**An end-to-end setup to trade forex algorithmically**

This document details the strategy we’ve used to trade Forex.

**Author:** [**José Carlos Gonzáles Tanaka**](https://www.linkedin.com/in/jose-carlos-gonzales-tanaka/)

**Webpage:** [**https://www.quantinsti.com/**](https://www.quantinsti.com/)

**################################## DISCLAIMER #####################################**

**This file is documentation only and should not be used for live trading without appropriate backtesting and tweaking of the strategy parameters.**

**#####################################################################################**

**Copyright 2023 QuantInsti Quantitative Learnings Pvt Ltd.**

**Licensed under the Apache License, Version 2.0 (the "License").**

**You may not use this file except in compliance with the License.**

**You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0**

**Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.**

**Table of contents**

[Introduction 1](#_cr9fzc5ypaxq)

[Setup of the variables and functions of the strategy 2](#_b73559a8t0og)

[Output files 4](#_ryefyq1h8dp7)

# Introduction

We are happy to introduce our working setup to trade Forex algorithmically with the Interactive Brokers API.

We refer to our labor of love as a Python-based setup, trading app, or similar names. We hope it’s self-evident that they all refer to the same thing!

This script allows you to execute transactions in the Forex market using a customizable strategy and swap out Forex assets as needed. The script-based application aims to teach you the basics of using a ready-made IB-API-based trading app and how it works during each trading period.

This documentation is to understand the basics of the strategy implementation in the trading app. You will learn here the steps followed in the strategy and the variables used in the same. If you want to make major changes following the functions’ outputs, you can do it. However, if you want to make changes beyond the functions’ outputs, we suggest you revise the entire script’s code to modify them as per your trading needs.

# Setup of the variables and functions of the strategy

The file name is “**strategy\_file.py**”. The functions we explain can be changed at the trader’s discretion.

* **create\_classifier\_model**: This function outputs the machine-learning model. The model is a bagging with a random-forest algorithm. The only parameter to be changeable is the seed. If you want to vary more parameters or change the model, please modify the function at your discretion. The function should output the model object. If you want the function to output something else or change the output, please modify this file and the other ones such that they run properly.
* **set\_stop\_loss\_price**: This function sets the price target for the stop-loss order.
  + signal: The signal for the market order that is obtained from the **get\_signal** function.
  + last\_value: The Forex asset's latest price available in the market. This value is obtained within the trading app. To make the function work, keep this input.
  + risk\_management\_target: Explained in the Quick implementation guide.
  + stop\_loss\_multiplier: Explained in the Quick implementation guide.
* **set\_take\_profit\_price**: This function sets the price target for the take-profit order.
  + signal: The signal for the market order that is obtained from the **get\_signal** function.
  + last\_value: The Forex asset's latest price available in the market. This value is obtained within the trading app. To make the function work, keep this input.
  + risk\_management\_target: Explained in the Quick implementation guide.
  + take\_profit\_multiplier: Explained in the Quick implementation guide.
* **prepare\_base\_df**: This function creates the database **base\_df** used to fit the model. To sum up:
  + It creates the prediction feature based on the next day’s return sign.
  + Create datetime dummy features
  + Create technical features
  + Create buy-sell signals based on rolling moving averages and standard deviations with different windows as complement features
  + Transform all the relevant previous features into their z-score versions.
  + The function returns the base\_df, the names of all input features to be used as inputs for the machine learning model, as a list, and the names of the scaled features as a list. The function should output the base\_df dataframe. If you want the function to output something else or change the output, please modify the other files, too.
  + The function inputs are:
    - df2: The historical data resampled as per the trading data\_frequency.
    - max\_window: Explained in the Quick implementation guide.
    - test\_span: Explained in the Quick implementation guide.
    - train\_span: Explained in the Quick implementation guide.
    - scalable\_features: The scaled features’ names as a list. This input is unnecessary if this function is used in the strategy\_parameter\_optimization function. This input is necessary if the function is used in the trading app.
* **Get\_signal**: This function returns the signal of the machine learning model in the trading app. This function will output the signal as a +1 or -1. This function has as inputs:
  + market\_open\_time: The Forex market opens at 5 pm Eastern time.
  + base\_df: Explained in the **prepare\_base\_df** function.
  + final\_input\_features: Explained in the **prepare\_base\_df** function.
  + purged\_window\_size: Explained in the Quick implementation guide.
  + embargo\_period: Explained in the Quick implementation guide.
  + logging: The trading app outputs a log file in which all the trading app, IB requests, and received information are saved. The log file is saved in the logging variable. Do not eliminate this variable to output the print messages inside this function.
  + This function works as follows:
    - Creates the X and y data
    - Splits the X and y data into train and test data
    - Creates the directional change R indicator to be used as an input feature for the HMM.
    - Estimation of a Hidden Markov model (HMM) and its predicted hidden states are used as an input feature.
    - Returns the long or short signal.
  + If you want the function to output something else, change the output, or change the signal creation process, please modify the other files and functions.
* **Strategy\_parameter\_optimization**: This function optimizes all the machine learning models available as per the **random\_seeds** list.
  + The function has as inputs:
    - market\_open\_time: Explained in the **get\_signal** function.
    - seed: Explained in the Quick implementation guide.
    - random\_seeds: Explained in the Quick implementation guide.
    - data\_frequency: Explained in the Quick implementation guide.
    - max\_window: Explained in the Quick implementation guide.
    - file\_address: It’s the historical\_minute\_data\_address string obtained with the main file. This string is automatically generated with the main file variables’ setting.
    - base\_df\_address: Explained in the Quick implementation guide.
    - train\_span: Explained in the Quick implementation guide.
    - test\_span: Explained in the Quick implementation guide.
  + The function does the following:
    - Get the annualized factor to compute the Sharpe ratio of each machine learning model cumulative returns.
    - Computes the test span if needed
    - Reads the historical minute frequency data
    - Creates the base\_df dataframe with the **prepare\_base\_df** function.
    - Creates the X and y data
    - Splits the X and y data into train and test data
    - Creates the directional change R indicator to be used as an input feature for the HMM.
    - Estimation of a Hidden Markov model and its predicted hidden states are used as an input feature.
    - Estimates the selected features with the Boruta-Shape algorithm.
    - For each seed:
      * Creates a machine-learning object
      * Fits the model with the train data
      * Computes the cumulative returns of the strategy
      * Computes the Sharpe ratio with the net returns
    - Gets the seed that maximizes the Sharpe ratio
    - Save the final and scaled features in an Excel file
    - Save the Hidden Markov model as a pickle file.
    - Save the best-seed model object as a pickle file.
  + If you want the function to output something else, change the output, or change the signal creation process, please modify the other files and functions.
  + The strategy is a simplified version of the one implemented in the MLT-04 lecture. If you want to implement a better strategy, please refer to that lecture to learn more about it.

# Output files

The trading app outputs the following files:

* email\_info.xlsx: Here are the details of the Gmail app email and password used by the trading app to send you the details of each period traded.
* epat\_trading\_app\_database.xlsx: This dataframe is used to save all the relevant trading information. You will have information regarding the open orders, orders status, open and close positions, executions and commissions report, the equity curve in the “cash\_balance” sheet, the time spent to run the strategy for each trading period, and the periods the app has traded. You can check it once you start live trading.
* optimal\_features\_df.xlsx: It contains all the technical indicators features names and the features used to apply a rolling z-score.
* historical\_data.csv: The historical data resampled as per the “data\_frequency” explained in the Quick documentation.
* app\_base\_df.csv: The dataframe to be used to train the model and obtain the signal.
* hmm\_model\_year\_month\_date.pickle: The Hidden Markov model (HMM) object used in the strategy.
  + Where:
    - “year”: The year in which the object was saved.
    - “month”: The month in which the object was saved.
    - “day”: The day in which the object was saved. The day will be a Saturday.
* model\_object\_year\_month\_date.pickle: The Bagging-Random-Forest model object used to predict the next signal. Where “year”, “month,” and “day” are explained above.
* app\_asset\_df.csv: Where “asset” depends on which Forex asset you choose in the main file.