Long Short Portfolio Using Machine Learning

G. V. Divakar

February 24th, 2018

MLND Capstone Project Proposal

Proposal

Domain Background

Trading is one of the most challenging professions, and it is a place where even a small human error can cost millions. This is also a field that attracts huge investments from leading banks across the world. In this project, I have attempted to implement a new version of one of the most extensively used strategies in the finance world - A Market Neutral Long-Short Portfolio Strategy.

This particular strategy is very hard to optimize, using a traditional quantitative approach. As the information factored in the market is hard to extract, and is very difficult to predict how the traders in general would react.

In this project, I have attempted a different approach to create the Long-Short portfolio. Here I have tried to predict if the market would beat the day before's Highs and Lows. I will use the machine learning classifier algorithms to accomplish this.

The motivation for picking up this particular challenge is that as a trader, with 10 years of successful experience, I have developed many algorithmic trading strategies, both for the firm and myself, and in due course of the time I have trained many traders. It has been my observation that the traders find it hard to trade the trend ignoring the noise. I believe using the same inputs as a human trader, a machine learning algorithm will successfully filter out the noise and classify the trend (Up/Down) correctly.

Problem Statement

In the Long Short strategy, the portfolio will consist of stocks in two categories:

1. Long - Those which I buy
2. Short - Those which I sell

I will allocate equal money to both the categories and then rebalance the portfolio at the end of each day. By having equal money to short and to long, the portfolio will be delta neutral, in other words, our portfolio will not be affected by any unforeseen (six sigma) events in the market. I will create my own indicators that represent how traders see the price action and how they consider price rejections to build their expectations for the following day. These indicators are fundamentally based on Price Auction Theory. The classifier that I will build will use these features to train on the past data to predict the latest trend in the market. In the end, I will use these predictions to complete the strategy and backtest it's performance. I intend to use the final returns of the strategy, after deducting the commissions,tax, and transactions costs, as the basis for measuring the algorithms performance. The results can be verified and authenticated as the data that will be used will be from public domain and fetched directly from the NSE (National Stock Exchange of India).

Datasets and Libraries

I have used a python library called NSEPY, which fetches stock data from the NSE website.

The data will be in the OHLCV format: opening price (Open), highest price the stock traded at (High), lowest price the stock traded at (Low), the price at which the trading closed for that day(Close), and how many stocks were traded (Volume) for the Bank-Nifty index (it is a Banking sector Index traded on NSE) and Nifty-50. I fetch the maximum possible data that is available (since 1st Jan 1997 to current date). I will elaborate how to install this package in my Ipython notebook submission.

Apart from this library, I will be using other Python libraries:

1. Pandas
2. Numpy
3. Scikit-Learn
4. Matplotlib
5. Talib
6. Scipy

I will be using these libraries to create new features that will extract the information from the OHLCV data. In turn, these features will help the classifier predict the behaviour of the market on the following day.

Solution Statement

Given the simple daily data of any stock and its corresponding set of features, the classifier will be able to learn and predict the trend on the test dataset. The features created will be mathematical in nature and devoid of any forward looking bias. The input data will be universal in nature, meaning it should be able to accept the OHLCV data of any stock in the world and perform the predictions. Although the accuracy will vary depending on the quality of the data provided (without null or random values) and the continuous nature (should contain data for all trading days without any gaps) of the data being provided.

Benchmark Model

I will compare the performance of the strategy with a simple buy and hold strategy. The buy and hold strategy will buy one and sell the other stocks, and it will hold these positions for the entire period of testing without daily rebalancing. It is similar to the base case classifier, used in classification project, where I assumed that all the predictions on the test data are the same. In this case I assume that the prediction is a ‘Buy’ for one stock for the entire length of the testing time and ‘Sell’ for the other stock for the entire length of the testing time. Once I make a comparison in terms of a base case classifier. I will compare our strategy with other leading industry hedge funds’ performance to get a better understanding of the trend in the markets.

Evaluation Metrics

The performance of the classifier will be measured in terms of its accuracy on test data, the percentage profitability of the strategy, and the Sharpe ratio of the performance. The Sharpe ratio will give much better indication of the volatility in the predictions.

Project Design

Step 1: Import Libraries and Data
Step 2: Check data visually and mathematically for consistency
Step 3: Data Preprocessing
Step 4: Creating Functions to generate Signals(target variables) and Indicators (Features)
Step 5: Standardize and Scale the data and Justify the approach adopted
Step 6: Train the different Classifier algorithms from an ensemble to choose the best performer from the validation set
Step 7: Train the classifier using Randomized search to get an intuition about the optimization parameters
Step 8: Use GridSearch to perform a thorough search in the vicinity of the best results obtained in step 7
Step 9: Print the accuracy score to verify if the results are consistent in train, test and validation data
Step 10: Repeat Process 6-9 for predicting a different target variable on same stock data
Step 11: Perform Backtesting and plot the results to see the performance of the algorithm
Step 12. Perform Step 1 to Step 11 for another stock in the portfolio
Step 13: Combine all the predictions and create the Long Short portfolio
Step 14: Plot the portfolio performance with respect to the base case portfolio
Step 15: Justifying the results and explaining the choices made for the stocks
Step 16: Detail the future course of action and any further improvements