Machine Learning Capstone Project

Klemen Čas

Date: 2016-11-23

Github: https://github.com/KlemenCas/ML_Final

Readme to the technical implementation:

https://github.com/KlemenCas/ML_Final/blob/master/README.md

Abstract

The area of focus is Investment and Trading, narrowed to trading with SP500 stocks.

Without machine learning, the approach in trading is to work with indicators and heuristics. People e.g. compare fundamentals with long term market averages, indicators like Price Earnings Ratio or Price Book Value Ratio, to assess whether a stock is over- or undervalued. Or, the investor Sentiment is being looked at, as an indicator of a bull or a bear market. Technical indicators like Momentum, Price movement around Bollinger Bands®, Moving Averages, 52-weeks Low/High Prices and many others are being compared, people trying to gain an understanding of the data that goes beyond what most other market participants know.

Statistics shows that over a longer period of time most fund managers fail to perform better than the index. Despite all efforts, the data is difficult to interpret and the heuristics often don't work. This is what makes Machine Learning interesting and this is what this project is about; to train the models on past data, limiting the human heuristics to the selection of the data, and then letting the machine build its own understanding of what the right label (e.g. expected price direction) value is, given the features.

Data is mostly sourced from Quandl, the SP500 composition originates from Wikipedia. Following information is included; stock prices, index prices, index composition, fundamentals (earnings per share and price book ratio), short selling volumes and sentiment. With the input, 16 features and 8 labels have been calculated. For optimization, the processed data has been stored locally.

Scikit-learn Support Vector Machines, Decision Trees, Random Forrest, kNeighbors, Adaptive Boosting and Naïve Bayes have been trained, on individual stock level and one the whole data set. Where grid search is available, it is being used for parameter optimization. Principal Component Analysis reduces the dimensionality of the input. Clustering helps reducing label's complexity, for forecasting of price changes. To be able to select the best performing model later on, the accuracy has been logged in a csv file. Locally saved configured models improve the response times of the forecasts.

The models do not recommend the actual buy/sell action. Q-Learning approach derives the later, by using the forecasted labels as state.

To visualize the performance the simulation module initializes two portfolios. The first one reflects the index composition throughout the simulation - this is the benchmark. The second one is being traded based on algorithm's buy/sell recommendations. The performance is the difference between the two.

In addition to the simulation a small module for individual stock prediction, with a very simple Tkinter interface, has been implemented.

Machine Learning Capstone Project

The report is divided into 8 chapters; Chapter I describes the investment approach and the how the performance of the overall implementation will be measured. Chapter II is about data retrieval and processing. The outcome is a separate pandas DataFrame for each stock, stored locally, containing features and labels. Chapter III, Training, explains how the data is being used to train the scikit models, and how the results of the training are being logged. This chapter also looks at the performance of the individual training approaches.

Chapter IV describes the simulation of the index and the trading portfolio. Chapter V focuses on individual recommendations, in which the user selects a stock symbol and the system provides a recommendation.

Finally, chapter VI, briefly summarizes the results, chapter VII sums up the various optimizations that have been done in the implementation, and in chapter VIII there are some final thoughts.

I. Investment Approach and Performance Metrics

The scenario is as following; for every SP500 index the investor is being given two times the same amount to invest in the market. One investment is like an exchange traded fund (ETF) that tracks the respective index.[1] The task of the implementation is to, daily, take the market capitalizations of the stocks and realign the ETF.

The second amount is invested in a portfolio that initially has the same composition as the index. From there onwards it is though being traded per recommendations by the machine learning algorithms. The implementation does try to stay reasonably close to the index – when there is an amount to be reinvested, the

[1] Index ETF's composition is based on market capitalizations of corporations that are included in the index. E.g. if corporations A, B and C were in the index, and A had market capitalization of 40, B had capitalization of 25 and C had one of 35, then the ETF (in its simplest form) would consist of a portfolio with 40% of the investment in A, 25% in B and 35% in C. As the valuations change, the ETF realigns its investments.

allocation to individual stocks reflects the share of the respective stock in the index – though, only stocks are being bought that the system considers worth buying and only stocks are being sold that the system recommends to sell.

The performance metric for the implementation is the relative performance vs index – the value of the investment portfolio vs the value of the ETF.

II. Data Retrieval

Figure 1 visualizes the data retrieval process and the subsequent calculation of features and labels. First step is to retrieve the SP500 index composition. This is done by querying the Wikipedia's page List of S&P 500 companies[2]. The composition is stored in 3 dictionaries:

1. sp500_ticker; with stock ticker as the key and the industry as the value
2. sp500_composition; with industry as the key and the list of included stocks as the value
3. sp500_index; with the industry as the key and the quandl access key as the value

The information is then being used to access the following through Quandl available databases: Google Finance (GOOG) for index prices, Wiki EOD Stock Prices (WIKI) for stock prices, splits and dividends, Free US Fundamentals Data (SF0) for Earnings Per Share and Price Book Value, Core US Fundamentals Data (SF1) for market capitalization, Financial Industry Regulatory Authority (FINRA) for short selling volumes and American Association of Individual Investors (AAII) for the sentiment.

[2] https://en.wikipedia.org/wiki/List_of_S%26P_500_companies

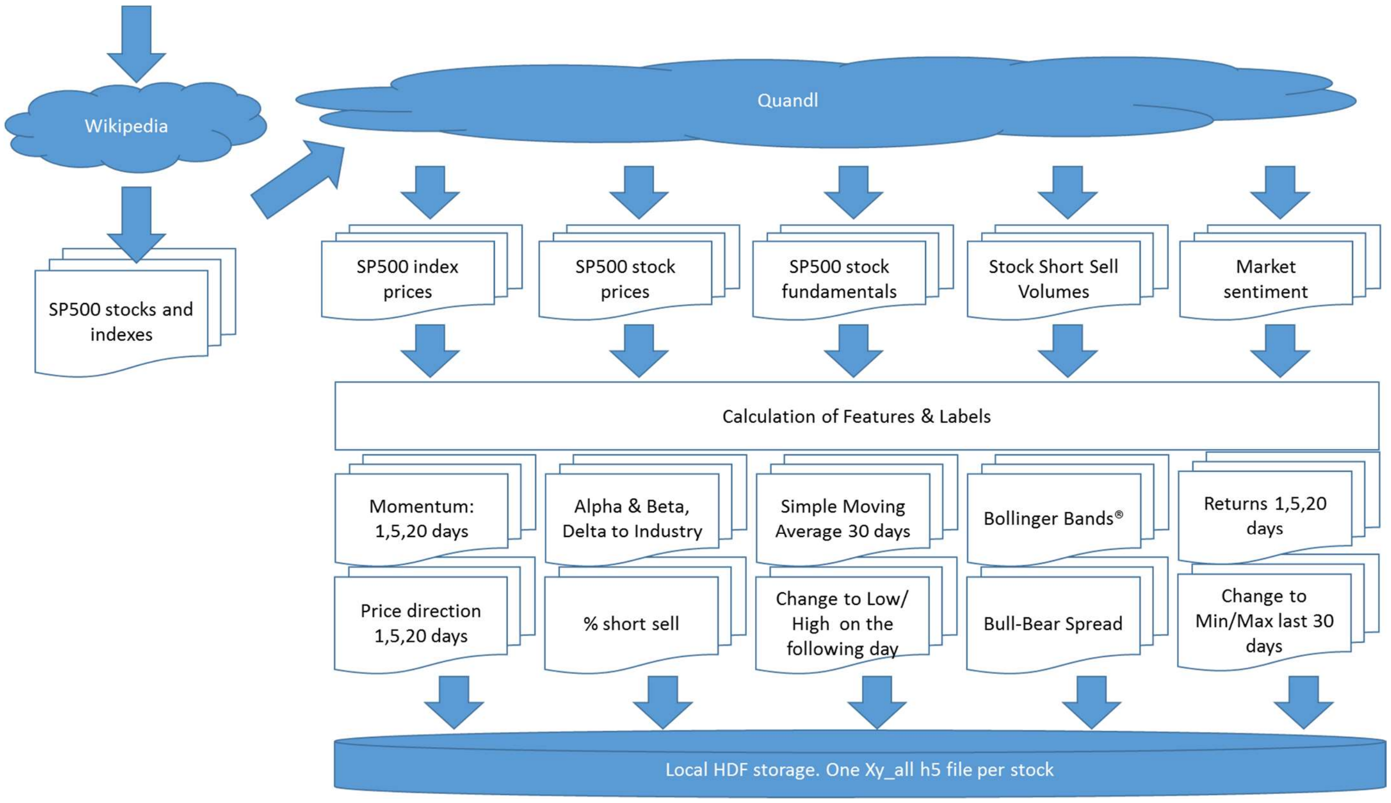

Wikipedia
Quandl
SP500 stocks and indexes
SP500 index prices
SP500 stock prices
SP500 stock fundamentals
Stock Short Sell Volumes
Market sentiment
Calculation of Features & Labels
Momentum: 1,5,20 days
Alpha & Beta, Delta to Industry
Simple Moving Average 30 days
Bollinger Bands®
Returns 1,5,20 days
Price direction 1,5,20 days
% short sell
Change to Low/ High on the following day
Bull-Bear Spread
Change to Min/Max last 30 days
Local HDF storage. One Xy_all h5 file per stock

Figure 1:Data Retrieval Process[3]

The retrieved data is loaded into pandas DataFrames, which are stored locally. Delta handling takes the local storage into account, so that at each future update it only queries the missing delta from Quandl. If there was a dividend payment or a stock split, then the complete stock history is being reloaded.

A few data source related observations:

1. *SP500 index prices* are available daily[4], with Open, Close, High and Low. The system uses them to calculate Alpha and Beta, to subsequently look at the deviation of an individual stock's price from the alpha/beta based expectation.

[3] The python module *mydata_get_delta.py* organizes the process. To reduce the data volume and the processing times, a demo mode has been implemented. It's controlled by the flag *demo_scenario* is the module *commons.py*.

[4] *Daily* as on every trading day, not on every calendar day.

2. *SP500 stock prices* are available daily. They contain the Open, Close, High and Low prices, in two sets; one set are the actual prices of the day. The other set are the adjusted prices. The adjustment is in recalculating historic prices by taking paid dividends and stock splits or reverse stock splits into account. Our implementation works with adjusted prices, hence allows historical comparisons.
3. *SP500 stock fundamentals* are updated daily, but, in contrast to the first two sources, the data is not available for every day; it is only populated for the day of the announcement and we need to treat the NaN values by first back- and then forward-propagating.
4. *Stock Short Sell Volumes* contain the percentage of the stocks that have been short sold.[5] Data is available daily.
5. *Market Sentiment* shows the ratio between the people who believe that the market will increase and the people who believe that the market is about to fall. Data is available daily.

Most NaN values are in the fundamentals data – due to how the source has been designed; it only shows values on the day of the announcement. Though, there can also be NaN values in other sources. This happens for example when, for whatever reason, a certain stock is not being traded on a certain day, or when short selling volumes are not reported. To handle the missing entries data is being back- and then forward-propagated. This is in line with the thinking that reported figures apply to the past, and only in cases when we don't know a certain value today, we look for the last known indication.

In next phase the system calculates the features:

1. *Momentum* for 1,2 and 5 days; the % change in the Close price over the last 1, 2 and 5 days,
2. *Price Book Value*, compared to industry average,
3. *Price Earnings Ratio,* compared to industry average,
4. *Alpha and Beta* of the stock, to calculate *Delta to Industry Development* for the past 1, 2 and 5 days,
5. *Delta to Simple Moving Average* of the last 30 days,
6. *Bollinger Bands®,* the area within two standard deviations of the simple moving average*,*
7. *Volatility* of last 30 days,

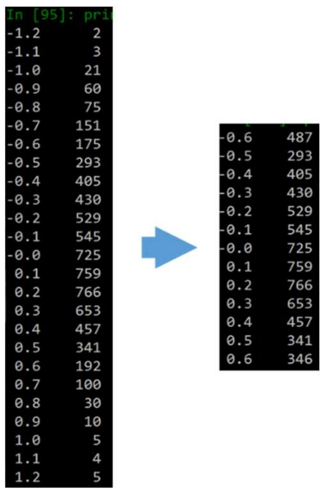
[5] Short selling is the transaction where someone borrows a stock and sells it, assuming that the price will fall before the date when the stock needs to be returned. If this happens, then the seller can repurchase the stock at a lower price, return it to the lender and keep the profit. Obviously, if the stock rises in-between the borrower will make a loss.

8. *Delta to Min and Max* of the last 30 days,
9. *Percentage of Short Sell,* NASDAQ and New York Stock Exchange,
10. *Bull-Bear Spread*, the difference between the percentage of investors that expect the market to rise (bulls) vs the percentage of investors that expect the market to fall (bears),

and the labels:

1. *Direction of Price Change* next 1,5 and 20 days
2. *Price Change from Close to Low* next day
3. *Price Change from Close to High* next day

The analysis of the data showed that there tend to be a few outliers that effectively skew the range and are more exceptions than the rule in how price develops. To cut these tails, the function *value_counts()* has been used to bucket the values. Next the value of the records that belong to the top and the bottom 5% of the buckets are set to the closest value inside the acceptable 90%. Figure 2 shows an example.



```
In [95]: pri
-1.2        2
-1.1        3
-1.0       21
-0.9       60
-0.8       75
-0.7      151
-0.6      175
-0.5      293
-0.4      405
-0.3      430
-0.2      529
-0.1      545
-0.0      725
 0.1      759
 0.2      766
 0.3      653
 0.4      457
 0.5      341
 0.6      192
 0.7      100
 0.8       30
 0.9       10
 1.0        5
 1.1        4
 1.2        5
```

```
-0.6      487
-0.5      293
-0.4      405
-0.3      430
-0.2      529
-0.1      545
-0.0      725
 0.1      759
 0.2      766
 0.3      653
 0.4      457
 0.5      341
 0.6      346
```

Figure 2: Example; cutting the tails for Bollinger Bands® for stock FTR

In the next step scikit's MinMaxScaler transforms Momentum, SMA, Min, Max, Volatility, Bollinger Bands®, Bull-Bear-Spread, Price Book Value, Earnings Per Share and Delta to Industry Returns to the range 0..1. The idea behind this is that this way the data is being scaled to a unified range and hence differences in the range of individual features do not influence the algorithms.

The transformations are done on by stock base, hence there is no cross dependency between stocks.

Figure 3 shows as an example correlations between the features for the stock FTR. There is high correlation between min/max and the standard moving average. Equally there is, not surprisingly, high correlation between the momentums. In a further evolution of the implementation it might make sense to merge the 1, 2 and 5 days momentum into one feature. Equally likely was the correlation between the short selling and the sentiment (Bull Bear Spread), whereby it only shows for NYSE. This is presumably due to less trade or less liquidity for stock FTR at NASDAQ (for APPLE the correlation is the same for NYSE and for NASDAQ).

The heatmap does not suggest that there are redundant features, but there is possibility to further refine the model, and it makes sense to train with and without PCA and to then compare the results.

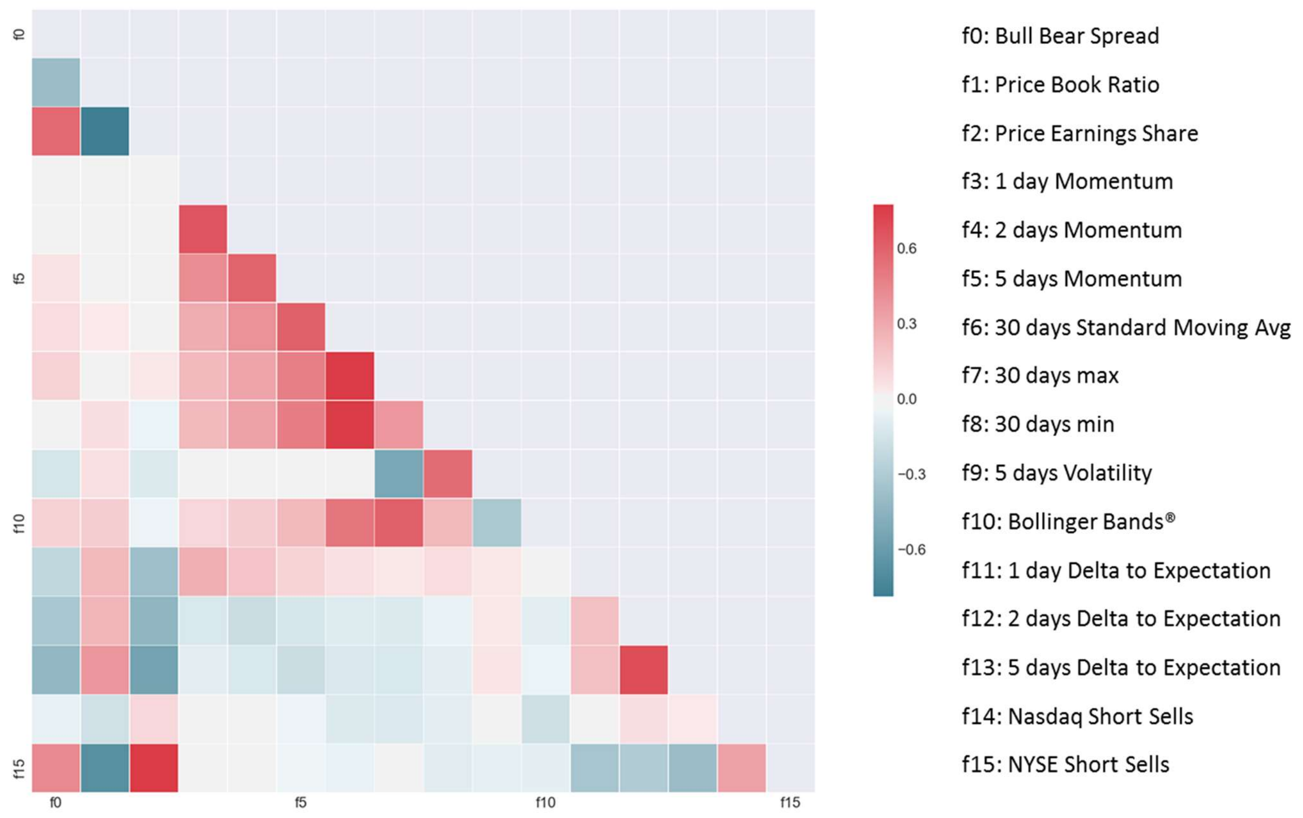


Figure 3: Features correlation heatmap

The output of the process is one hdf file by stock symbol, named Xa_all_<symbol> and stored locally.

III. Training

The input for the training are the Xy_all hdf files from previous chapter. The task is classification. Six machine learning algorithms have been selected from the scikit learn: Support Vector Machines, Decision Trees, Random Forrest, AdaBoost, kNeighbors and Naïve Bayes. The selection is primarily based on fact that they can all classify, there is no preference for one of the algorithms. One can assume that Random Forrest will in general perform better than a singular Decision Tree, but the assumption is not built into the implementation.

Figure 4 displays the flow.

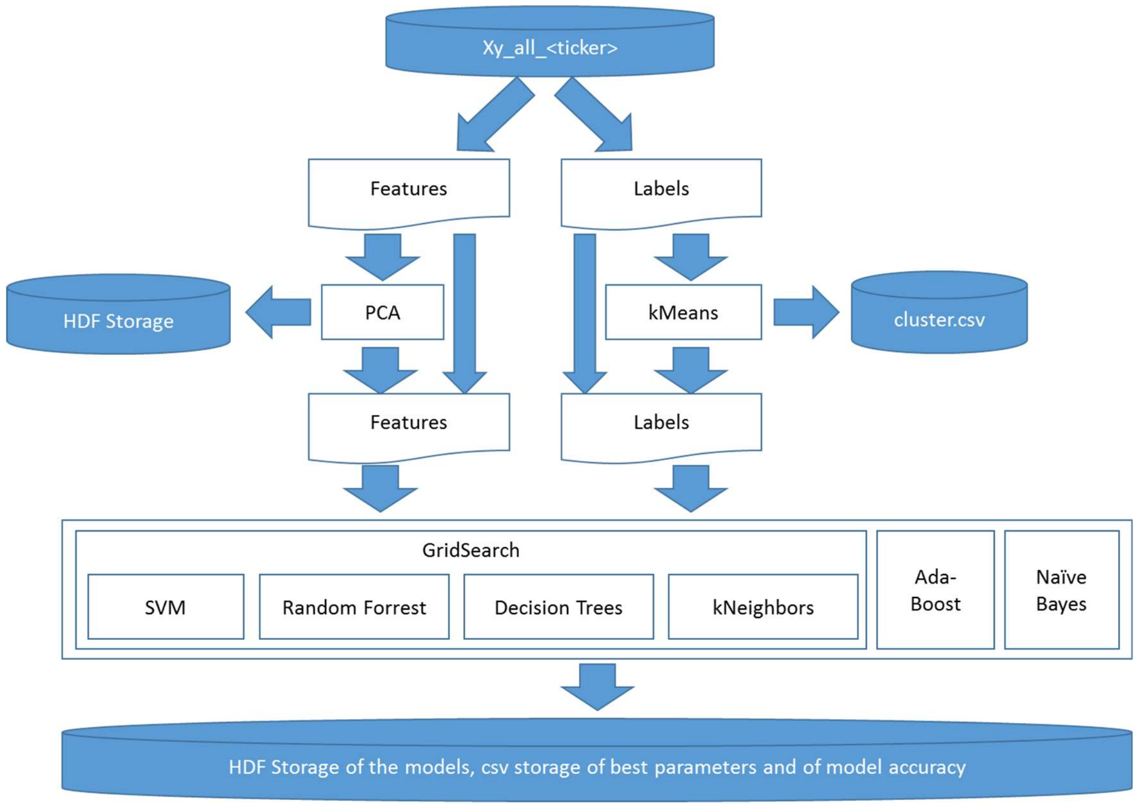


Figure 4:Process Flow for Training[6]

All 6 algorithms are being trained, and the accuracy logged. The forecasting will later select the best performing algorithm from the log (see further below).

For SVC, Random Forrest, Decision Trees and kNeighbors optimal parameters have been determined with the help of Grid Search. For some settings trainings with PCA transformed features – from 16 to 9 dimensions – delivered a higher accuracy, presumably due to less overfitting of the models.

[6] The training on stock symbol level is performed by the module *train.py*. The generic training, for models that train with data to all stocks of an industry takes place in *train_generic.py*.

The number of values for the Close to Low and Close to High price change has been reduced by kMeans clustering. The training and the forecasting is hence not predicting the exact percentage change anymore, but only the affiliation of the percentage change with one of the 5 clusters.

The accuracy with a certain model/parameter setting, the specific trained model and the PCA object have been logged and stored locally as a pickle object (the configured models) or in a csv file (accuracy statistics). This way the best performing setup can be loaded during forecasting and reused.

The training started with trying out different numbers of PCA dimensions for 20 stocks. This indicated that in case of a dimension reduction 9 dimensions delivered the best results. Therefore, subsequently, the whole dataset has been trained twice; once without a dimension reduction and once with 9 input dimensions. In total, more than 75,000 trainings have been performed.

On the average dimension reduction delivered slightly higher accuracy, 73.5% versus 72.5% (Figure 5). The labels are:

1. *x dd_Close:* Direction of price movement, x day(s) into the future
2. *chr_cluster_x:* The percentage change between the Closing price and the High price on the follow-up day in the cluster x
3. *clr_cluster_x:* The percentage change between the Closing price and the Low price on the follow-up day in the cluster x

Average of value	**Column Labels**	
Row Labels	**0**	**9**
1dd_Close	51.0%	51.6%
20dd_Close	63.6%	64.1%
5dd_Close	56.1%	56.9%
chr_cluster_0	81.9%	83.8%
chr_cluster_1	60.9%	62.5%
chr_cluster_2	65.1%	65.7%
chr_cluster_3	85.2%	85.9%
chr_cluster_4	96.2%	97.1%
clr_cluster_0	96.6%	97.7%
clr_cluster_1	86.7%	87.5%
clr_cluster_2	66.7%	67.5%
clr_cluster_3	60.1%	61.4%
clr_cluster_4	79.6%	81.0%
Grand Total	**72.5%**	**73.5%**

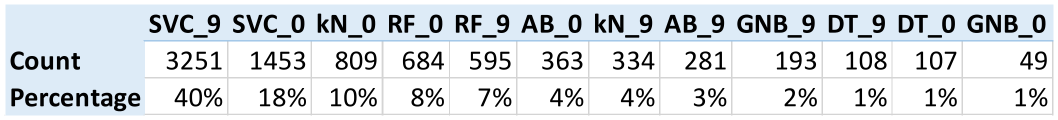
Figure 5: Accuracy with no dimension reduction (column 0) and with reduction from 16 to 9 dimensions

As for the models, on average SVC delivers the highest accuracy (Figure 6).

	AB		**DT**		**GNB**		**kN**		**RF**		**SVC**	
	0	**9**	**0**	**9**	**0**	**9**	**0**	**9**	**0**	**9**	**0**	**9**
Average of value	**72.2%**	**72.3%**	**72.0%**	**72.1%**	**68.7%**	**73.0%**	**72.6%**	**72.4%**	**73.7%**	**73.8%**	**75.9%**	**77.2%**

Figure 6: Accuracy, broken down by model

These though are average values, from which the individual cases deviate. As one can see in the *stats for report.xlsx* file in the project directory, in more than 40% of the setups a model different to SVC delivered the highest accuracy (Figure 7).



	SVC_9	SVC_0	kN_0	RF_0	RF_9	AB_0	kN_9	AB_9	GNB_9	DT_9	DT_0	GNB_0
Count	3251	1453	809	684	595	363	334	281	193	108	107	49
Percentage	40%	18%	10%	8%	7%	4%	4%	3%	2%	1%	1%	1%

Figure 7:Number of highest accuracies by model and the number of input dimensions

IV. Trading Simulation

The simulation module picks the last 6 months[7], assembles an index portfolio based on the index composition, as described in the data retrieval chapter, and an identical simulation portfolio. It does this 10 times. For each day in a simulation run there is an update calculation:

1. The Index Portfolio is being reassembled based on the market capitalization of the stocks on the day before.
2. The Simulation Portfolio is being changed according to the labels forecasted by the models and a Q-learning approach in the mapping from the labels to the actual buy/sell actions.

Figure 8 shows the flow in detail; The simulation runs by the index, and by an individual stock within the index. The overall performance is the comparison between the value of the index portfolio and the simulation portfolio. After the index and the symbol have been selected, the accuracy statistics from the training phase is being used to select, load and execute the best model. The labels part of the Xy_all HDF storage is being ignored.

The model predicts the state of the stock symbol; the expected direction of the price movement for the next 1,5 and 20 days and the affiliation to Close-Low and Close-High clusters.

[7] The time span is controlled by the start date in *market.py*.

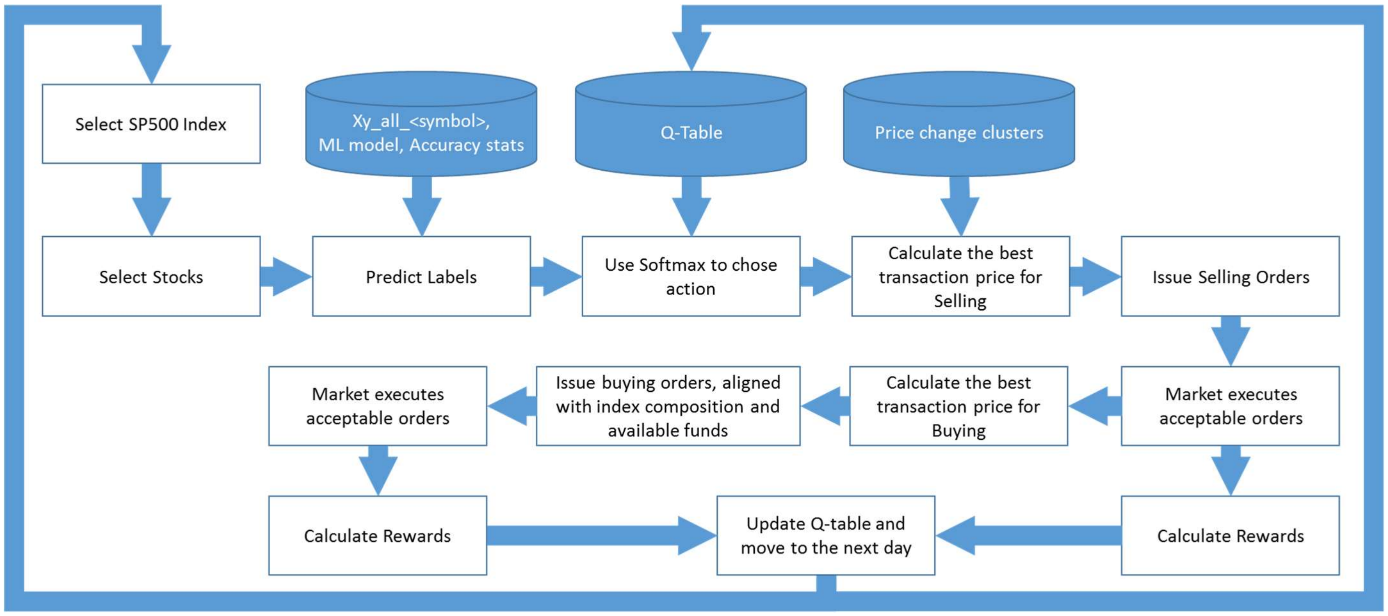

Select SP500 Index
Xy_all_<symbol>, ML model, Accuracy stats
Q-Table
Price change clusters
Select Stocks
Predict Labels
Use Softmax to chose action
Calculate the best transaction price for Selling
Issue Selling Orders
Market executes acceptable orders
Issue buying orders, aligned with index composition and available funds
Calculate the best transaction price for Buying
Market executes acceptable orders
Calculate Rewards
Update Q-table and move to the next day
Calculate Rewards

Figure 8: Simulation Flow

Softmax determines the recommended action, with which then the order price is being calculated, based on the price change clusters. This is the price with which the transaction will be issued. The market will reject it, if it's not above daily Low (for buying orders) or if it is above the daily High for selling orders. To avoid rejections and to optimize the price, two more checks are in place;

1. if the suggested price for a buying order is above the opening price of the follow-up day then the order price is reduced to the opening price. This assumes that the orders are only being released after the markets open. Similarly, for selling orders, if the selling price is below the opening price, then the order price will be lifted to the opening price.
2. If the order has not been executed by x minutes before the markets close, then the order price will be changed to the anticipated close price. The algorithm assumes that this is, on average, not more than 10 basis points from the actual closing price (= .1%).[8]

[8] The 10 basis points are picked arbitrarily, and would need to be verified with actual intraday stock movement statistics.

Selling order always contains the full volume of the stock in the portfolio. The buying orders are only being released, after the selling orders have been executed and the system knows how much cash it has available for the purchases. Once released, the purchase volume by stock is being calculated. The aim is to have a portfolio that is close to the index. Hence, the system first calculates how much of each recommended stock – given the overall value of the portfolio – it should purchase, to realign with the index portfolio. Once that is executed, it purchases all recommended stocks for the remaining amount in the same distribution as what the index portfolio has.

At the end of a trading day the cash position can be non-zero. This happens because the alignment of the buying list is performed at the beginning of the day, based on the planned purchase order price. The actual price can deviate though, either due to lowering to the opening price, or due to lifting to the close price towards the end of the day.

The rewards for Q-Learning are calculated at each requested transaction:

1. Close price has risen and the recommendation was Buy; Reward= +200
2. Close price has risen and the recommendation was Sell; Reward= -200
3. Close price has fallen and the recommendation was Buy; Reward= -200
4. Close price has fallen and the recommendation was Sell; Reward= +200
5. Close price has not changed: Reward = 100

With the reward, alpha=.5 and gamma=.8 the q-table is then updated.[9]

[9] Gamma and Alpha have been determined by running a simulation over gamma 0..1, and alpha 0..1, both in .1 increments. For each combination five consecutive runs have been calculated, with q-table being reinitialized before the first run. On average, best performance was at gamma .6, .8 or .9, and alpha .1, .3 or .5.

The combination alpha=.5 and gamma=.8 did not have the highest performance – it was the third highest. If one though summed up the performance of the 3x3 matrix around it (gamma.7 to .9, alpha .4 to .6), then it was by far the highest overall number. Hence the assumption that the combination leads to the most robust results, that there is the least luck through softmax randomness involved.

V. Individual Stock Recommendation

The individual stock recommendation is a two-step process. First the missing data must be queried from Quandl, to update the feature tables (Xy_all<symbol>, see chapter I to data retrieval). This is a time-consuming step that is being executed before the actual user forecasts are being ran. Once done, the user interface can provide stock price recommendations for the day[10].

Another difficulty with the daily update is that two of the data sources are not free, and can only be queried by having the Quandl subscription in place. Hence, for demonstration purposes, the user forecast date has been set to Oct 31st 2016, which is after the training for all models.[11] These have been trained with data up to Oct 7th.

As for the process, it's a special case of what has already been described in the chapter about simulation:

1. The user selects the stock symbol that she would like to have forecast to. (Figure 9)
2. The system loads the feature data and the per training statistics best forecast model
3. Model is being ran and the user receives the forecasted direction of the price change, cluster centers to the Close-to-Low and Close-to-High forecasted percentage change and the expectation in which cluster the change will fall in.

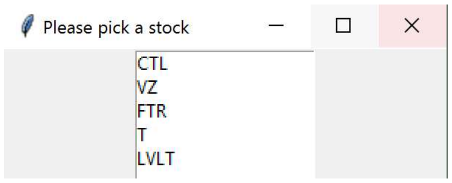
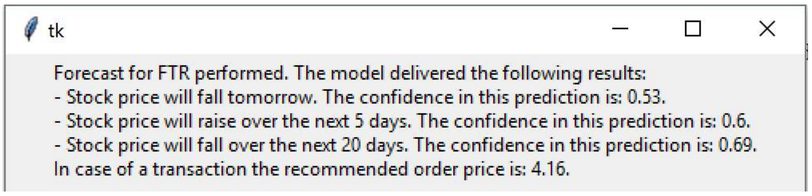


Figure 9: Stock selection



```
Forecast for FTR performed. The model delivered the following results:
- Stock price will fall tomorrow. The confidence in this prediction is: 0.53.
- Stock price will raise over the next 5 days. The confidence in this prediction is: 0.6.
- Stock price will fall over the next 20 days. The confidence in this prediction is: 0.69.
In case of a transaction the recommended order price is: 4.16.
```

Figure 10: Forecast

[10] The recommendations are called by executing the module *UI_forecast.py*.

[11] See *self.demo_date* in *UI_forecast.py*.

VI. Model Performance

Our benchmark is the development of the reference SP500 index (see also chapter I, Investment Approach and Performance Metrics). We manage a portfolio that tries to align to the index composition, and trades (buy/sell) based on the recommendation by the machine learning algorithm.

Simulation results indicate performance above the index; for the SP500-50 the model delivers roughly 20% better results than the index (see Figure 11).[12] The reliability of this number depends though heavily on the assumption that the trading strategy is feasible in real markets. Four hypotheses need to be verified:

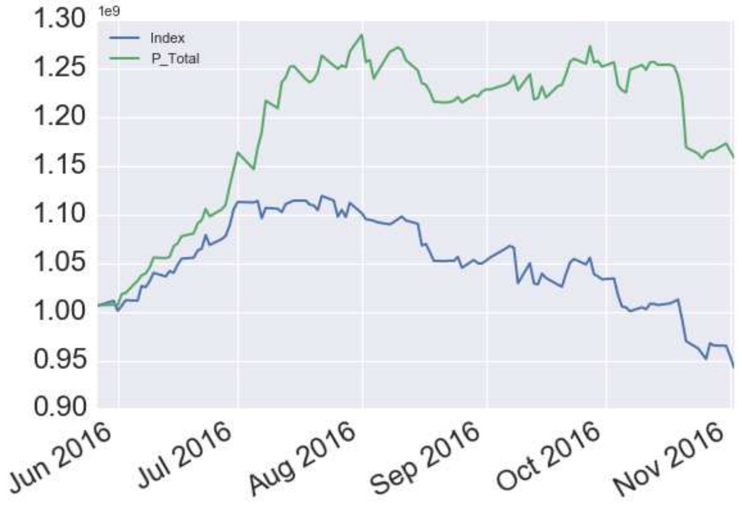


Figure 11:Examplary performance. (green=ML portfolio)

1. Stocks that are to be sold can be sold at the market opening price. This is for instances where the forecasted High price is below the opening price.
2. Stocks that are to be bought can be bought at the opening price, for instances where the forecasted Low price is above the opening price.
3. A transaction envisioned shortly before the market close can, on average, be executed within 10 basis points of the actual Close price.
4. The model is not overfitting on the historical data.

[12] See csv files *1performance_log ..10performance_log* in */stats/* directory.

VII. Model Optimizations

With regards to the documentation of how the algorithms have been improved over time, several modifications had to be made to achieve the above results.

The accuracy was better once the percentage based Close-to-Low and Close-to-High forecasts have been changed to forecasts of cluster affiliation. The latter is sufficient; we use the cluster centers to determine the boundaries (mid of the distance) and determine the expected transaction price.

Relying solely on the expected cluster for the Close-to-High and Close-to-Low price changes does not work though. The errors in the prediction lead either to

1. a trade not being executed due to the system expecting a price that is above the market, when selling, or below the market, when buying, or
2. they lead to suboptimal transactions. The latter means that the even the opening price would have been more beneficial than the forecasted one,

hence the system now executes either at the opening price, or if by shortly before the end of the trading day there is no transaction, at the closing price (+/- 10 basis points).

The actual algorithms have not been optimized in the process, as the implementation was with Grid Search from the beginning. There is probably further potential, it is time intense though – one training run for 500 stocks on an average PC takes several days of processing. Obviously one could work with a subset of stocks.

PCA has been used, but this was again with the same approach as what has been taken for the algorithm training; train with different PCA and see what performs best. The correlation between the features and the PCA itself did indicate that there was potential for improvement.

Finally, the simulation ran for alpha and gamma in the range 0 to 1, with .1 increments. This way the optimal combination has been identified.

VIII. Final Thoughts

The implementation delivers a much better performance than the index, but a lot of further verification and kind of quality assurance would be needed to trust the results. A challenge is computing power. It took several days to train the models for all 500 stocks. Equally, changes in the calculation of features could take days. Hence, optimizations and verification of the outcome is a tedious process.

Furthermore, there are some potentially problematic assumptions in implementation. The highest risk is the assumption that the purchasing/selling price can be executed on average very close to the actual opening or closing price. The author did not have any intraday trading data at his disposal, hence does not know.

In general, data is an issue. We know that there is much more to the stock development than the stock price development, two fundamentals, short selling and sentiment. For the approach to improve, additional data sources would need to be implemented.

Finally, this document should stay in the range of 15 pages. This is too short to address all the details of the implementation, the findings and the challenges along the way. The author has therefore focused on the high-level description.