**Stockie Version 2 Blueprint**

**Frank Kornet**

**3 March 2020**

**frkornet@gmail.com**

**Introduction**

This document explains the results of Frank Kornet’s Flatiron final project and recommends next steps for the project. It outlines, based upon the experience gained, a new blueprint for the stock trading system.

**Why a Stock Trading System?**

I have always been interested in stocks and the stock market. It was one of the main reasons for pursuing an MBA during my career. So, building a trading system is exciting and mentally stimulating.

A successful trading system provides better investment performance at a significant lower cost. A successful trading system is of interest not just to me but also to other people. That, in turn, offers additional venues to monetize a successful trading system.

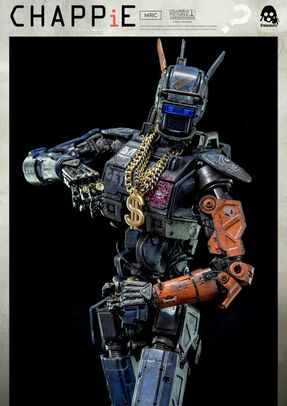
A note of realism is important here, though. The trading system under development is not designed to compete with hedge funds or other major stock market players. These players have vastly more resources at their disposal, than a single person can ever muster.

Time series data occur in many situations and time series modeling skills are highly useful.

**Product Name**

I enjoy watching science fiction movies. One of the movies that I took inspiration from is called Chappie by Neil Blomkamp (2015, South Africa). In Dutch “ie” is typically used to make something small and cute, e.g. “jochie” stands for little (and cute) boy. In honor of this movie, I am calling my trading system Stockie. The picture below shows Chappie with the name Stockie on top of it.

I also like the name Stockie since it acknowledges that this is a small system compared to systems operated by the large stock market players.



STOCKIE:

The Money-Making Machine

**Data Source**

The trading system reads the stock prices from Yahoo Finance. The system uses the python yfinance library as a wrapper for accessing Yahoo Finance. More details can be found at <https://pypi.org/project/yfinance/>.

The list of tickers was derived from Renaissance Technologies’ F-13 Filing. This is a quarterly filing that is open to the public and documents all the stocks owned by Renaissance Technologies. All large players have to file this to the SEC and are publicly accessible through the Edgar database.

The F13 filings use CUSIP numbers instead of stock tickers. Yahoo Finance uses stock tickers instead of CUSIP numbers. As a result, I had to convert CUSIP numbers to stock tickers. The following website (<https://quotes.fidelity.com/mmnet/SymLookup.phtml?QUOTE_TYPE=D>) can be used to convert CUSIP numbers to stock ticker symbols. To automate that, a web scraper has been developed.

**Initial Envisaged Blueprint**

As part of Module 4 project, I implemented market up models using ARX and GARCH (two different time series modelling techniques.) Having these models available, meant the final project had a head start. The project also had 100 stocks available from the start. Forecasting using these models still had to be implemented. The stock trading system envisaged the following architecture:



Figure 1: Initial Trading System Blueprint

**The Final Project Trading System**

The trading system implements the following functionality:

* web scraping of 2,000 stocks from Renaissance Technologies F-13 filings to SEC,
* a backtesting engine to simulate and test how well different trading strategies work,
* forecasting for the ARX and GARCH up market models,
* local minima and maxima models,
* profit and loss (PnL) functionality to track performance over time,
* buy recommendations (but not sell recommendations),
* hourly updates using the python yfinance library,
* calculating statistics per stock, and
* heuristics using statistics per ticker to decide which stocks will be traded or excluded

Learnings and findings while building the trading system:

* The performance of the system gets better as we have more stocks and that is the reason for web scraping 2,000 stocks instead of just using the 100 stocks as originally planned.
* The forecasting of the ARX and GARCH models is weak and tends to converge quickly to zero.
* The local minima and local maxima models turn out to be very good at predicting when to buy and when to sell (they make the trading system possible and allow us to implement a buy low and sell high trading strategy).
* Given the weak forecasting performance of the ARX and GARCH models, there is no need for combining the market up models and the local maxima and local minima models simplifying the buy recommendations functionality.
* Although, the trading system works well it occasionally predicts wrong. To manage the downside, the trading system uses stop-losses. A stop-loss means a stock is automatically sold if the share price drops below a threshold. The system uses a ten percent stop-loss.
* The criteria used by the backtester to decide which stock is better has a big impact on the final outcome of the trading strategy. The best results are achieved using the mean daily returns of all the gains per ticker (i.e. excluding the losses).
* Heuristic rules to manage downside are used to decide which stocks to include or exclude from trading. The best results are achieved using the heuristic rules: 1) a positive combined daily return covering gains and losses, 2) a gain to loss ratio of 70% or higher, and 3) the mean percentage gain needs to be greater than the absolute mean percentage loss. This reduces the number of possible stocks from 2,000 to 436 stocks, but significantly increases the money earned over a three-year period from around $220k to around $320k.

Note: the chart shows the best result so far (1st March, 2020). The most recent result (3rd of March, 2020) shows an overall result of $280k. Current performance of the trading system falls between 0.4 – 0.5 percent increase a day (compounded!). This may not sound like a lot but is far, far better than what the general stock market returns (175 – 250 percent p.a. against 30 percent p.a. in a good year)

A picture containing wall, indoor

Description automatically generated

Figure 2: Result of Trading System Over Three Years.

The trading system implemented is shown below in Figure 3.

backtest

PnL

possible

trades

capital

backtest\_nnn

heuristic

ticker

stats

exclude

data quality

Jupyter notebook

Python

CSV file

Figure 3: Blueprint Implemented Trading System.

The heuristic notebook uses possible trades as input to calculate the statistics per ticker. It calculates an array of statistics for gains, losses, and total. It calculates the mean, standard deviation, count, and daily compounded return.

The backtester is implemented in Python and contains all the code. The backtest notebook simply makes a call to backtester to do all the work. The notebook is mainly used for presentation purposes and easy access to the backtester.

The data quality notebook identifies stocks that have a data quality issue and exclude them from being used. Yahoo finance is returning invalid data for these stocks and are hence these stocks are excluded. Some stocks have gaps, zero prices, and interpolated values.

**Limitations and Issues**

Although the trading system works well and much better than anticipated, there are issues and limitations. These are documented below.

* All python code is currently located in backtest.py (circa 1,200 lines of code) and that becomes harder and harder to maintain. As a result, the code needs to be broken up into smaller components in different files.
* The backtester function exceeds 200 lines of actual code (no comments) and needs to be broken up into smaller chunks to make the code more manageable and maintainable longer term.
* The current trading system focuses on backtesting. It needs to be tested to demonstrate how well the trading strategy works in the future (i.e. buying today and selling at a future date)
* The system lacks sell recommendations. This functionality is needed to test how well the trading strategy works going forward in time.
* The backtesting strategy does not yet include rebalancing stock holdings, capping the holding period if a certain percentage gain has been reached, and limiting the number of trading days per trade. These features may improve the performance but they need to be tested.
* The trading system does currently not do portfolio management. The trading system limits the number of stocks we can hold to five stocks. This small number forces the system to select the “best” performing stocks based upon statistics per stock.
* The trading system does currently not schedule the backtester in the background. This has to do with issues of starting a notebook in batch. This can be done, but it is easier to schedule a python job in the background.
* The ARX model and GARCH market up models are not providing much predictive power. According to literature these models are able to predict stock prices well. A careful review needs be carried out to ensure that we are not prematurely excluding ARX and GARCH models from the system.
* Related to ARX and GARCH models is testing whether a neural network is perhaps better suited to predict whether or not the market will go up in n days and if so by how much.
* The trading system has currently only 436 stocks that meet the heuristic criteria. As performance gets better with more stocks, we need to add more stocks to the system. The next step is to incorporate the BlackRock F-13 filing.

**Version 2 Blueprint**

To address these limitations and issues, we will have to make changes to the current trading system. The envisaged blueprint is shown below (figure 4.)

Common functions will be stored in util.py. The module will contain functions to smooth the close price curve of stocks, a weight of evidence balanced scorecard, and so on. The profit and loss functionality will be provided in PnL.py. The module will track both stock details and overall results. It will also be able to save and restore its state.

The functionality to get all possible trades for the specified stock tickers will be provided in trade.py. The module will create the possible trades CSV file for a list of stock tickers. The possible trades information is used in turn by heuristic.py to generate statistics per stock ticker. The ticker stats CSV file will have a flag whether or not a particular stock is a good stock. A good stock means that the backtester can trade in the stock.

possible

trades

PnL + capital

strategy

PnL

ticker stats

exclude

quality.py

Jupyter notebook

Python

CSV file

trade

heuristic

util

Figure 4: Version 2 Trading System Blueprint

The code that exists in data quality notebook will be transported into a python file that runs periodically in the background. The program will check for stocks that contain zero close prices. It will issue a warning if new stock with problems are identified. A list of tickers is given as input.

The backtester will reside in strategy.py. The module will also contain buy and sell recommendations functionality. At the end of the run, it will print out the buy and sell recommendations. It will provide a chart of the backtesting results (saved as an image file).

Note that the system will not yet contain the ARX/GARCH models or neural network. Neither will it contain portfolio management. These will be implemented after restructuring the trading system as outlined below.

**Estimated Effort and Timeline**

Building the next version of the trading system is estimated to take two to three weeks of effort. Implementing the sell recommendations is not considered overly complex and can be handled in that period.

After implementing the new system, we can start testing how well the buy and sell recommendations work and what performance is achieved in practice.

While testing the system, we can then also review the ARX/GARCH model as well as test whether or not a neural network performs better. And, if a neural network performs better to reverse engineer the features it is using to achieve the better performance. With this information in place, we can then start outlining the functionality for version 3 of the trading system.

**Acknowledgements**

The author wants to thank Bryan Arnold for his help while building the trading system. Without Bryan’s help the system would not be as good as it is today. I also want to take the opportunity to thank Carson Lloyd for writing the web scraping functionality to convert CUSIP numbers into stock ticker symbols. This helped me a lot in downloading 2,000 stock tickers.