Machine Learning Engineer Nanodegree

Capstone Proposal

Riccardo Rizzari
May, 2017

Project Domain: Investment and Trading

Project Title: Market sentiment prediction for Commodity prices

Domain Background

Quantitative investing and the quest for an algorithm or a mathematical formula to beat the market have a long history. One of the earliest documented attempt is the 1967 famous book **Beat the market**: *a Scientific Stock Market System* (https://www.researchgate.net/publication/275756748_Beat_the_Market_A_Scientific_Stock_Market_System) by E. O. Thorp and S. T. Kassouf.

In recent years, the number of firms that use algorithms to trade financial markets has grown steadily (https://www.ft.com/content/4b2a4a0a-2ef7-11e6-bf8d-26294ad519fc).

On the other side though, there is strong scepticism about the possibility to beat or predict the market consistently. There are also very famous theoretical results in Financial Economics on this topic, like the **Efficient Market Hypothesis** (https://en.wikipedia.org/wiki/Efficient-market_hypothesis). According to the Efficient Market Hypothesis (EHH), current asset prices in financial markets already reflect all available information. Therefore it should not be possible to profit from the market in a consistent way. Although the EHH's implication about the impossibility to beat the market has been criticized (for instance pointing out that the EHH does not explicitly say that you can't beat the market), the financial industry has widely applied the principle of the *impossibility of arbitrage* opportunities in the market. One example is the Black-Scholes model (https://en.wikipedia.org/wiki/Black%E2%80%93Scholes_model) for option pricing (particularly in the assumption that stock prices can be modelled as *martingales*).

As a Commodity Trader in the Investment Banking industry, it is of paramount importance to deploy quantitative techniques to try to understand the market trend in the near future. As a result of the specific market I trade, the analysis of the capstone project will focus on Commodity Futures markets.

Problem Statement

As we have seen in the previous section, the problem that is to be solved is whether it is possible or not to consistently predict the market. Specifically, a way of restricting the scope of the analysis is: **given the information we have today, can I predict whether the market is going up or down tomorrow?**

As anticipated above, I will restrict the analysis to liquid Commodity Futures. I will start by considering one of the most liquid and traded market: the West Texas Intermediate Crude Oil (https://en.wikipedia.org/wiki/West_Texas_Intermediate) (WTI).

A simple way to picture the problem is the following: I am an investor who is looking for an investment strategy in the Oil market. One of the very first steps I might find useful is to have a methodology to assess whether the market is going to trade higher or lower from the moment I take a position in the market. What information is available for this methodology? All present information is available. Information can be of technical nature (such as technical analysis indicators) or fundamental nature (such as data regarding the world Oil production).
In the next paragraph I will detail what kind of information I will use.

According to the Efficient Market Hypothesis, it should not be possible to beat the market. Therefore, if the EHH is correct, any algorithm or methodology to predict the market should be exactly equivalent to the process of flipping a coin (given that according to the EHH, past information does not give us any information on prices in the future, there are equal chances that the market will move higher or lower). For an unbiased coin flip, the chance that the market goes up or down are `50%`. This 50%-chance can be seen as the default metric for this problem.

Datasets and Inputs

As inputs features, I will select a bunch of technical indicators together with some data regarding trading activity. The dataset will be a time series, with every row being a collection of technical and fundamental indicators for a specific date in the past.

In particular, I will start by considering data for the Oil market. These are the columns of the dataset:

- 'Dates': given that we are working with time series, we will store the dates in the first columns
- 'y': the labels. `1` if the market on the corresponding date was up, `0` if it was down.

Price indicators

- 'CHG_PCT_1D': the percentage change of the market on that specific trading date
- 'CHG_PCT_5D': the percentage change of the market on the last five trading days
- 'PX_OPEN': the opening price
- 'PX_HIGH': the highest price during the trading day
- 'PX_LOW': the lowest price
- 'PX_VOLUME': the total volume of contracts traded
- 'PX LAST': the closing price

Technical indicators

- 'MOV_AVG_5D': the five days moving average
- 'MOV_AVG_30D': the moving average calculated over the previous 30 days
- 'RSI_9D': the 9-day Relative Strength Index, which is defined according to the following formula:
$$RSI = 100 - [100/(1 + Avg_{Up}/Avg_{Down})]$$
Where Avg_{Up} is the average of all day-on-day changes when the security closed up for the day during the period. Avg_{Down} is the average of all down changes for the period.
- 'RSI_14D': the RSI for the last 14 trading days
- 'RSI_30D': the RSI for the last 30 trading days

Volatility Market activity indicators

- 'VOLUME_TOTAL_CALL': the total daily volume traded in call options
- 'VOLUME_TOTAL_PUT' the total daily volume traded in put options
- 'VOLATILITY_10D': the 10-day realised volatility
- 'VOLATILITY_30D': the 30-day realised volatility
- '30DAY_IMPVOL_105.0%MNY_DF': the 105% strike implied volatility for options expiring in 30-days time
- '30DAY_IMPVOL_100.0%MNY_DF': the 100% strike implied volatility for options expiring in 30-days time

- '30DAY_IMPVOL_95.0%MNY_DF': the 95% strike implied volatility for options expiring in 30-days time
- '30DAY_IMPVOL_110.0%MNY_DF': the 110% strike implied volatility for options expiring in 30-days time
- '30DAY_IMPVOL_90.0%MNY_DF': the 90% strike implied volatility for options expiring in 30-days time

Dataset size

I will have a separate dataset for each commodity futures market. Each dataset will be in CSV format. Each CSV file has a size of approx 70 KB and it is formed of 500 entries, one for each trading day.

Data source

I download the data and generate the CSV file for each market via an Excel spreadsheet (named *Bbg_Market_Data.xlsm*). This Excel spreadsheet has a size of 923 KB. It contains Bloomberg Excel formulas to download data via the Bloomberg Excel API. All the features are downloaded from Bloomberg. The labels instead are calculated in the spreadsheet (essentially, by looking at the column 'CHG_PCT*1D', the label will be 1 if the daily*%_change was positive, and `0` otherwise). Once the dataset is loaded in Pandas, I will shift the labels with the Pandas Shift function, so that, at every time step, the corresponding labels will be `1` if the **next** trading day showed a positive performance, `0` if the performance was negative.

Training/Test dataset split

To produce a train and test set I will use the so called *walk-forward model validation*. This is how it works.
We have 500 time steps in total (one step is a trading day). The first train set will be made of the first (i.e. the oldest) 250 data points. The data point number 251 will be the first test set. Then we will move forward by one step. We now can incorporate the data point number 251 in the train set (we could call it 'train set number 2') and the new test set will be the data point number 252. In this way we will end up with 250 data for test sets and 250 training sets (with an increasing number of data points as we move along the time direction).

Solution Statement

The problem of predicting the market trend on a given day, given the information available from the previous days, can be viewed as a binary classification problem. In fact, for each dataset `X` representing all available information at time `t0` (this dataset `X` will be a time series), there will be a label `y` which is `1` if the market on the next day goes up, `0` otherwise. We can therefore treat the market sentiment prediction problem with the tools from *supervised learning*. My purpose is to compare different supervised learning algorithms, and calculate the accuracy score for each of the methods deployed. If the accuracy is greater than 50% (the coin flip), we have a good candidate for a market predictor. We will then check if this algorithm is robust enough to maintain an accuracy of 50% consistently, i.e. moving the time forward as the algorithm faces data that has never seen yet. We will have to check also if the success of the algorithm does not depend on the particular market chosen, say, the oil market, but is able to generalize well for also the corn market or the natural gas one.

In terms of the *choice of algorithms*, I am interested to see if by delpoying progressively more sophisticated models, we get better results, i.e. a better accuracy score.
For this reason I will start by using a simple *Bernoulli Naive Bayes*, where there is the strong assumption that features are independent of each other. This assumption is usually wrong in financial markets, as we often observe some mean reversion in the data that tend to generate some form of autocorrelation across time series of marekt prices.

Then I will investigate the performance of a more complex model: the *Random Forest*. Using an ensemble method like Random Forest should better suit a financial market context, having less chances to overfit than a non-ensemble method.
Finally I will move to the central part of the capstone, where I analyze the performance of a Neural Network. I will split this section in two sub-parts: first I will use a simpler *feed-forward multi-layer perceptron*. Then in the second part I will implement a *LSTM* network. The Long Short-Term Memory (LSTM) (https://en.wikipedia.org/wiki/Long_short-term_memory) Network should be a better candidate for time series data, given that we want the network to mantain some memory of the past when it tries to predict future outcomes.

If we see an improvement in performance by scaling up in terms of model complexity, we can get an empirical confirmation about the possibility to forecast financial markets consistently.

Benchmark Model

In the market sentiment prediction problem, we could consider three benchmark models:

1) the random walk, i.e. at every step when we try to predict the market trend, we flip a coin. At the end of the time series (*the walk*) we calculate the accuracy over the real trend we observed in the market, i.e. we calculates how many times the coin flip managed to guess the market trend correctly over the total number of steps in the time series.

2) the Naive predictors, i.e.:

- 2a) a predictor which predicts that the market is always rising
- 2b) a predictor that predicts that the market is always trending lower

However, 1) (the random walk) seems to be a more plausible choice of benchmark. Under the Efficient Market hypothesis, flipping a coin should make no difference compared to any algorithm which tries to analyze the past to predict the future. The Naive predictors could be distorted, because the market could be in fact trending higher or lower for many consecutive days, and we want our algorithm to capture this effect. On the contrary, we could have a market which trades roughly around the same levels for many days in a row. In this case, the Naive predictors would both perform poorly, while the random walk should still provide a solid benchmark to beat.

Another interesting benchmark, which is particularly suited for time series data, is the *Persistence Model Forecast* (http://ww2010.atmos.uiuc.edu/guides/mtr/fcst/mth/prst.rxml). This is a form of dummy forecast where each time step forecast simply says: 'Tomorrow is going like today'. So, whether the market went up or down in a given trading day, the Persistence model forecast will simply assume that the next day prediction is equal to the trend in that given trading day.

Evaluation Metrics

As anticipated above, the best evaluation metric for this problem is the *accuracy score* (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html). In most supervised learning applications, the accuracy score is defined as the number of data points from the test set which are correctly classified over the total number of data points in the test set. In simple terms, the accuracy score tells us the frequency of our correct predictions.

Project Design

This is the theoretical workflow I will follow in the capstone project.

1) I will start with a dataset composed of the data detailed above. I will focus the analysis on the Oil market at first. The data will be downloaded via the Bloomberg Excel API in an Excel file and exported from this file in CSV format.

2) I will load the CSV in Pandas and do some data wrangling: separating labels from features, checking for skew in the features, splitting the time series in order to have train and test datasets, scaling the features.

3) Given that at any point in time we can always dispose of all the past information, it does not make sense to push the prediction forward by more than 1 step in the future. In the project, our basic time step will be a single trading day. So, at every step, we will have a number of data from previous dates to use as training set, and one data point as a test set. This means that if the whole dataset is composed of 500 data points (i.e. past trading dates), we will start by taking as training set the set of the first 250 dates (i.e. the oldest 250 dates) and as training set the next upcoming date in the future. For each train and test set data point, the corresponding label will represent the trend on the *following* trading day . Then we will move this window forward by one step (one day), i.e. we will have the oldest 251 dates as training set (because we can incorporate all past information in the algorithm) and the next upcoming date as test set. This way we will end up with 250 test sets.

At some point we may also decide, at each single step, to include only the latest 250 data points, instead of including all available past information.

4) Once we have training and testing dataset available, as we move along the time series, we can compare different supervised learning algorithms, in terms of accuracy score over the test set (once they have been trained on the training set). First I will compare two 'standard' supervised classification algorithms: a) Naive Bayes (http://scikit-learn.org/stable/modules/naive_bayes.html), and b) Random Forest (http://scikit-learn.org/stable/modules/ensemble.html#forest). Then I will train and test a Neural Network via a Multi-Layer Perceptron (http://scikit-learn.org/stable/modules/neural_networks_supervised.html). In the final section I will turn a more complex model, the LSTM, where I will check the performance of this type of Recurrent Neural Network against all the three previous approaches and the benchmark models.

5) From the observed performances, I will try to draw a conclusion about the possibility to consistently forecast daily trends in commodity markets.
