# Morfeo protocol

## Abstract

## Introduction

The goal of the Morfeo protocol is to run a bot which operates with the stock market based on machine learning models.

## Infrastructure

The bot which operates in the stock market is able to buy or sell assets depending on certain conditions. These conditions are adjustable by parameters. Thus, these parameters can be optimized depending on the situation of the market. The bot uses a machine learning model to find the most optimal parameters and maximize gains.

The protocol uses a software infrastructure as describes in the figure 1.0.

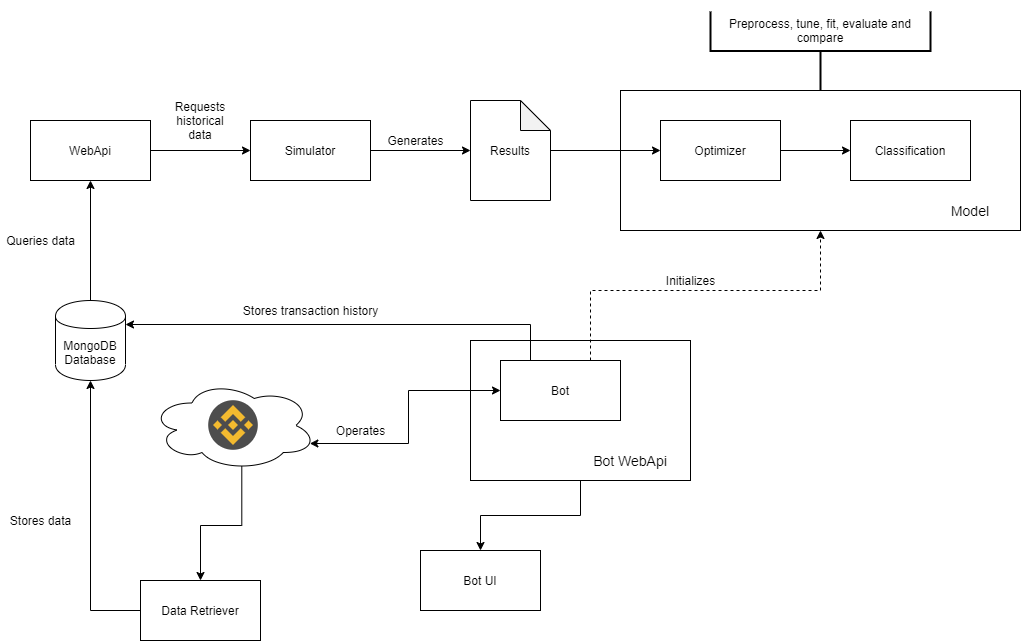


Figure 1.0 system components diagram

Such an infrastructure aims to satisfy a fluid data flow between the components. Its implementation ensures that no dependencies would lock the execution of a component. For example, if a new model is being generated with the most recent data obtained from the markets, the bot would use a previous version of the model without waiting until the update completes. Obviously, excluding the initialization of the protocol.

A requirement that the protocol follows is the scalability. That means that different models can be generated, and different instances of the bot can use them in parallel.

## Core concepts

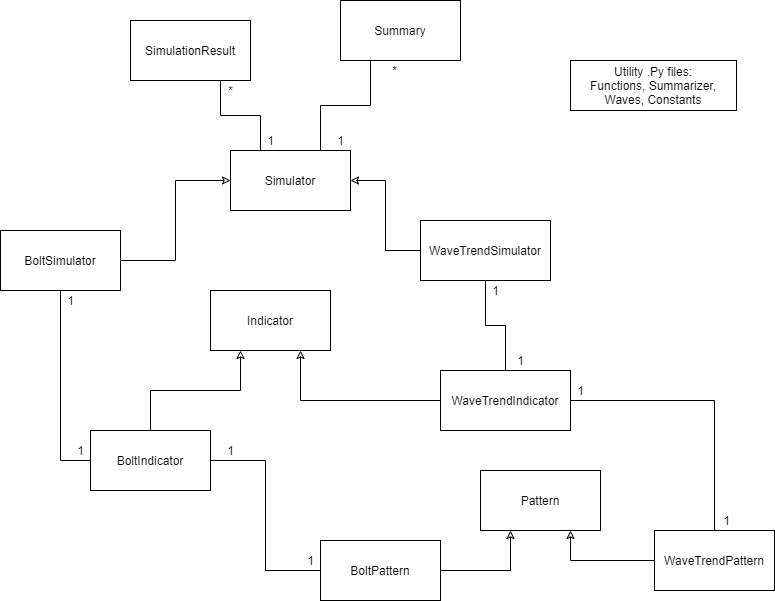
Any model that can be created is based on variables which describe statistically the market data of a certain time period, and results from simulating operations under certain conditions. The time periods used to make the simulation do never overlap to avoid overfitting when building the model. The way in which these operations are structured is described in the figure 2.0.

Figure 2.0 class diagram used to do simulations

This is a representation of the software classes used to make simulations on the marked data. It is important to note that the same classes used to make simulations, are later used to keep the state of the bot. The are three main types. The Indicator type is the central one. It is responsible of ingesting the market data. It keeps the state of a simulation in numeric attributes (e.g. initial budget, coins spent for a buy order, the calculated gains, etc). Moreover, it has an attribute of the type Pattern. Pattern classes have two methods that return Boolean values. That methods are buy and sell conditions. Finally, this is encapsulated in a Simulator class. The simulator class simply generates combinations of different versions of the patterns and ingest the same data to all of them via a generated indicator class.

## Results

## Conclusions

## Discussion

## References