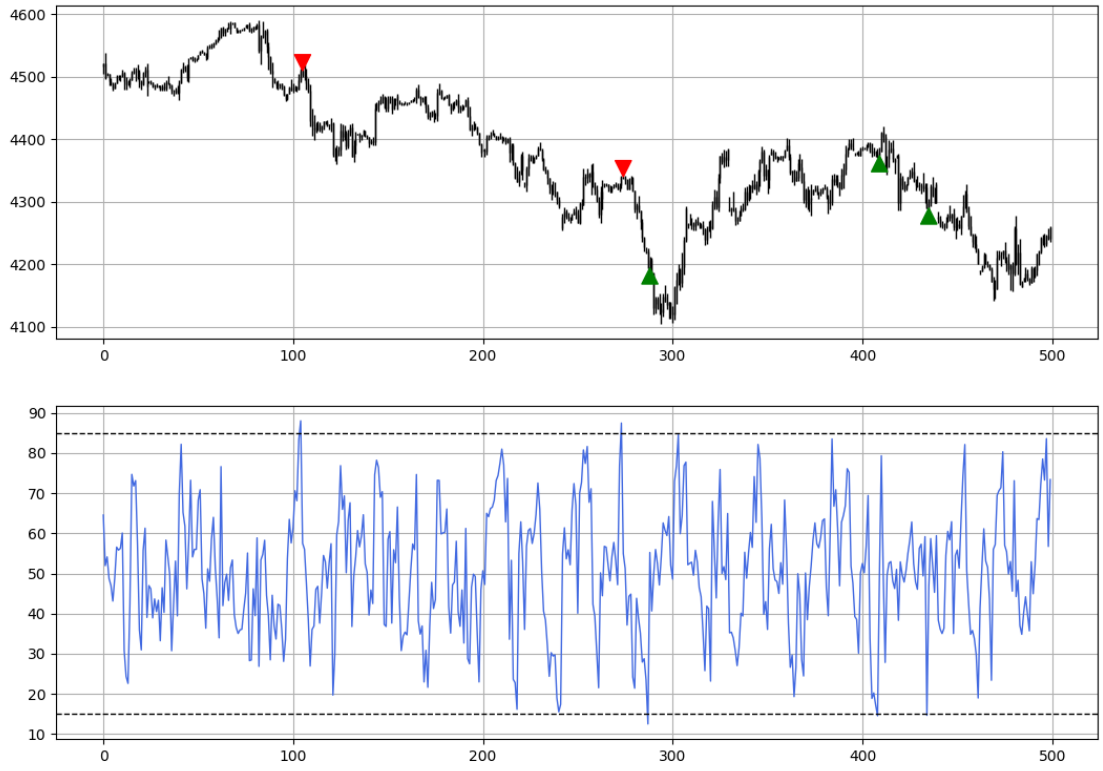


Figure 7 Signal chart on S&P500.

To summarize, the Fisher-RSI is a transformation that combines the MFT with the RSI in a bounded indicator. The strategy should not be used alone as the back-testing results show that it has not immense predictive power. However, it can be tweaked and optimized.

STRATEGY #4 PIVOT POINTS

A support level is an area supposed to keep prices from going down further. Naturally, when the market price approaches a support area, the right decision is to have a bullish bias since the support should induce a bounce. A resistance level is an area supposed to keep prices from going up further. Naturally, when the market price approaches a resistance area, the right decision is to have a bearish bias since the resistance should induce a consolidation.

Pivot points are simple mathematical calculations used to find implied[47] support and resistance levels. They are very simple and easy to use. Some intraday traders calculate them every day and project levels that last for the duration of the trading day. Other traders calculate the levels on a monthly and on a yearly basis. For example, on the first day of January, a trader might calculate the projected support and resistance levels using last year's values. The framework of this technique is as follows:

- **Defining the needs**: By calculating the pivot points, you are searching for support and resistance levels which will give you a certain bias if the market approaches them.
- **Determining the relevant levels**: By finding the levels needed for the formulas that I will later present, you will be able to calculate the implied support and resistance levels. You will need yesterday's high, low, and close prices if you are using hourly charts to trade on the intraday horizon. If you are an investor, you will need last year's high, low, and close prices so that you have an idea on this year's implied support and resistance levels.

[47] I say implied because they are merely expected levels from where a reaction is to be expected. However, they are not foolproof.

- **Acting on the levels**: By drawing the levels on your chart and setting alerts, you will be able to make a decision whenever the market price approaches them. I believe you know that from the tone of this book, you will be buying on support and selling on resistance.

I will present the details on pivot points from an intraday perspective which means that the formulas are calculated using daily values but then projected on intraday hourly charts.

The following formula is used to calculate the pivot point which is the basic ingredient to calculate the implied support and resistance levels:

$$Pivot\ point_i = \frac{High\ price_{i-1} + Low\ price_{i-1} + Close\ price_{i-1}}{3}$$

Therefore, today's pivot point is calculated as the average between yesterday's high, low, and close prices. Let's see how to calculate today's implied support and resistance levels (note that you must do this in the morning of any time zone):

$$First\ support = (Pivot\ point_i\ x\ 2) - High\ price_{i-1}$$

$$Second\ support = Pivot\ point_i - (High\ price_{i-1} - Low\ price_{i-1})$$

$$Third\ support = Low\ price_{i-1} - 2\ x\ (High\ price_{i-1} - Pivot\ point_i)$$

$$First\ resistance = (Pivot\ point_i\ x\ 2) - Low\ price_{i-1}$$

$$Second\ resistance = Pivot\ point_i + (High\ price_{i-1} - Low\ price_{i-1})$$

$$Third\ resistance = High\ price_{i-1} - 2\ x\ (Pivot\ point_i - Low\ price_{i-1})$$

Note that the most important levels are the first support/resistance levels. What is the utility of the pivot point then? Basically, it can be used as a magnet where the market should revert back to it in case it deviates away. Another

method is to use it as the profit level where after initiating a reversal position, you can target the pivot point as an exit. Basically, when the market closed the previous day, consider the following:

- High = 1.20950
- Low = 1.20275
- Close = 1.20650

Following the above formulas, the pivot point should be 1.2062 and the first support should be 1.2030. The first resistance is 1.2100.

The trading conditions for a pivot point strategy are as follows:

- **A long signal is generated whenever the market price reaches the first pivot support level.**
- **A short signal is generated whenever the market price reaches the first pivot resistance level.**

Of course, with such a simple and known strategy, it is unlikely to easily have a consistently profitable strategy. It is possible that at any moment the support or resistance level fails to hold, this is where you should say that the move is confirmed in the other direction.

When trading the pivot points, make sure you combine them with other indicators and patterns to maximize your chances of success. Also, it may be better not to go against the trend. You may use a long-term moving average for this and check only for buy opportunities if the market is above the moving average. In parallel, you should only initiate short opportunities whenever the market is below its moving average. Therefore, a more sophisticated pivot points trading strategy may have the following conditions:

- **A long signal is generated whenever the market price reaches the first pivot support level while simultaneously being above the 100-period moving average.**
- **A short signal is generated whenever the market price reaches the first pivot resistance level while simultaneously being below the 100-period moving average.**

To summarize, pivot points are a great price action tool that complements your already existing analysis method. Strategies based on pivot points should not be used alone nor in the opposite direction of the ongoing trend.

STRATEGY #5 THE VAMA BANDS & THE STOCHASTIC OSCILLATOR

The volatility-adjusted moving average (VAMA) is an overlay technical indicator created by Tushar S. Chande. The steps needed to create the VAMA are as follows:

- Measure the alpha factor which can be found through this formula below:

$$\alpha_i = \frac{\sigma_{Short}}{\sigma_{Long}} \; x \; 0.20$$

- To calculate the VAMA, follow these next steps:

$$VAMA_i = \; \alpha_i \; . Close \; price_i + (1 - \alpha_i) \; . VAMA_{i-1}$$

The first VAMA value is simply the closing price, and then the algorithm can take the form of the above formula.

The VAMA adjusts to volatility in a way that reflects the current market conditions making it one of the most up-to-date moving averages. Compared to a simple moving average, the VAMA has the advantage of including much more information in its calculation.

The VAMA bands are simply Bollinger bands that use the VAMA instead of a simple moving average in the formula. The first part of the strategy is the VAMA bands. The second part is the stochastic oscillator seen in chapter 2.

```
def vama_bands(data, lookback_volatility_short, lookback_volatility_long, lookback_volatility, std, close,
               position):
    data = volatility_adjusted_moving_average(data, lookback_volatility_short, lookback_volatility_long,
           close, position)
    data = volatility(data, lookback_volatility, close, position + 1)
    data = add_column(data, 2)
    data[:, position + 2] = data[:, position] + (std * data[:, position + 1])
    data[:, position + 3] = data[:, position] - (std * data[:, position + 1])
    data = delete_row(data, lookback_volatility)
    data = delete_column(data, position + 1, 1)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation including the current one, the variable lookback_volatility_short is the short volatility lookback, the variable lookback_volatility_long, the variable lookback_volatility is the volatility lookback of the close, the variable std refers to the volatility multiplier, the variable close refers to the column of the close price, and the variable position refers to the column where you place the output. The trading conditions of the strategy are as follows:

- **A long signal is generated whenever the market price touches the lower previous[48] VAMA band while the 20-period stochastic oscillator is below 10.**
- **A short signal is generated whenever the market price touches the upper previous VAMA band while the 20-period stochastic oscillator is above 90.**

[48] As the position of the current band is unknown when dealing with hourly bars.

```
def signal(data, high, low, upper_vama_band, lower_vama_band, stoch_column, buy,
sell):
    data = add_column(data, 5)
    for i in range(len(data)):
        try:
        # Bullish signal
        if data[i, low] < data[i, lower_vama_band] and data[i - 1, low] > data[i –
          1, lower_vama_band] and data[i, stoch_column] < lower_barrier:
            data[i, buy] = 1
        # Bearish signal
        elif data[i, high] > data[i, upper_vama_band] and data[i - 1, high] <
          data[i - 1, upper_vama_band] and data[i, stoch_column] >
          upper_barrier:
            data[i, sell] = -1
        except IndexError:
            pass
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable upper_vama_band refers to the column of the upper band, the variable lower_vama_band refers to the column of the lower band, the variable stoch_column refers to the column of the stochastic oscillator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Important

Notice the absence of i + 1 in the signal conditions. This is intentional and it is to show you how to generate signals immediately when touching the previous bands as opposed to awaiting a close above the upper band or below the lower band.

The next Figure shows an example of signals generated on USDCHF.

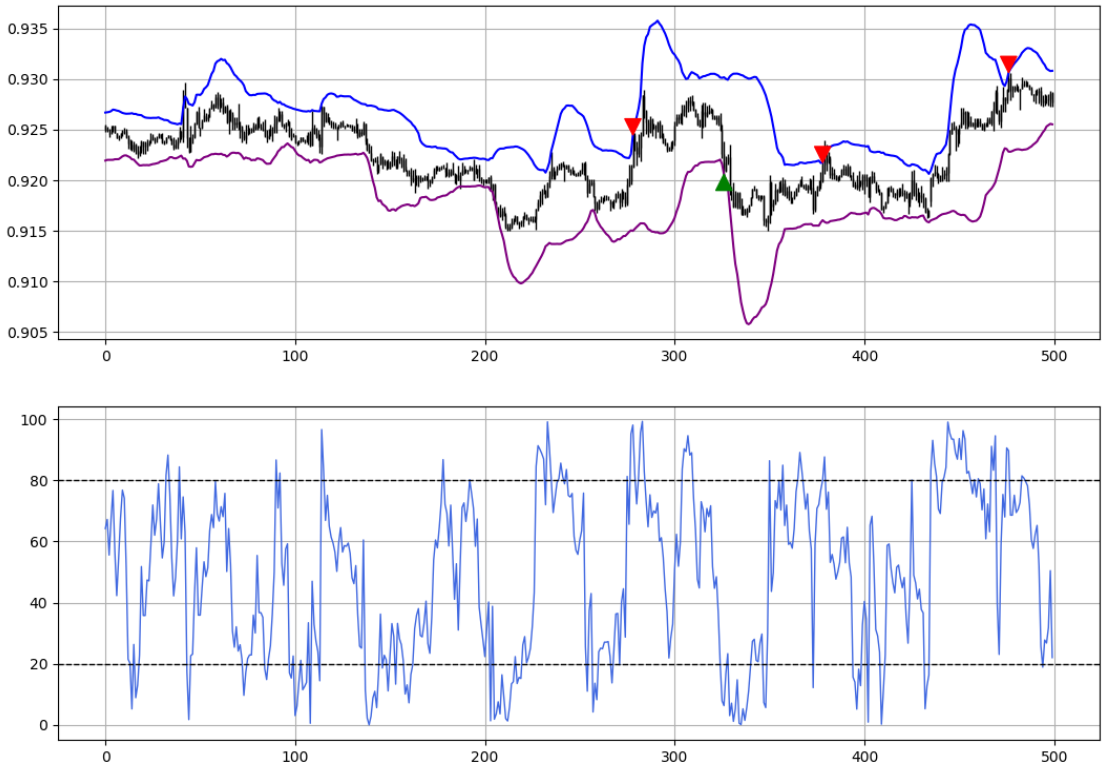


Figure 8 Signal chart on USDCHF.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	50.36%	0.77	0.76	407
USDCHF	48.45%	0.76	0.81	355
GBPUSD	48.88%	0.83	0.87	360
AUDUSD	50.45%	1.02	1.00	331
USDCAD	51.78%	0.76	0.70	336
XAUUSD	44.32%	0.74	0.93	388
XAGUSD	49.12%	0.77	0.80	342
NIKKEI225	57.69%	1.21	0.89	182
S&P500	52.42%	1.02	0.93	227

The next Figure shows an example of signals generated on AUDUSD.

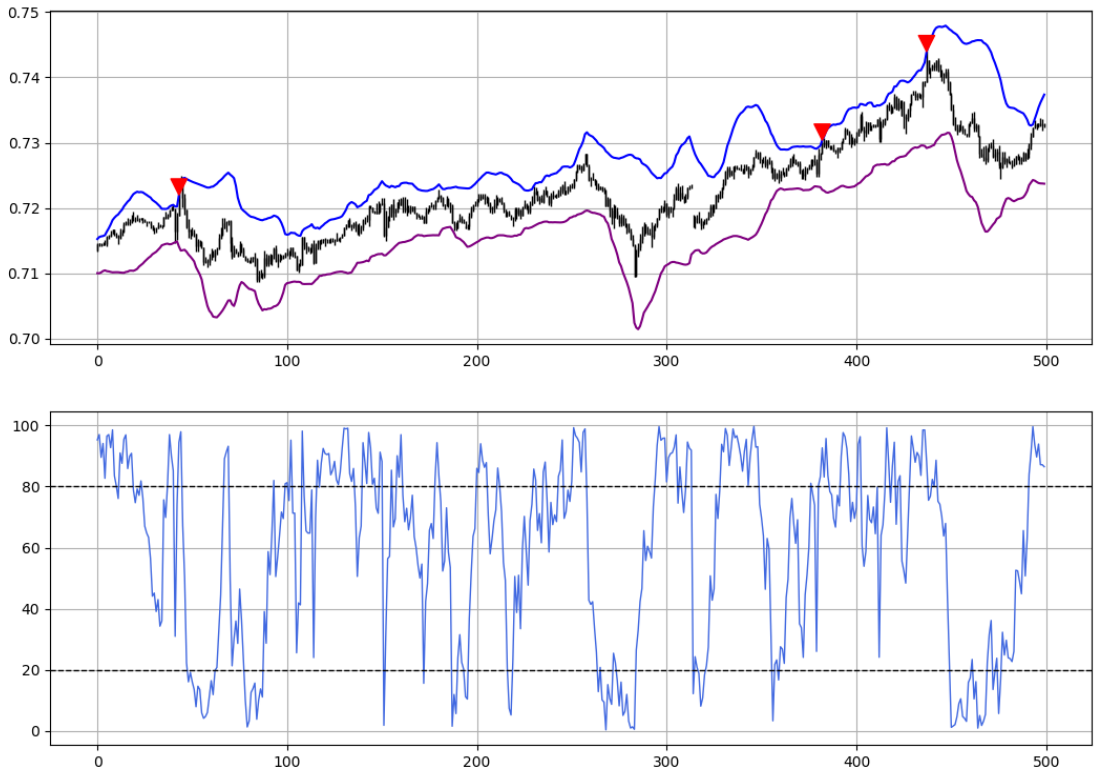


Figure 9 Signal chart on AUDUSD.

To summarize, the VAMA bands are like the Bollinger bands but use the volatility-adjusted moving average, a powerful type of overlay indicators that takes into account volatility. The strategy relies on confirmation from the stochastic oscillator but needs to be optimized for the number of signals and profitability.

Reminder

Remember that the strategies presented are not investment advice as evidenced by the back-testing results. The key takeaway is that tweaking them will add value and give rise to better results. The main aim of this chapter is to see how to combine different indicators and techniques in a harmonized way.

STRATEGY #6 PSYCHOLOGICAL LEVELS

A psychological level is generally a round number or a par-number such as 1.0000 on USDCHF or $100.00 on Apple's stock price. The question is, if you approach psychological levels, is it worth initiating trades?

Psychological levels form an important part in any analysis. The reason for this is because mentally, they are given more attention than other levels. For instance, which price would you retain more in your mind if you came across it? 1.1500 on EURUSD or 1.3279 on GBPUSD?

For example, par-values or par-levels such as 1.000 on USDCHF or 100.00 on USDJPY are considered psychological levels. The basic idea is that around these levels, the market participants may choose to place their orders, hence, a form of reaction may happen. By creating an algorithm that scans these levels and combining it with a simple technical indicator such as the stochastic oscillator, you can derive the following trading rules:

- **A long signal is generated whenever a psychological level is reached while the stochastic oscillator is below 15.**
- **A short signal is generated whenever a psychological level is reached while the stochastic oscillator is above 85.**

I have added the stochastic oscillator to deliver the directional message to the algorithm. Basically, it will scan for round numbers like 1.1400 EURUSD or 0.9100 USDCHF and then mark these points. After that, the stochastic will be able to tell whether the market is oversold or overbought whenever it hits the psychological level.

The following function scans for psychological levels in markets such as EURUSD, GBPUSD, and USDCAD. For markets like S&P500 and Gold, I will define another function.

```
def psychological_levels_scanner(data, close, position):
    data = add_column(data, 10)
    # Rounding for ease of use
    data = rounding(data, 4)
    # Threshold
    level = 0
    # Scanning for Psychological Levels
    for i in range(len(data)):
        for i in range(len(data)):
            if data[i, close] == level:
                data[i, position] = 1
        level = round(level + 0.01, 2)
        if level > 5:
            break
    return data
```

The function defines the scanner where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

The next figure shows how rare it is to find psychological levels coupled with a signal from an indicator. Notice that the results will not be indicative as the number of signals may not be sufficient for the study to be meaningful especially with a starting test period of 2020.

Figure 10 Signal chart on USDCAD.

The trading conditions for this strategy are as follows:

- **A long signal is generated whenever the scanner detects a psychological level while the stochastic oscillator is at or below 25.**
- **A short signal is generated whenever the scanner detects a psychological level while the stochastic oscillator is at or above 75.**

```
def signal(data, psychological_column, stoch_column, buy, sell):
        data = add_column(data, 5)
        for i in range(len(data)):
                try:
                # Bullish signal
                if data[i, psychological_column] == 1 and data[i, stoch_column] <=
                  lower_barrier:
                        data[i + 1, buy] = 1
                # Bearish signal
                elif data[i, psychological_column] == 1 and data[i, stoch_column] >=
                    upper_barrier:
                data[i + 1, sell] = -1
                except IndexError:
                        pass
        return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical data, the variable psychological_column refers to the column containing the psychological neutral[49] signals, the variable stoch_column refers to the column of the stochastic oscillator. The buy and sell variables refer to the columns where bullish and bearish signals are outputted.

[49] The word neutral refers to a plain signal that has detected a round number, but it says nothing about the position of the market relative to the psychological level. This is why you need an indicator to help you understand whether the level is a support or a resistance.

The next Figure shows an example of signals generated on EURUSD.

Figure 11 Signal chart on EURUSD.

Markets like Nikkei225 are quoted in thousands and therefore, the scanner function must be modified to account for this. You will now tell it to start from 1,000 until 30,000 with rounds of 100 in-between. This should cover the psychological levels in Gold, Nikkei225, and S&P500 around these recent years.

The next code snippet shows how to do this.

```
def psychological_levels_scanner(data, close, position):
        data = add_column(data, 10)
        # Rounding for ease of use
        data = rounding(data, 0)
        # Threshold
        level = 1000
        # Scanning for psychological levels
        for i in range(len(data)):
                for i in range(len(data)):
                        if data[i, close] == level:
                                data[i, position] = 1
                level = round(level + 100, 2)
                if level > 30000:
                        break
        return data
```

Important

Do you think that you can apply the above function on Silver? If you have answered no, then you are right. I will let you do the necessary tweaks in the function to find the right answer.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	48.57%	1.03	1.09	35
USDCHF	47.22%	1.35	1.51	36
GBPUSD	48.27%	1.50	1.60	58
AUDUSD	60.00%	2.42	1.62	40
USDCAD	40.00%	0.86	1.30	40
XAUUSD	43.75%	0.72	0.92	16
XAGUSD[50]	46.55%	0.83	0.95	116
NIKKEI225	61.36%	1.13	0.71	44
S&P500	53.33%	0.75	0.65	30

To summarize, psychological levels are levels that impact traders more than other levels. By combining them with the stochastic oscillator, the algorithm will understand whether the signal should be bullish or bearish. I recommend you always keep an eye on near psychological levels as the probability of a reaction on that level is greater than average.

It is crucial to design exit rules that optimize the results of your strategies. This is something that I will present in detail in a subsequent book.

[50] After applying the tweak that you are supposed to find on your own.

STRATEGY #7 THE RSI² DIVERGENCE

The RSI is a good contrarian indicator. However, human creativity allows us to push the limits even more. The RSI² is an RSI calculated on another RSI. This means that to calculate this indicator, you should follow these steps:

- Calculate a 14-period RSI on the close prices.
- Calculate a 5-period RSI on the previous RSI's values. This is called the RSI².

The trading conditions are as follows:

- **A long signal is generated whenever a bullish divergence is spotted on the RSI².**
- **A short signal is generated whenever a bearish divergence is spotted on the RSI².**

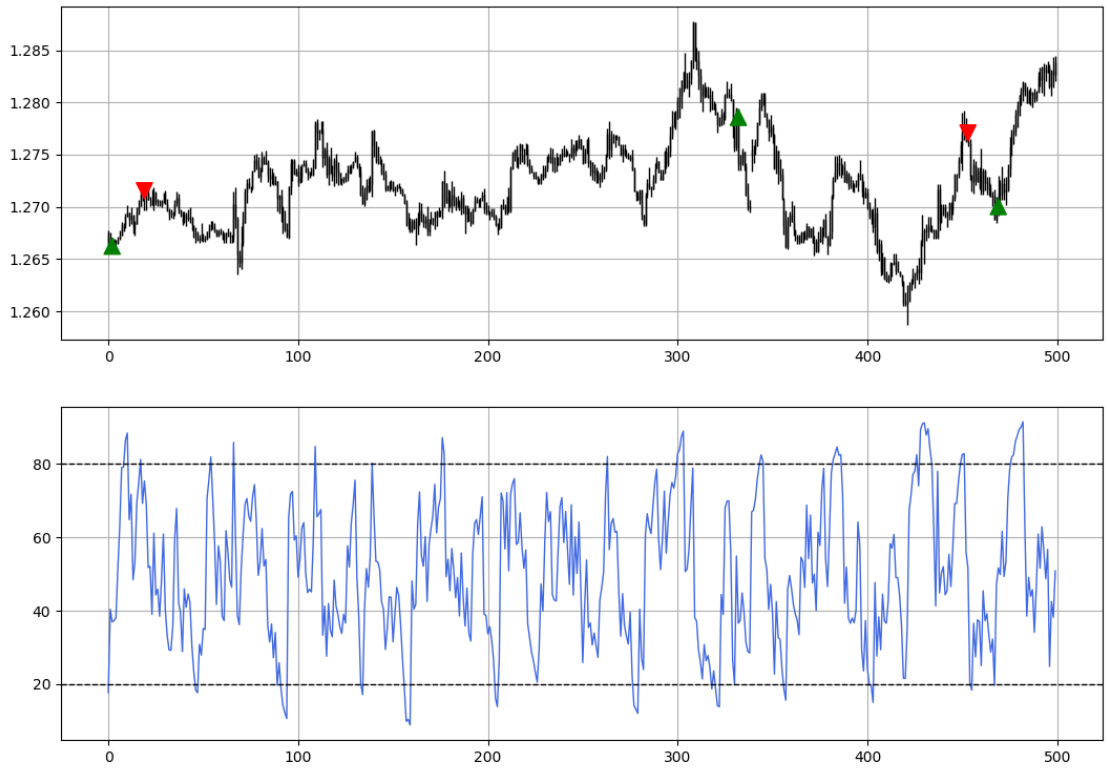


Figure 12 Signal chart on USDCAD.

The signal function of the divergence technique is lengthy which is why I have decided it to only put it in the GitHub repository. You can access it through the link[51] or through the below QR code.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	44.66%	0.83	1.03	150
USDCHF	53.47%	0.91	0.79	144
GBPUSD	51.38%	0.87	0.83	144
AUDUSD	44.09%	0.75	0.95	161
USDCAD	48.76%	0.69	0.72	162
XAUUSD	45.32%	0.79	0.95	139
XAGUSD	44.14%	0.69	0.87	111
NIKKEI225	46.31%	0.82	0.96	95
S&P500	49.01%	1.17	1.22	102

[51] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

The next Figure shows an example of signals generated on BTCUSD.

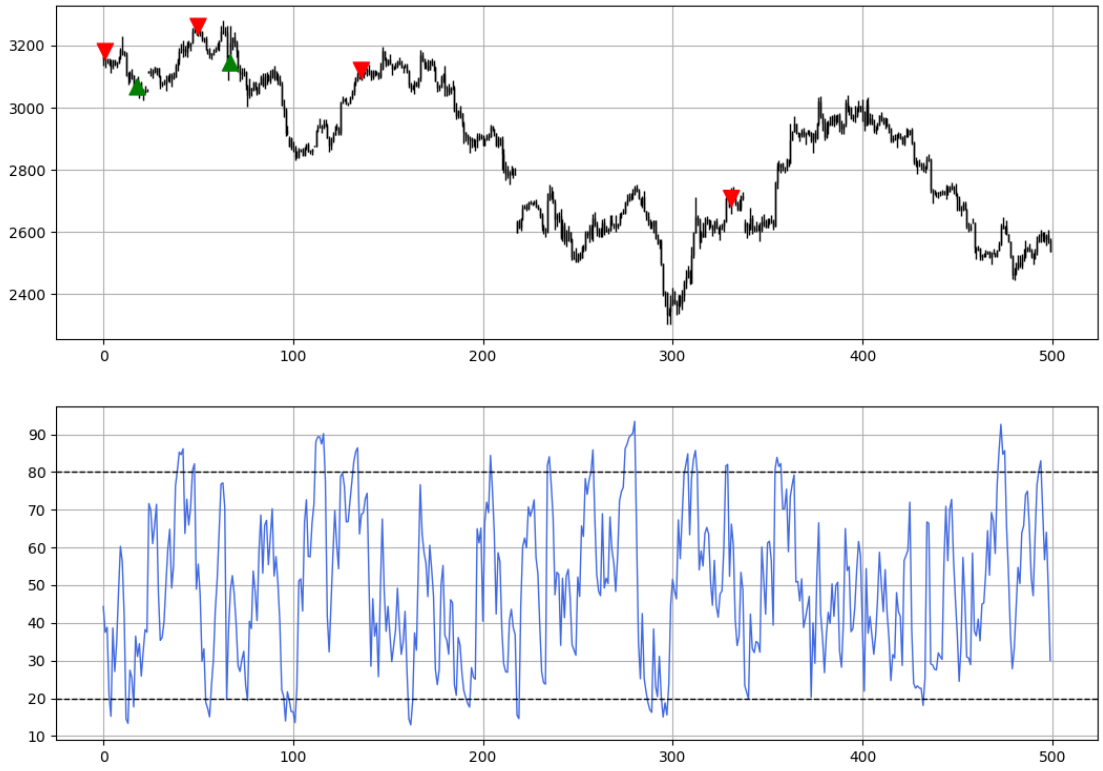


Figure 13 Signal chart on ETHUSD.

To summarize, the divergence on the RSI² might signal a reversal in the RSI which in turn might signal a reversal in the market price. This is a strategy based on immense lag due to the RSI² but with the divergence being a leading[52] indicator, it offsets some of that issue.

[52] A leading indicator shows signals before the validation while a lagging indicator shows either an instant or a late signal.

STRATEGY #8 THE KELTNER CHANNEL

The Keltner channel is an envelopment indicator created by Chester Keltner that uses a concept similar to Bollinger bands. Below are the main differences between the two:

- Bollinger bands use the simple moving average while the Keltner channel uses the exponential moving average.
- Bollinger bands use standard deviation to calculate the bands while the Keltner channel uses the ATR to calculate the bands.

To create the Keltner channel, use the following code:

```
def keltner_channel(data, lookback, multiplier, close, position):
    data = add_column(data, 2)
    data = ema(data, 2, lookback, close, position)
    data = atr(data, lookback, 2, 1, 3, position + 1)
    data[:, position + 2] = data[:, position] + (data[:, position + 1] * multiplier)
    data[:, position + 3] = data[:, position] - (data[:, position + 1] * multiplier)
    data = delete_column(data, position, 2)
    data = delete_row(data, lookback)
    return data
```

The function defines the indicator where the variable data refers to the OHLC array containing the historical values in four columns, the variable lookback refers to the number of past time periods to consider in the calculation of the exponential moving average[53] as well as the ATR, the variable multiplier is the volatility multiplier, the variable close refers to the column of the close price, and the variable position refers to the column where you place the indicator.

[53] The code of the exponential moving average can be found in the Master_Functions.py file in the GitHub repository.

The trading conditions are as follows:

- **A long signal is generated whenever the market price touches the lower previous Keltner band.**
- **A short signal is generated whenever the market price touches the upper previous Keltner band.**

```
def signal(data, high, low, upper_kc, lower_kc, buy, sell):
    data = add_column(data, 5)
    for i in range(len(data)):
        try:
            # Bullish signal
            if data[i, low] < data[i, lower_kc] and data[i - 1, low] > data[i - 1,
              lower_kc]:
                data[i, buy] = 1
            # Bearish signal
            elif data[i, high] > data[i, upper_kc] and data[i - 1, high] < data[i -
                1, upper_kc]:
                data[i, sell] = -1
        except IndexError:
            pass
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable upper_kc refers to the column of the upper band, the variable lower_kc refers to the column of the lower band, and the variables buy and sell refer to the columns containing bullish and bearish signals.

The next Figure shows an example of signals generated on USDCHF.

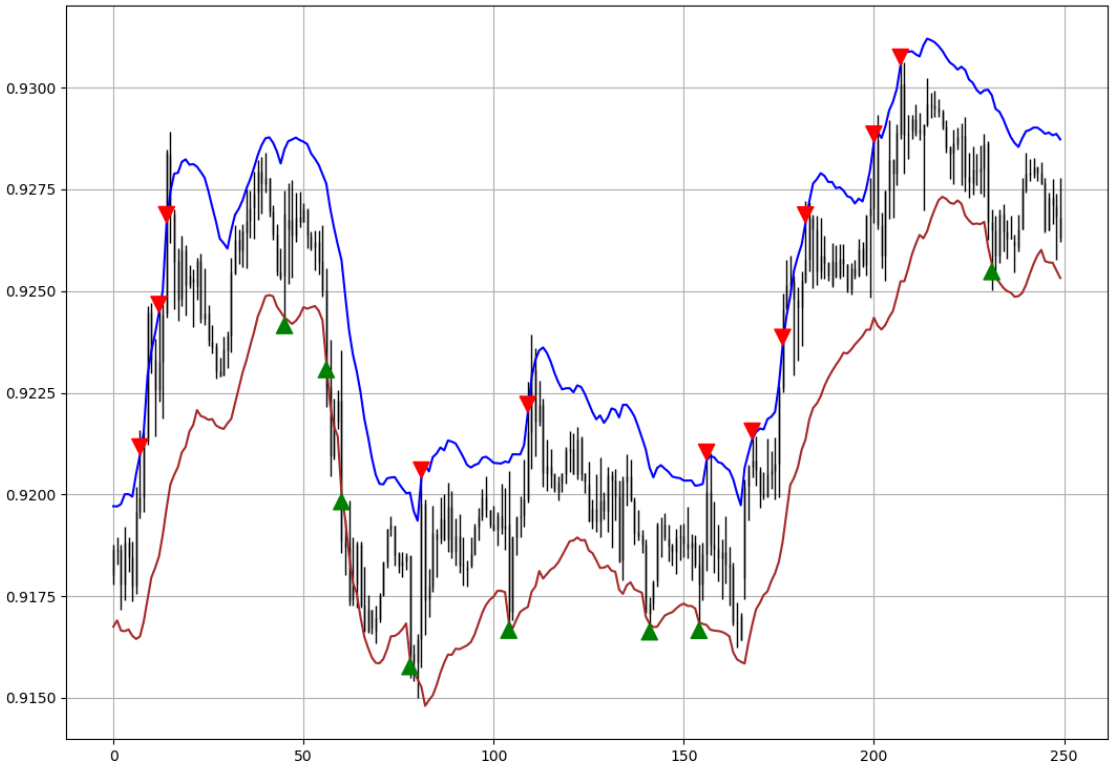


Figure 14 Signal chart on USDCHF.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	48.87%	0.96	1.00	1866
USDCHF	50.60%	0.97	0.94	1834
GBPUSD	50.10%	0.99	0.99	1886
AUDUSD	49.37%	1.04	1.06	1930
USDCAD	51.69%	0.91	0.85	1863
XAUUSD	46.37%	0.98	1.14	1695
XAGUSD	47.73%	0.96	1.05	1701
NIKKEI225	49.51%	1.10	1.12	1143
S&P500	44.86%	1.03	1.27	1130

The next Figure shows an example of signals generated on GBPUSD. Notice how the channel provides bad signals in a trending market.

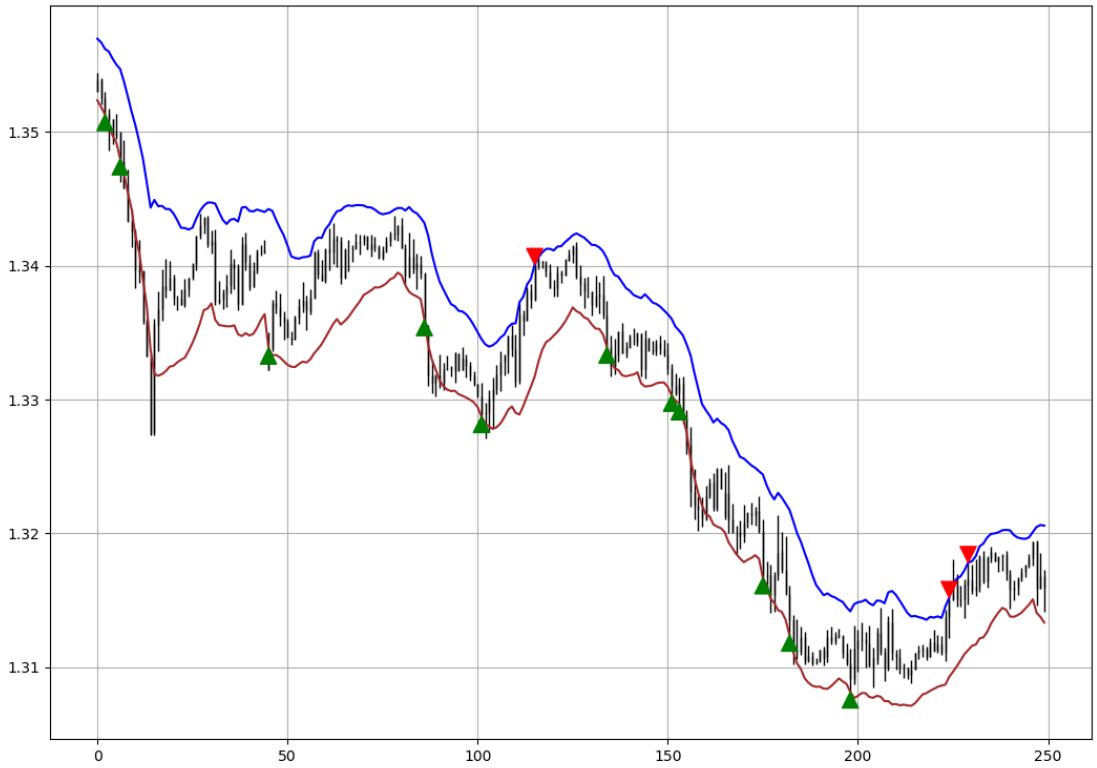


Figure 15 Signal chart on GBPUSD.

To summarize, the Keltner channel is an indicator similar to the Bollinger bands and it needs to be tweaked for every market so as to find the right balance. I recommend setting the multiplier to at least 2. Note that the back-testing results of this strategy are for the default version of the Keltner channel that uses a 10-period exponential moving average and a multiplier of 2.

STRATEGY #9 THE BOUNDED SLOPE INDICATOR

This strategy uses the concept of slopes and normalizes the values so that they are bounded between 0 and 100. Before I present the strategy, I will show you how to calculate the bounded slope indicator (BSI). The steps required to calculate the indicator are as follows:

- Calculate the 14-period slope of the price. The slope can be calculated using the following formula:

$$Slope_i = \frac{Close\ price_i - Close\ price_{i-lookback}}{Lookback}$$

- Calculate the RSI of the slope calculation found from the previous step to get the BSI.

The trading conditions for this strategy are as follows:

- **A long signal is generated whenever the 14-period BSI reaches the oversold level at 30.**
- **A short signal is generated whenever the 14-period BSI reaches the overbought level at 70.**

```
def signal(data, bsi_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        try:
            # Bullish signal
            if data[i, bsi_column] > lower_barrier and data[i - 1, bsi_column]
              < lower_barrier:
                data[i + 1, buy] = 1
            # Bearish signal
            if data[i, bsi_column] < upper_barrier and data[i - 1, bsi_column]
              > upper_barrier:
                data[i + 1, sell] = -1
        except IndexError:
            pass
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable bsi_column refers to the column of the BSI, and the variables buy and sell refer to the columns containing bullish and bearish signals.

Figure 16 shows the signals generated on GBPUSD.

Figure 16 Signal chart on GBPUSD.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	49.04%	1.08	1.13	210
USDCHF	52.79%	0.97	0.87	197
GBPUSD	51.30%	0.82	0.78	191
AUDUSD	46.50%	0.83	0.95	200
USDCAD	44.21%	0.84	1.07	190
XAUUSD	44.50%	0.88	1.10	200
XAGUSD	39.32%	0.51	0.79	178
NIKKEI225	51.09%	1.14	1.09	137
S&P500	48.43%	0.86	0.92	128

The next Figure shows an example of signals generated on USDCAD.

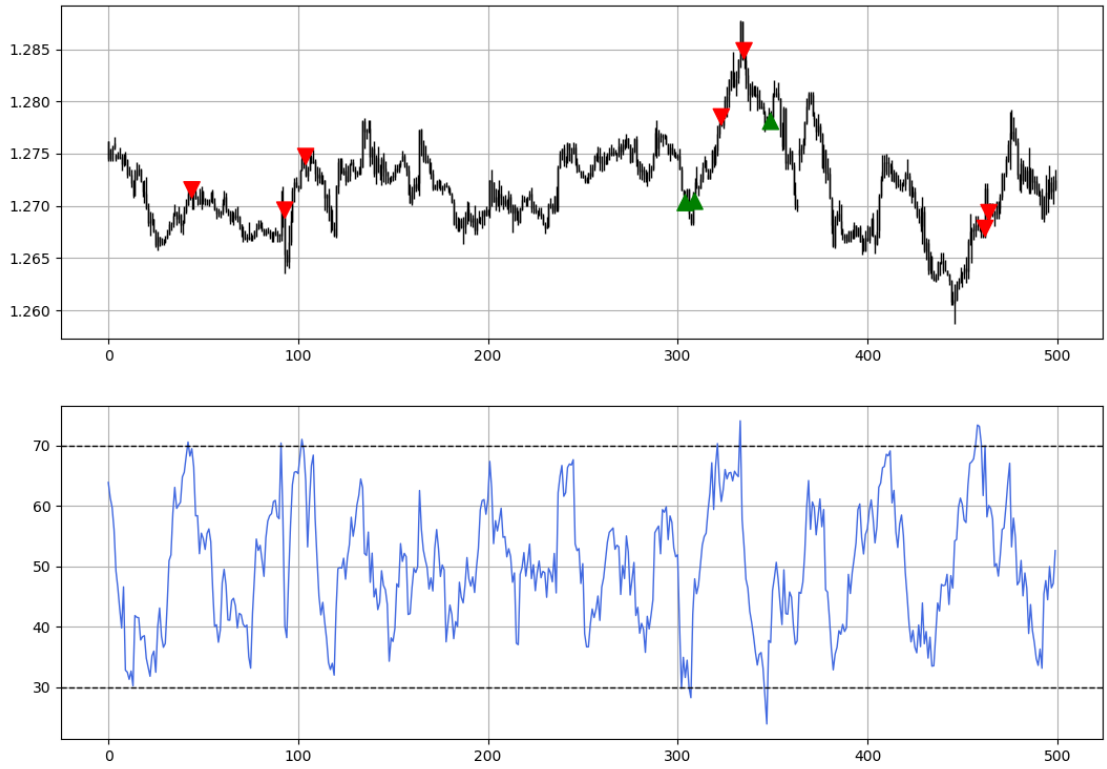


Figure 17 Signal chart on USDCAD.

To summarize, the BSI is a simple transformation that combines the slope concept and the normalization concept (manifested through the RSI). It is relatively a new indicator that I have presented, and I highly recommend confirming its signals with the RSI. Still, the results of the sub-optimal exit technique that I have mentioned show that on absolute, it does not add any value due to the trending regimes that occur once in a while.

STRATEGY #10 THE ROB BOOKER'S REVERSAL INDICATOR

Rob Booker's reversal indicator uses a special combination between the MACD oscillator and the stochastic oscillator. The indicator has the following rules:

- **A long signal is generated whenever the MACD oscillator surpasses the zero line while the 10-period smoothing[54] of the 70-period stochastic oscillator is below the lower barrier 30.**
- **A short signal is generated whenever the MACD oscillator breaks the zero line while the 10-period smoothing of the 70-period stochastic oscillator is above the upper barrier 70.**

The next Figure shows an example of signals generated on S&P500.

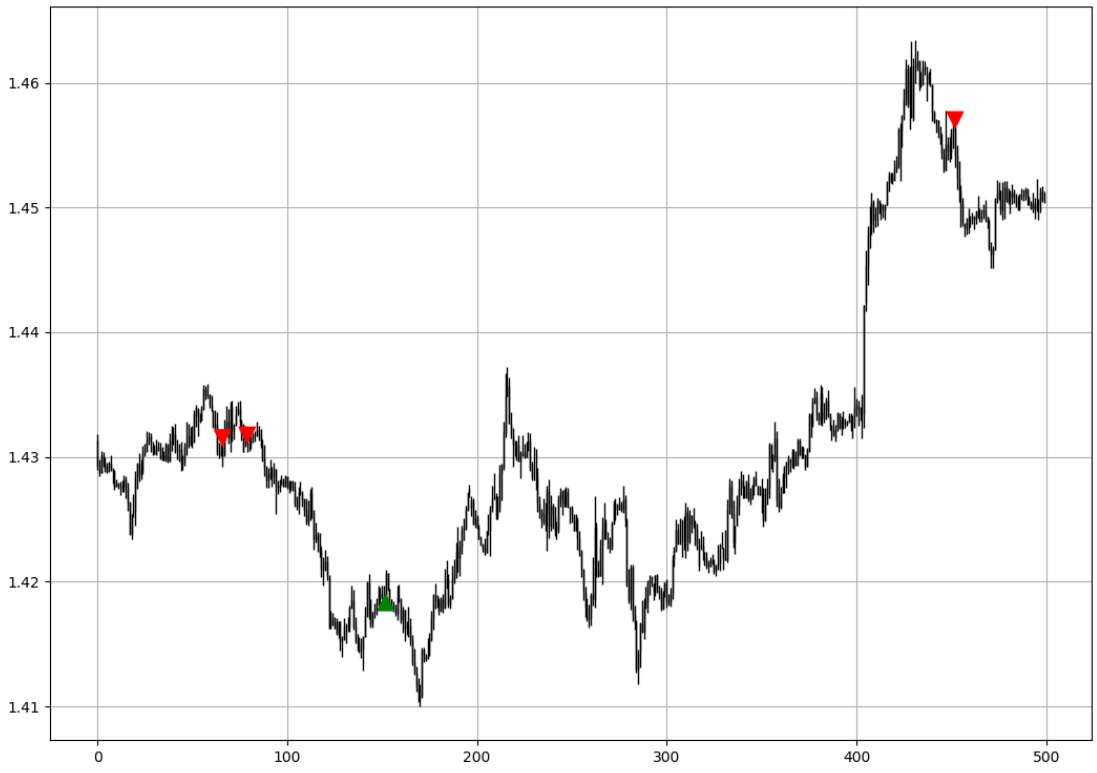


Figure 18 Signal chart on GBPUSD.

[54] Make sure you have understood what smoothing is and its difference to slowing. Refer to chapter 2 for more information.

The code required for the indicator can be found in the following snippet:

```
def rob_booker_reversal(data, high, low, close, position, buy, sell):
    # Calling the MACD function
    data = macd(data, close, 26, 12, 9, position)
    # Removing the MACD signal line
    data = delete_column(data, position + 1, 1)
    # Calling the stochastic function
    data = stochastic_oscillator(data, 70, high, low, close, position + 1,  slowing =
                                 True, smoothing = True, slowing_period = 10,
                                 smoothing_period = 10)
    # Removing the unnecessary columns
    data = delete_column(data, position + 1, 2)
    # Adding signal columns
    data = add_column(data, 5)
    # Creating the Rob Booker reversal indicator
    for i in range(len(data)):
        if data[i, position] > 0 and data[i - 1, position] < 0 and data[i, position +
          1] < 30:
            data[i + 1, buy] = 1
        elif data[i, position] < 0 and data[i - 1, position] > 0 and data[i, position
           + 1] > 70:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable high refers to the column of the high price, the variable low refers to the column of the low price, the variable position refers to the column where you place the indicator,

and the variables buy and sell refer to the columns containing bullish and bearish signals.

The indicator can be found in some charting platforms which is already a recognition that it is accepted among the trading community. However, I do not recommend using it alone.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	43.45%	0.71	0.93	214
USDCHF	42.85%	1.04	1.39	196
GBPUSD	51.51%	1.60	1.50	198
AUDUSD	51.28%	1.14	1.08	195
USDCAD	41.28%	0.80	1.14	218
XAUUSD	44.87%	0.99	1.22	156
XAGUSD	41.95%	0.88	1.22	174
NIKKEI225	41.81%	1.04	1.45	110
S&P500	49.18%	1.17	1.21	122

Figure 19 Signal chart on GBPUSD.

To summarize, Rob Booker's reversal indicator relies on a clever touch between the stochastic and the MACD oscillators. The confirmation factor may induce some lag but may also signal the beginning of strong trends. This is a high-potential strategy as it uses two robust technical indicators which is why I recommend you tweak the parameters in order to find the right balance to add to your trading framework.

STRATEGY #11 PATTERN RECOGNITION ON THE RSI

Technical pattern recognition is a mostly subjective field where the analyst or the trader applies theoretical configurations such as double tops and bottoms in order to predict the next likely direction of the market. However, they are rarely applied on technical indicators which may not be intuitive but worth a shot.

This strategy uses the RSI to scan for the double top and bottom patterns that I have presented earlier in the book. The difference is that I will show you a slightly more sophisticated algorithm which does not impose successive conditions and actually detects patterns seen with the visible eye.

It is worth mentioning that Welles Wilder Jr., the creator of the RSI has recommended applying reversal patterns on the indicator meaning that this strategy is not really new, it is just not commonly applied. The trading conditions for this strategy are as follows:

- **A long signal is generated whenever the RSI shapes a double bottom.**
- **A short signal is generated whenever the RSI shapes a double top.**

The signal function of the divergence technique is lengthy which is why I have decided it to only put it in the GitHub repository. You can access it through the link[55] or through the below QR code.

[55] https://github.com/sofienkaabar/Contrarian-Trading-Strategies

The next Figure shows an example of signals generated on S&P500.

Figure 20 Signal chart on S&P500.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	32.67%	0.79	1.64	101
USDCHF	47.16%	1.14	1.27	106
GBPUSD	41.28%	1.24	1.76	109
AUDUSD	33.98%	0.58	1.12	103
USDCAD	39.24%	0.86	1.33	79
XAUUSD	46.72%	1.55	1.77	107
XAGUSD	43.20%	0.86	1.13	81
NIKKEI225	46.26%	1.40	1.63	67
S&P500	36.53%	1.09	1.90	52

The next Figure shows an example of signals generated on USDCHF.

Figure 21 Signal chart on USDCHF.

To summarize, the strategy relies on the concept of double tops and bottoms but unlike the simplistic version seen in part 1, this algorithm waits for a clear pattern to form before validating the signals. The results show that it may be rare to find the patterns, but parameter tweaks can help increase the frequency of signals. I recommend you use the signals only if they are in the same direction of the trade.

Reminder

Remember the potential of double tops and bottoms seen earlier in the book? Try applying them as exits for this strategy. Be sure to apply the potential on the RSI.

STRATEGY #12 THE MANDI

The idea of this strategy is to transform a certain moving average into a distance measurement which is bounded between 0 and 100, thus simplifying the interpretation. Remember the normalization function you have previously seen with the stochastic oscillator?

You will now use precisely that calculation and apply it on the difference between the moving average and the market price to get an objective measurement of when the market is too far from its moving average, I call this indicator the moving average normalized distance index (MANDI).

Theoretically, the market should not deviate too much from its moving average which gives traders contrarian opportunities. To create the MANDI, follow these steps:

- Calculate a 20-period moving average on the close price.
- Calculate the absolute difference between the close price and the moving average.
- Normalize the absolute differences between 0 and 100 using the below formula:

$$x_{normalized} = \frac{x_{original} - x_{low}}{x_{high} - x_{low}}$$

Follow this syntax to calculate a 20-period MANDI.

```
def normalized_index(data, lookback, close, position):
        data = add_column(data, 1)
        for i in range(len(data)):
                try:
                        data[i, position] = (data[i, close] - min(data[i - lookback + 1:i + 1,
                                                close])) / (max(data[i - lookback + 1:i + 1,
                                                close]) - min(data[i - lookback + 1:i + 1,
                                                close]))
                except ValueError:
                        pass
        data[:, position] = data[:, position] * 100
        data = delete_row(data, lookback)
        return data
def mandi(data, lookback, lookback_trend, close, position):
        data = ma(data, lookback, close, position)
        data = add_column(data, 1)
        data[:, position + 1] = abs(data[:, close] - data[:, position])
        data = normalized_index(data, lookback, position + 1, position + 2)
        data = delete_column(data, position + 1, 1)
        data = ma(data, lookback_trend, close, position + 2)
        return data
```

The trading conditions for this strategy are as follows:

- **A long signal is generated whenever the 20-period MANDI is at 100 while the previous MANDI is below 100. Simultaneously, the current market price must be below its 20-period moving average but above its 100-period moving average.**
- **A short signal is generated whenever the 20-period MANDI is at 100 while the previous MANDI is below 100. Simultaneously, the current market price must be above its 20-period moving average but below its 100-period moving average.**

The reason I have added a long-term moving average is to filter the signals when there is a trend. Remember, even though I am taking a contrarian stance in this book, you should not forget that the trend is your friend and if you are designing contrarian strategies, make sure you do them in accordance with the aggregate trend.

For example, a buy signal from a contrarian indicator (hence, during a falling recent price action) must be validated if the current aggregate (global) trend is going up. This way, you are a contrarian because you are buying when the market is correcting to the downside, but you still retain the support of the invisible hand. The signal function can be found in the next code snippet.

```
def signal(data, close, ma_column, mandi_column, trend_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        try:
            # Bullish signal
            if data[i, mandi_column] == 100 and data[i - 1, mandi_column] <
              100 and  data[i, close] > data[i, trend_column] and data[i,
              close] < data[i, ma_column]:
                data[i + 1, buy] = 1
            # Bearish signal
            elif data[i, mandi_column] == 100 and data[i - 1, mandi_column]
                < 100 and data[i, close] < data[i, trend_column] and data[i,
                close] > data[i, ma_column]:
                data[i + 1, sell] = -1
        except IndexError:
            pass
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price, the variable ma_column refers to the column of the short-term moving average, the variable mandi_column refers to the column of the MANDI, the variable trend_column refers to the column of the long-term moving average that determines the trend, and the variables buy and sell refer to the columns containing bullish and bearish signals.

The next Figure shows an example of signals generated on EURUSD.

Figure 22 Signal chart on EURUSD.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	32.07%	0.85	1.79	106
USDCHF	35.55%	0.53	0.95	90
GBPUSD	41.32%	0.92	1.31	121
AUDUSD	43.56%	1.30	1.69	101
USDCAD	41.58%	1.72	2.41	101
XAUUSD	37.23%	1.01	1.70	94
XAGUSD	37.80%	0.53	0.88	82
NIKKEI225	32.69%	0.89	1.84	52
S&P500	47.05%	0.73	0.82	51

To summarize, the strategy combines trend following and contrarian techniques to measure the amount of outstretch between the market price and its moving average. This can be considered as a mean-reversion strategy but needs to be tweaked and the exit conditions can be enhanced. One way to exit is for the market price to return to the moving average.

STRATEGY #13 FIBONACCI RETRACEMENTS ON THE RSI

When presenting the Fibonacci moving average, I have talked about the sequence that follows the Fibonacci pattern. However, that is not the only way to use the sequence in trading. A special type of ratios can be calculated using this formula:

$$Fibonacci\ ratio_i = \frac{Fibonacci\ number_i}{Fibonacci\ number_{i-1}}$$

When doing this and approaching infinity, the results converge to 1.618 which is known as the golden ratio. Try out the below code to input the Fibonacci ratio that will tend towards 1.618 as you approach infinity.

```
# Defining the Fibonacci function
def fibonacci_number(n):
    if n == 1:
        return 1
    elif n == 0:
        return 0
    else:
    return fibonacci_number(n - 1) + fibonacci_number(n - 2)
# Print the Fibonacci ratios
for i in range(2, 20):
    fib_ratio = fibonacci_number(i) / fibonacci_number(i - 1)
    print(fib_ratio)
```

The first results you will find are as follows {1, 2. 1.5, 1.66, 1.60, 1.625, 1.615, 1.619, 1.6176, 1.6181, 1.6179, 1.61805, 1.61802, …}. Notice how it tends to converge to 1.618 the more it goes deeper in the sequence. This number has an interesting mathematical property, and it is that it is almost identical to its

reciprocal. A reciprocal is when you divide one by the number, therefore 1 / 1.618 = 0.618. Notice how 1.618 and 0.618 are almost identical. Thus, this 1.618 ratio and its reciprocal have been adopted in the world of technical analysis as reactionary retracements which I will discuss briefly. What do traders mean they say that the market has reacted from the 61.8% or 161.8% level? They are referring to a technique called Fibonacci retracements where moves are retraced to measure the amplitude and find support and resistance levels based on certain ratios, the most famous being 61.8% and 161.8%. Let's take a look at the next illustration. You can see a falling market that has shaped a bottom and is in the middle of nowhere at the moment. Using Fibonacci retracements, you calculate the length of the move from the high to the bottom and then search for the levels that can be interesting to play a reaction should the market reach them. In this example, let's stick to 61.8%.

Figure 23 Fibonacci retracement bearish example.

To find the 61.8% resistance level of this bearish move, subtract the highest high since the beginning of the drop which in this case is 120 from the lowest low in the bottom, which is 100, thus giving 20. Then, multiply 20 by 0.618 which gives you ~12.35 and add it to the bottom. Therefore, your resistance level lies at 112.35 from where you should expect a bearish reaction if the market approaches it.

Traders using the Fibonacci technique tend to sell close to the resistance level while confirming with other indicators. The most used retracement is 61.8% as opposed to 161.8% and this is precisely why the strategy will rely on only using 61.8%. The trading conditions for this strategy are as follows:

- **A long signal is generated whenever the RSI retraces back 61.8% from a leg that went from the lower barrier to the upper barrier. This insinuates a bullish continuation as the RSI has reached a support level.**
- **A short signal is generated whenever the RSI retraces back 61.8% from a leg that went from the upper barrier to the lower barrier. This insinuates a bearish continuation as the RSI has reached a resistance level.**

The signal function of the retracement technique is lengthy which is why I have decided it to only put it in the GitHub repository.

Figure 24 Signal chart on Gold.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	40.49%	1.31	1.93	121
USDCHF	49.61%	1.10	1.12	129
GBPUSD	45.31%	1.10	1.33	128
AUDUSD	39.53%	0.89	1.36	129
USDCAD	44.02%	1.38	1.75	134
XAUUSD	46.84%	1.10	1.25	111
XAGUSD	39.44%	1.00	1.53	109
NIKKEI225	38.66%	0.98	1.55	75
S&P500	44.00%	1.03	1.31	75

The next Figure shows an example of signals generated on USDCHF.

Figure 25 Signal chart on USDCHF.

To summarize, the strategy is a bit special in that it applies a subjective technique such as Fibonacci retracements on a technical indicator. I recommend you do not take the back-testing results very seriously because there are many variations on how you can design the algorithm. I suggest you take this strategy as an idea and develop it further on your own.

STRATEGY #14 BOLLINGER BANDS & THE RSI

This strategy will combine two of the classical indicators together to unite their signals into one. Basically, you are looking at coinciding signals from the two indicators. The trading conditions for this strategy are as follows:

- **A long signal is generated whenever the market price is at or below the lower default Bollinger band while the 14-period RSI is below 30.**
- **A short signal is generated whenever the market price is at or above the upper default Bollinger band while the 14-period RSI is above 70.**

```
def signal(data, close, rsi_column, upper_bollinger, lower_bollinger, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        # Bullish signal
        if data[i, rsi_column] < lower_barrier and data[i - 1, rsi_column] >
          lower_barrier and data[i, close] < data[i, lower_bollinger]:
            data[i + 1, buy] = 1
        # Bearish signal
        if data[i, rsi_column] > upper_barrier and data[i - 1, rsi_column] <
          upper_barrier and data[i, close] > data[i, upper_bollinger]:
            data[i + 1, sell] = -1
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable close refers to the column of the close price, the variable rsi_column refers to the column of the RSI, the variable upper_bollinger refers to the column upper Bollinger band, the variable lower_bollinger refers to the column lower Bollinger band, and the variables buy and sell refer to the columns containing bullish and

bearish signals. The next Figure shows an example of signals generated on EURUSD.

Figure 26 Signal chart on EURUSD.

The following table summarizes the performance of the strategy:

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	36.88%	0.90	1.54	320
USDCHF	41.00%	1.02	1.47	317
GBPUSD	38.48%	0.90	1.45	317
AUDUSD	38.69%	0.92	1.46	292
USDCAD	44.44%	1.19	1.49	261
XAUUSD	36.14%	0.76	1.34	296
XAGUSD	39.29%	0.94	1.45	257
NIKKEI225	37.77%	0.88	1.46	180
S&P500	38.94%	1.00	1.56	190

The next Figure shows an example of signals generated on AUDUSD.

Figure 27 Signal chart on Gold ON AUDUSD.

To summarize, the strategy uses concepts from technical analysis and descriptive statistics to diversify the way of finding signals. Tweaking the parameters may allow for better results on certain markets but you should try not to over-tweak so that you do not fall into the trap of overfitting[56]. I recommend you use the signals only if they are in the same direction of the trade.

[56] Overfitting is the act of predicting the past so well that the results will underperform in the future. It is a dangerous assumption where the trader creates the perfect model that shows great results in the past but says nothing about the future.

STRATEGY #15 THE STOCHASTIC OSCILLATOR & THE RSI

I will conclude the book with the simplest strategy. It uses two of the three primary contrarian indicators seen: the stochastic oscillator and the RSI. The strategy relies on a double confirmation factor by awaiting two signals simultaneously from both indicators. The trading conditions are as follows:

- **A long signal is generated whenever the 14-period RSI and the 5-period stochastic oscillator exit their oversold zone simultaneously. By default, the oversold level is 30.**
- **A short signal is generated whenever the 14-period RSI and the 5-period stochastic oscillator exit their overbought zone simultaneously. By default, the overbought level is 70.**

The next Figure shows an example of signals generated on Gold.

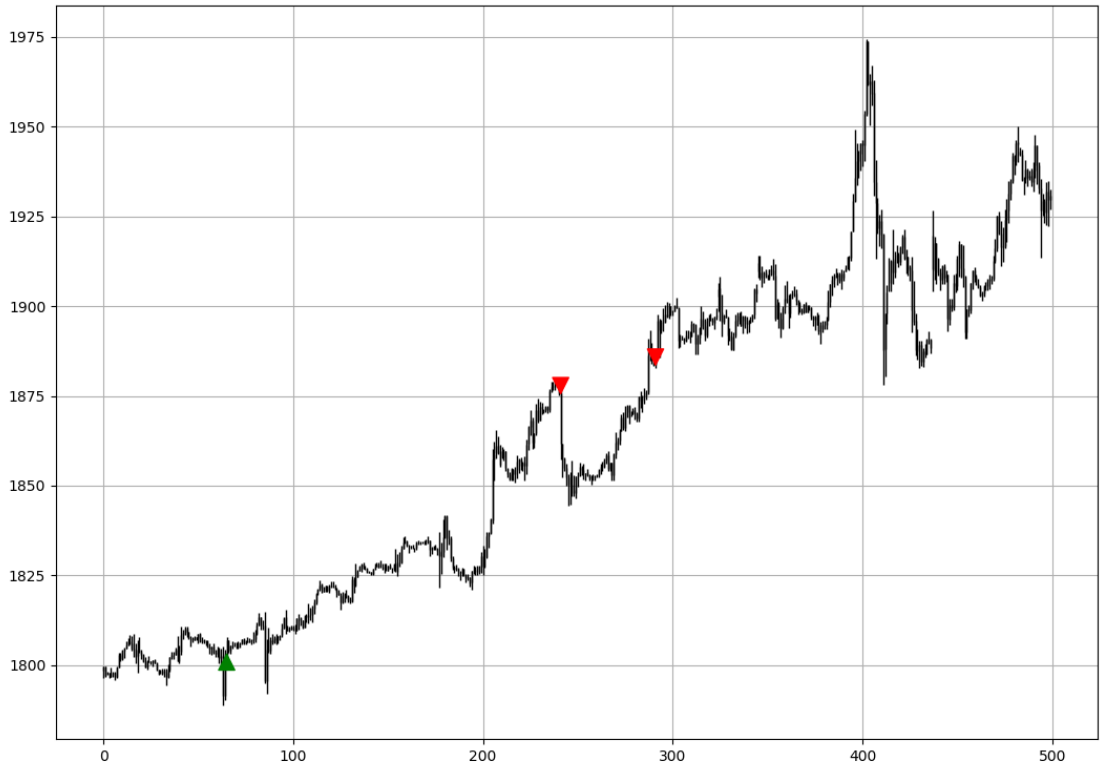


Figure 28 Signal chart on Gold.

The signal function of the strategy is as follows:

```
def signal(data, rsi_column, stoch_column, buy, sell):
    data = add_column(data, 10)
    for i in range(len(data)):
        try:
            # Bullish signal
            if data[i, rsi_column] > 30 and data[i - 1, rsi_column] < 30 and data[i, stoch_column]
              > 30 and data[i - 1, stoch_column] < lower_barrier:
                data[i + 1, buy] = 1
            # Bearish signal
            if data[i, rsi_column] < 70 and data[i - 1, rsi_column] > 70 and data[i, stoch_column]
              < 70 and data[i - 1, stoch_column] > upper_barrier:
                data[i + 1, sell] = -1
        except IndexError:
            pass
    return data
```

The function defines the signals where the variable data refers to the OHLC array containing the historical values in four columns, the variable rsi_column refers to the column of the RSI, the variable stoch_column refers to the column of the stochastic oscillator, and the variables buy and sell refer to the columns containing bullish and bearish signals.

The following table summarizes the performance of the strategy in hourly time frame since the beginning of 2020.

Element	Hit ratio	Profit factor	Risk-reward	Trades
EURUSD	39.56%	0.63	0.96	91
USDCHF	39.83%	0.89	1.35	123
GBPUSD	44.11%	1.07	1.36	102
AUDUSD	43.26%	0.71	0.94	104
USDCAD	53.68%	1.47	1.27	95
XAUUSD	50.00%	1.12	1.12	112
XAGUSD	42.85%	0.91	1.22	112
NIKKEI225	49.35%	1.27	1.30	77
S&P500	47.36%	1.07	1.19	76

To summarize, the strategy uses the conservative technique on two commonly used contrarian indicators and relies on a double confirmation factor to increase the conviction of the trade. As always, the market's regime is paramount in determining the profitability of any strategy. In a trending market, the strategy is unlikely to outperform other strategies due to the number of counter-trend false signals. I recommend you use the signals only if they are in the same direction of the trade.

CONCLUSION

Contrarian trading remains an art as valuable as trend following and while this book discusses many indicators and strategies, it is far from being exhaustive. Undoubtably, the point of the book was not to present all contrarian topics but to simply have a general view which can help the reader get ideas and apply them.

I must again point out that the strategies presented are not to be used in their raw form because they must be optimized for every market and must have solid risk management rules so that they become more stable by time.

You should be mostly concerned by the ability to profit from the reactions around certain levels and indications but not really by timing them exactly as that would be almost impossible. The previous edition of the book has covered trend following strategies and indicators and had the same aim as this edition which is to stimulate ideas and creative thinking so as to give technical analysis a new face based on objective measures and filtered indicators.

- Sofien Kaabar

APPENDIX: ALTERNATIVE DATA FETCHING TECHNIQUE

Users that want to bypass using MetaTrader5 to automatically import historical data may use an alternative albeit manual way. This entails that you already have an excel file with the OHLC data inside. The first thing to do is to make sure that the excel file is in the same directory as the one used by Python. In SPYDER, this can be done by modifying the text tab on the upper right.

Let's call the excel file **my_data** and put it in the desktop. The tab on the right may be changed to include desktop. After doing that, write the below code to import the excel file as an array.

```
# Importing pandas for data interpretation
import pandas as pd
# Importing numpy to convert from data frame to array
import numpy as np
# Using pandas to import the excel file from the desktop
my_data = pd.read_excel('my_data.xlsx')
# Converting the data frame to an array
my_data = np.array(my_data)
```

The above code will give you an array like the ones I have been presenting. The only issue is that you must provide the OHLC file yourself in the form of an excel file. Many third-party resources allow you to download intraday and daily financial data which you can clean and import.