KSCHOOL
Master in Data Science
edition 19

MACHINE LEARNING FOR ALGORITHMIC TRADING
An example for APPLE equity

Authored by
Isabel Puche Marín

Academic Year 2019-2020

Table of Contents

List of Tables

List of Abbreviations

TA	Technical Analysis
ML	Machine Learning
DL	Deep Learning
IDE	Integrated Development Environment

1. Introduction

1.1. Problem Statement

2. Trading and Backtesting

2.1. Concepts

According to investopedia, “**algorithmic trading** is a process for executing orders utilizing automated and pre-programmed trading instructions to account for variables such as price, timing and volume. Algorithmic trading makes use of complex formulas, combined with mathematical models and human oversight, to make decisions to buy or sell financial securities on an exchange”. Actually, machine learning models are frequently used for stock predictions that trading systems turn into signals, which can ultimately be monetised.

Hand in hand with trading, **backtesting** allows for a proper strategy’s performance and viability evaluation thus conducting a simulation and analysis for the risk and profitability of trading strategy over a period of time.

Both trading and backtesting can barely be programmed in Python with Numpy, Pandas and Scipy, but there exist already built-in libraries in Python, such as Zipline, with a somewhat large supporting community.

2.2. Libraries

According to literature, **Zipline** is one of the most mature of all currently available choices. It has a rich functionality and scales well with large amounts of data. It has been developed and maintained by Quantopian, a Boston based investment firm, aiming at untapping talent among users through open competitions. The selected strategy is allocated a sum of money and the author paid based on performance. The Quantopian website uses the backtesting engine Zipline, but the library can be installed locally to reproduce a research environment. Trading in the website is in principle easier, but robust quantitative modelling and trading with non-US equities need a local environment.

Zipline comes in with the functionality of extracting free data source from Quandl, a financial provider. It provides US equities from 1990 until 2018, the year that the service was discontinued. It also integrates other useful libraries for financial statistics (**Empyrical)** and backtesting (**Pyfolio**), and is compatible with other traditional Python libraries, such as Pandas, Numpy, Scipy and Matplotlib. Zipline also offers a wide range of built-in factors for modelling.

However, local installation is non-trivial, as it requires a conda environment installation for 3.5 (or 2.7) version of Python and it has its downsides in its use, as we shall see.

3. Methodological Approach

3.1. Trading Strategies

The project is based on two momentum strategies, consisting of exploiting upward and downward trends in the belief they will continue its current direction: a **dual moving average crossover strategy** (TA Strategy) and a **machine learning based strategy** (ML Strategy).

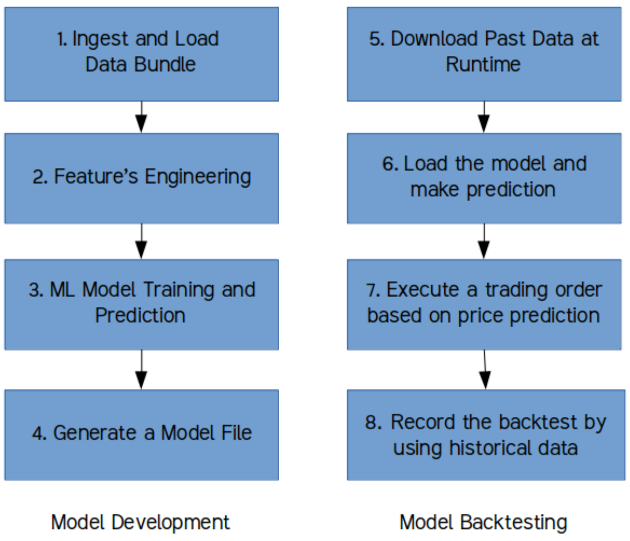
TA strategy's signals are triggered each time two rolling moving averages with different windows, slow (long) and fast (short), cross each other. When the fast moving average crosses up the slow moving average, a buying signal is recorded and vice-versa. ML strategy's signals are triggered at the end of each trading window (8 days) by comparing price prediction (with ML techniques) with (a subset of) historical data (32 days): if the prediction's maximum value is bigger than the mean of the historical sub-sample, a buying signal is activated and the other way round.

These are built on top of a *buy-and-hold strategy*, as defined by investopedia as "a passive investment strategy in which an investor buys stocks (or other types of securities such as ETFs) and holds them for a long period regardless of fluctuations in the market (…) and short-term price movements". Exit/selling conditions have not been specifically addressed.

TA Strategy will often be referred to as the **baseline strategy.**

3.2. Process Flow

In both strategies, there are two major processes —one on **model development** and another on **model backtesting**. For the ML strategy, the process flow is the following.


1. Ingest and Load Data Bundle
2. Feature's Engineering
3. ML Model Training and Prediction
4. Generate a Model File
Model Development
5. Download Past Data at Runtime
6. Load the model and make prediction
7. Execute a trading order based on price prediction
8. Record the backtest by using historical data
Model Backtesting

3.3. ML model development

3.3.1. Time Series

(creating custom bundles)

3.3.2. Multi-target Regression

(pipelines and multi-output regressor)

3.3.3. The choice of model

Notes on Random Forest, Gradient Boosting, etc

3.4. Model backtesting

3.5. Visualisation

4. Technological issues

4.1. Setting up the workspace

Following documentation, a conda environment has been created in order to (i) run Zipline in a Python version 3.5 environment, (ii) isolate Zipline's dependencies and (iii) control for possible interactions with base environment. The following libraries have also been installed: more importantly, Pandas, Numpy, Matplotlib, Seaborn and Altair, Pyfolio, Flask and Joblib. For replication purposes, check environment.yml file.

The IDE used for this project (ML trading algorithm) is PyCharm as it enables to write high quality Python code.

The Autoenv magic for automating the environment activation and deactivation has also been implemented.

4.2. Pitfalls

4.2.1. Zipline's version updates

Quantopian released in July 2020 a new Zipline version 1.4. Two issues should be then considered: (i) set benchmark to false (as the library was experiencing benchmark download problems during algorithm runs), (ii) set time to timestamp.

4.2.2. Package versions compatibility

For Python v3.5 compatibility purposes, older versions of libraries have been installed, thus limiting full potential. As an example, Scikit Learn v0.20 does not include stacking function for ensemble models. In other cases, libraries were directly not supported, such as Keras and Tensorflow for deep learning techniques and Streamlit and Altair-saver for visualisation tasks (Python>3.5).

4.2.3. Common errors

- **Pyfolio and Pandas for backtesting**

Pyfolio's backtesting tear sheets need a specific trading's report format for which context.has_position variable should be indicated in trading's algorithm handle_data function.

- **Zipline and Empyrical/Numpy interactions**

Errors during trading algorithm implementation occurred pointing to (i) returns calculation (empyrical), (ii) use of numpy’s log1p and (iii) other deprecated numpy’s functions. These errors could be in some cases partially solved, in others, test and trial error with algorithm examples.

5. Main Results and Conclusions

5.1. Main Results

- **Different strategy performance**

TA strategy records cumulative returns of 34,1%, consistent with small investors' conservative strategies, whereas ML strategy reports 6050%. Capital used at the end of the period amounts to 14.454 USD and 116.175 USD respectively.

- **Number of trading signals not comparable**

Trading signals vary in a 1:3 proportion according to trading strategy: (i) 26 signals stem from arithmetic moving averages crossover whereas (ii) 80 signals were issued from 8-day trading window (out 635 days) in ML stock's strategy. Considering that most of them are buying signals (see paragraph 3.1), more trading (buying) opportunities have arouse and more capital has fueled-in and has been capitalised in ML based strategy compared to TA strategy leading to extraordinary cumulative returns.

- **Increased volatility in ML strategy**

When the stock price trend becomes steeper from mid-2016 until mid-2017, volatility skyrockets at impossible levels, which is also reflected on the variability of the rolling sharpe (excess return over volatility) and risk exposure. The absence of shorts positions (paragraph 3.1), exit conditions, such as stop-losses, contribute to it. As a direct consequence, not only profits but also losses tend to be very large.

- **Huge drawdowns in ML strategy**

Drawdowns are huge and occur at the beginning of the trading period (until beginning 2017), when portfolio value does not compensate for losses. Whereas ML strategy's drawdowns are more dependent on volume, TA strategy's drawdowns are more constraint to stock's price variability.

5.2. Main Conclusions

Generally speaking, it can be stated that ML strategy is better than TA strategy in reaping positive momentum benefits as well as losses from negative momentum.

Nonetheless, in light of results, this seem a weak statement. In purity, both strategies are not comparable in terms of trading opportunities and capital used, all the more so, considering that basic trading features, such as the definition of short trades, stop losses, capital constraint and risk management have not been defined.

However, this project has been conceived as a *research in progress* and as such, it has the vocation to further delve into stock prediction and algorithmic trading. Possible further steps can be:

– explore different trading strategies other than *buy-and-hold* with aforementioned features

– contain volatility with VaR (Value-at-Risk) and CVaR (Conditional-Value-at-Risk) metrics, that can also be predicted with (i) parametric estimations, such as Monte Carlo estimations and/or (ii) non-parametric estimations with ML, such as SVR and KDE

– explore other machine learning and deep learning models for stock prediction

– expand features' engineering with readily available built-in factors at Quantopian's and,

– include more assets for portfolio diversification and optimisation.

Bibliography