DOMAIN BACKGROUND

Finance is one of the leading and precursor domain where the power of machine Learning predictability was explored. Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange .

The market efficiency was debate for years, The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable. But the regular profitability and the wide expansion of quantitative hedge funds had proved the inefficiency and the predictability of the market.

The firsts models and indicators for stock price was developed since the 80's, with the advent and democratization of new technologies and the increase in computing capacity. From the firsts models like Bollinger bands developed in excel spread sheet, to models using very fast, intelligent and object oriented language for HFT, such C++ and python to develop more complex models, and finally models using statistical arbitrage with machine learning to forecast prices, develop predictive models and managing risk; modeling has become an undeniable component of the financial world.

PROBLEM STATEMENT

We will base our work to explore and apply in a real market conditions the concepts developing by the thesis of Michael David Rechenthin. However, our approach will differ from all existing paper written as this date, indeed, the most models developed are focused to forecast or prediction of the future price or the future direction of the stock based in the historical data stock. It's make sense, such the goal is to make profits by predicting the direction of the market, and then making decisions based on those suggestions to sell or buy the market.

Most prediction are based on prices and technical analysis to proved such suggestions, by following this suggestion the trader make decision to buy or sell, but in the finance domain, making trading decision is more complicated than a sample direction forecasting, it based in complex strategies. One simple example strategy will be a simple moving average crossover strategy, another strategy will be based in some technical analysis like RSI, where the law level of the technical indicator indicate that the stock are oversell, and so a buying opportunity, or the high level indicate market overbought, and so an opportunity to sell.

Our approach consist on applying the machine learning algorithm's to learn from the backtesting results of a strategies (the trades records), giving a context noted by technical indicators, to predict if this strategy is good in a such condition (Winning or loosing trade), and make a decision to let the strategy take a trade or not.

SOLUTION STATEMENT

The machine learning solution applied will exploit the wrapper approach demonstrate by Michael David Rechenthin, which is the most developed approach, dealing with the concept drifts, a specificity of trading and data streaming learning.

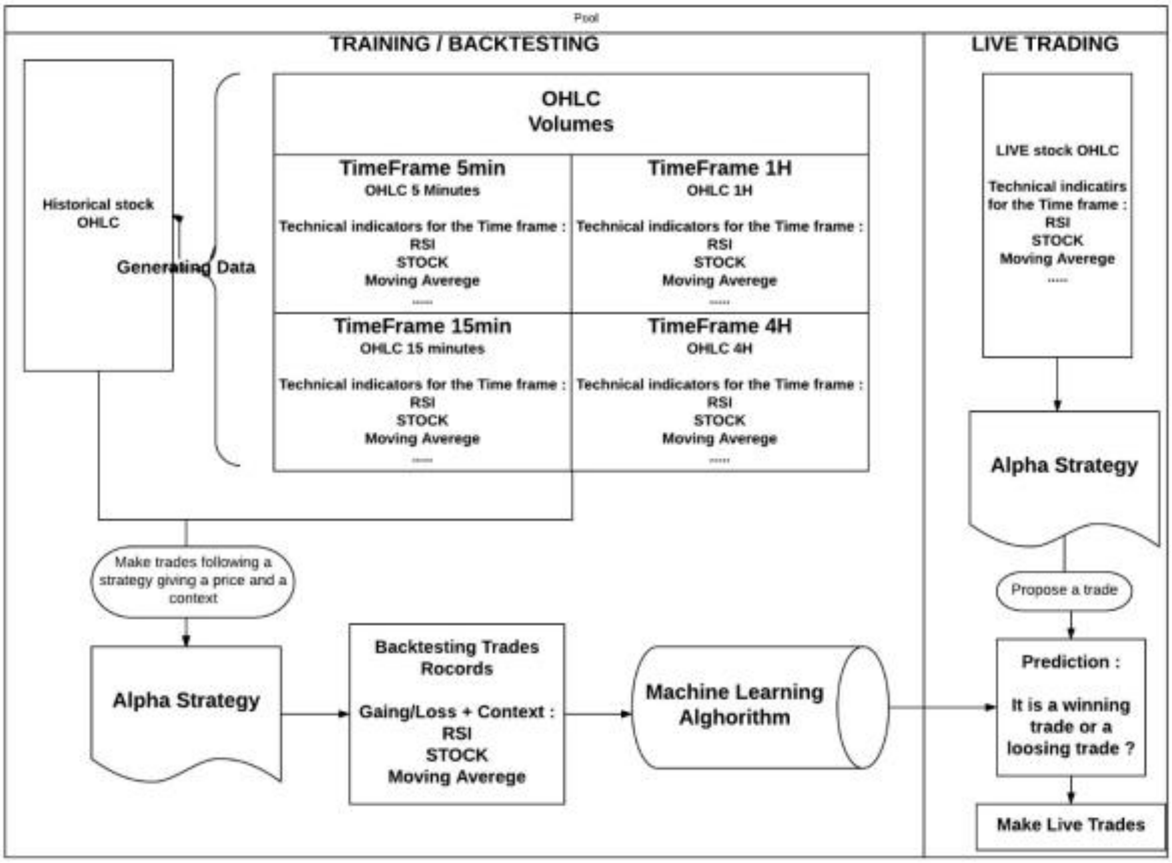
Concept drift occurs in the market for a number of reasons. For example, traders preference for stocks change; increases in a stock's value may be followed by decreases. The appearance of trends can cause other traders, not wanting to miss the price increase, to buy, thereby helping to fuel the enthusiasm for the stock and pushing it to higher levels. However, a widespread adoption of a particular trading approach is enough to drive the price either up or down enough to eliminate the pattern. This would cause a decrease in the predictability of a particular trading indicator, and thus concept drift would occur.

The changing of the underlying concept, or target variable, often complicates the learning of models from streaming data. This concept drift is unique to data streams and makes the task of keeping models relevant difficult. As the concept changes, model performance may decrease, requiring a change or update in the training data.

the wrapper method is more of a framework that uses traditional classifiers learned on chunks (i.e. subsets) of collected data. The data must be chosen in such a manner that allows for enough data for generalization, yet not too much so the learning time makes for an impractical algorithm for streaming data. Additionally, a training set that is too large may include too many concepts, thereby reducing predictability. However, if the dataset is too small, it will produce a model too quickly with poor generalizability and therefore, poor performance.

DATASETS AND INPUTS

The dataset in the input of the machine learning are generated by a strategy. The strategy makes trade by following a set of rules, it takes historical stock prices, volumes and technical indicators from multiple timeframes, and output trades by backtesting the strategy over the historical data. The structures of output trades are exposed below.



The output trades are split into features and targets and passed to the machine learning algorithm.

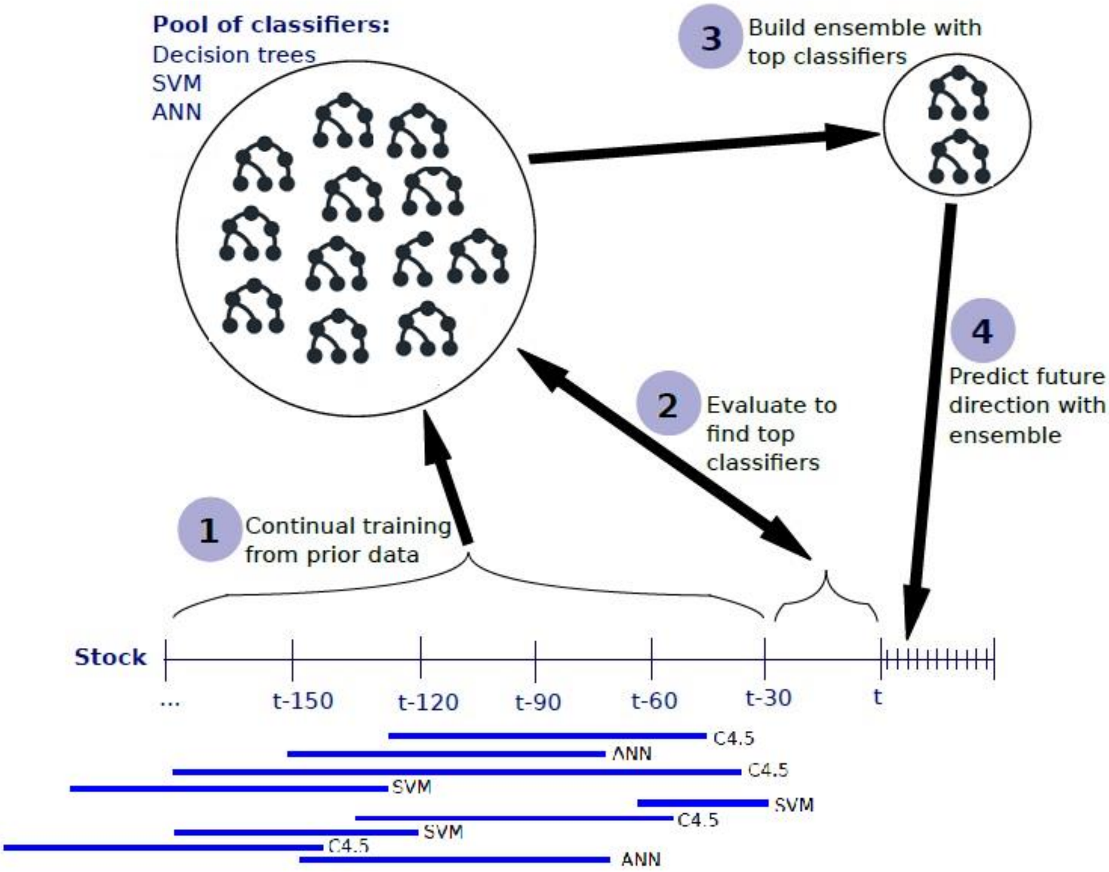
The features of dataset are construct from datestamp, some indicators by Timeframe : Moving Average of 5 min, RSI 5 min, Stochastics 5min, Moving Average of 15 min, RSI 15 min, Stochastics 15min, Moving Average of 1H, RSI 1H, Stochastics 1H ect …

The Target are the PnL of the trade (The amount of win/loss)

PROJECT DESIGN

We will assume that concept drift occurs, and to take this assumption into consideration when building the model; the actual level of concept drift or even if it actually does occur may not be measured

The first step is the training of classifiers from random-length chunks of prior data. The deepness of the data are 10 years, from 2007 to 2017. The chunk maximum and minimum sizes are chosen in such a manner that allow for enough data for generalization yet are small enough to include as few concepts as possible, the chunks are represented as chunk from 6 to 18 months. Ideally, all data within a chunk are from the same concept and generated by the same distribution, but because the future is unknown, it is unclear which training sets may be useful. For this reason the classifiers are built using random-length (and often overlapping) datasets as an attempt to include as few concepts, yet as much of that particular concept as possible.


Pool of classifiers:
Decision trees
SVM
ANN
3 Build ensemble with top classifiers
4 Predict future direction with ensemble
2 Evaluate to find top classifiers
1 Continual training from prior data
Stock
...
t-150
t-120
t-90
t-60
t-30
t
C4.5
ANN
C4.5
SVM
SVM
C4.5
SVM
C4.5
ANN

As the classifiers are built, they are added to a pool. To include additional diversity, we use tree classifier types (decision tree, SVM, ANN) along with different feature subset selection methods to both increase generalization and decrease training times. For our problem we chose two filter-based methods: Information Gain and the Correlation-based filter feature selection methods. we evaluate the performance of each classifier from the pool by testing on the instances from the most recent sliding window (one month back from t). Classifier performance is determined using a class weighted AUC. We choose the top 50 classifiers from the pool (as evaluated on the most recent data) to form an ensemble, and we use this newly formed ensemble to make a prediction on future stock price direction.

EVALUATION METRICS

The most evident metrics will be the AUC, AUC was used both for its popularity within the machine learning community and its improvement over existing methods such as accuracy or error rate. Accuracy is problematic within imbalanced datasets because the result can be high yet provide little predictive ability of the class that is important. AUC provides for a confidence of the prediction. It is this “confidence of decision” that we believe would be particularly important to a trader.

Another interesting evaluation metric will be the profitability, While the end result of predicting stock price direction is to increase profitability, the performance metrics discussed above evaluate classifiers based on the ability to correctly classify and not on overall profitability of a trading system. As an example, a classifier may have very high accuracy, AUC, etc. but this may not necessarily equate to a profitable trading strategy, since profitability of individual trades may be more important than being “right” a majority of times; e.g. making $0.50 on each of one hundred trades is not as profitable as losing $0.05 95 times and then making $12 on each of five trades.

BENCHMARCK MODEL

We will compare our result to the result proven by the thesis of Michael David Rechenthin, comparing the AUC the two approach.

We will compare too the profitability of a trading strategy with and without our machine learning framework.