Machine Learning Capstone Project

Klemen Čas

Date: 2016-11-03  
Github: https://github.com/KlemenCas/ML\_Final   
Readme to the technical implementation: https://github.com/KlemenCas/ML\_Final/blob/master/README.md

Abstract

This is the final project to the Machine Learning Nanodegree. The area of focus is Investment and Trading, narrowed to trading with SP500 stocks. 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 this input 16 features and 8 labels have been calculated. For optimization, the processed data has been stored locally.

Skit-learn Support Vector Machines, Decision Trees, Random Forrest, kNeighbors, Adaptive Boosting and Naïve Bayes have been trained, on individual stock level. Where grid search was available, it has been used for parameter optimization. Equally, Principal Component Analysis was used to reduce the dimensionality of the input, and clustering for reduction of label’s complexity, for forecasting of price changes. The accuracy has been logged locally, to be able to select the best performing model during the forecasting. Equally configured models have been saved locally, so that they don’t have to be trained on the fly.

The models do not forecast the trading action to be taken, they forecast the next state of the ticker – state being the labels. The trading action to be derived from the state has been implemented as Q-Learning.

The visualization of the implementation is two-fold. First, the simulation module simulates the development of a kind of index ETF; a portfolio that starts with the SP500 composition, sells according to the system recommendation, and when tries to realign to the index. As WIKI only provides today’s index composition, the implementation does not account for changes. This has been corrected in the benchmark too; it’s composition is being calculated daily based on the market capitalization of the included stocks.

Second, a small module has been implemented for individual stock prediction, with a very simple Tkinter interface.

The simulation chapter also shows the performance of the setup compared to the index benchmark.

Machine Learning Capstone Project

The report is divided into sections; Chapter I, Data Retrieval, describes the data retrieval and processing. Its outcome is a separate pandas DataFrame for each stock, stored locally, containing features and labels. Chapter II, 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.

Simulation, chapter III, describes the simulation of the stock and index values, whereby the stock portfolio trades base on the recommendations based on the second chapter.

Chapter IV then briefly describes the setup for individual recommendations, in which the user selects a ticker and the system provides a recommendation.

Finally, chapter V, briefly summarizes the results.

# 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 (Wikipedia S&P500, 2016). The composition is then stored in 3 dictionaries:

1. sp500\_ticker; with stock ticker as key and the industry as value
2. sp500\_composition; with industry as key and the list of included stocks as value
3. sp500\_index; with the industry as key and the quandl access key as value

This 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.

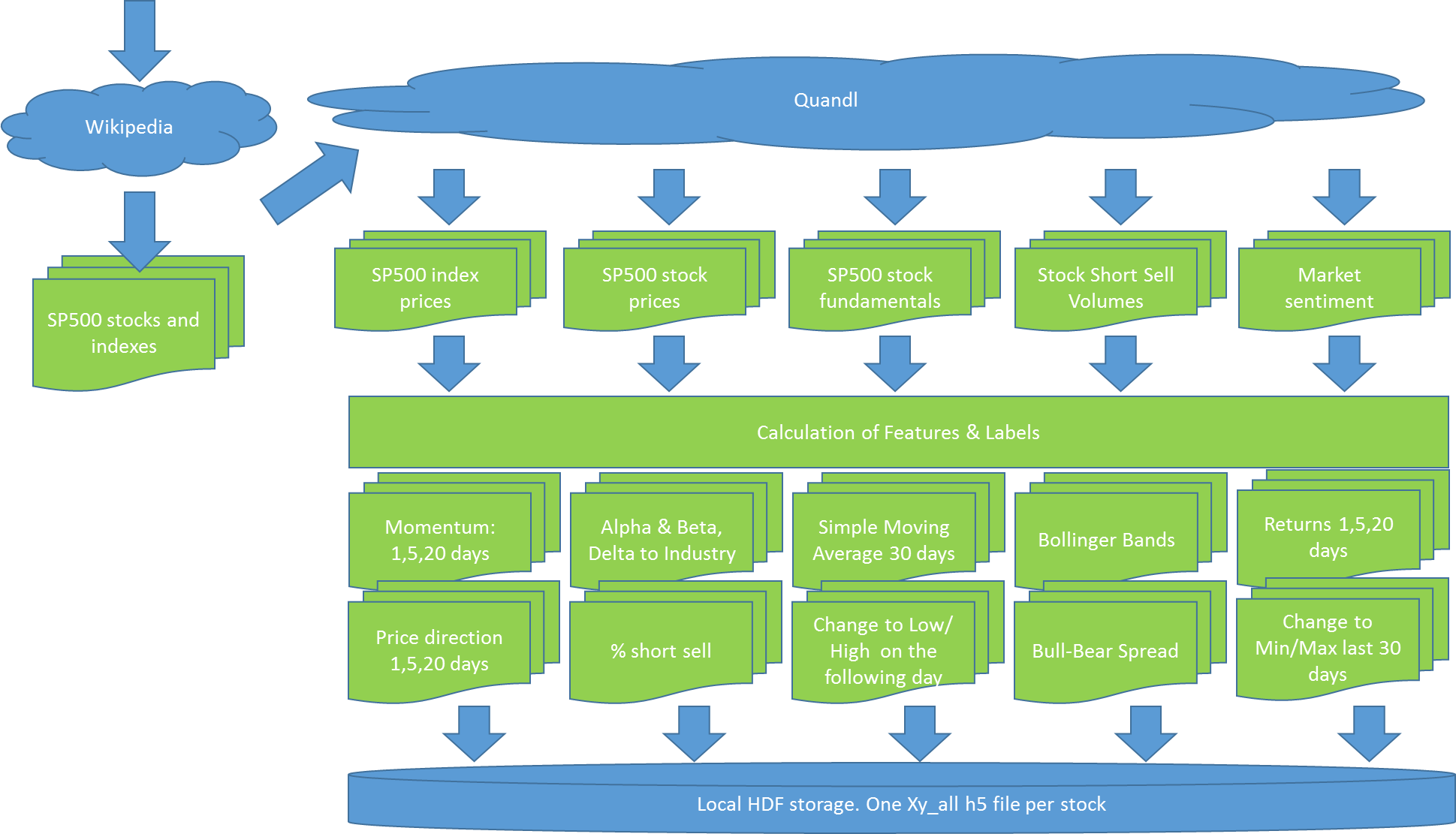


Figure :Data Retrieval Process

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

After the retrieval there is a large number of NaN values in the data sets. For example, the fundamentals data source delivers the value on the date when the indicator has been reported, but it does not back or forward propagate any values. Thus, all data sets are being first back propagated, and then the remaining NaNs are overwritten through forward propagation (Pandas DataFrame fillna, 2016). 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 features and labels. All calculations are done on adjusted prices, so that dividend payments and splits are taken into account. Features:

1. *Momentum* for 1,5 and 20 days; the % change in the Close price over the last 1,5 and 20 days,
2. *Alpha and Beta* of the stock, based on which further 3 indicators are being calculated. *Delta to Industry Development* for the past 1,2 and 5 days,
3. *Delta to Simple Moving Average* of the last 30 days,
4. *Bollinger Bands,*
5. *Volatility* last 30 days,
6. *Delta to Min and Max* of the last 30 days,
7. *Percentage of Short Sell,* NASDAQ and New York Stock Exchange,
8. *Bull-Bear Spread*

and labels:

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

In the next step the Momentum, SMA, Min, Max, Volatility, Bollinger Bands, Bull-Bear-Spread, Price Book Value, Earnings Per Share and Delta to Industry Returns are transformed to the range 0..1, by using the scikit MinMaxScaler. All these transformations are done on by stock, so that there is no cross dependency between stocks.

The output of the process is one hdf file by stock, named Xa\_all\_<Ticker> and stored locally.

# Training

The input for the training are the Xy\_all hdf files from the Data Retrieval phase. They contain a pandas data frame with all features and labels, one for every stock symbol. In the next step 6 machine learning algorithms are being trained, all from the scikit learn: Support Vector Machines, Random Forrest, Decision Trees, AdaBoost, kNeighbors and Naïve Bayes. Figure 2 displays the flow.

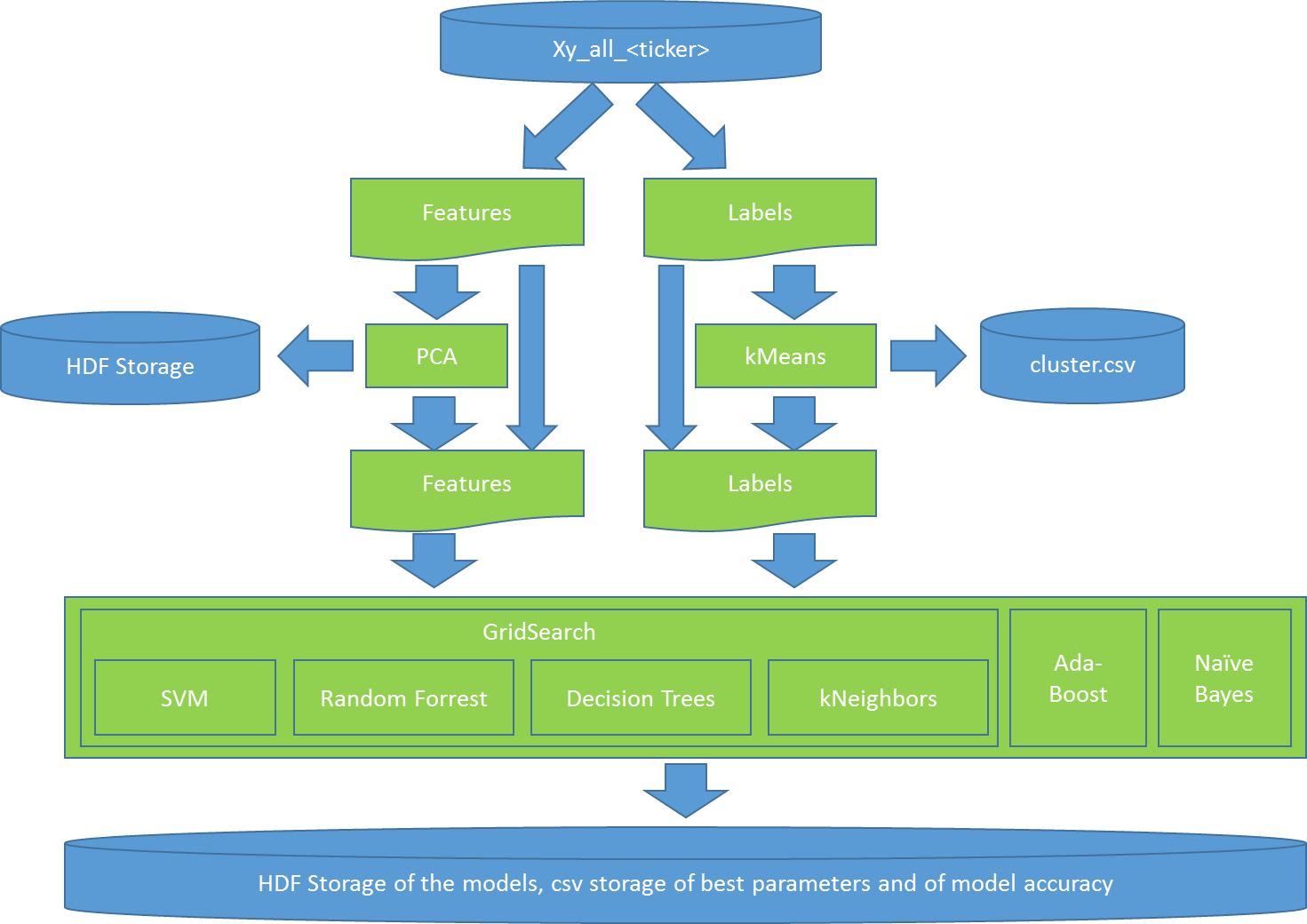


Figure :Process Flow for Training

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.

Similarly, 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 a specific trained model and the PCA object have been logged and stored as a pickle object (the models) or in a csv file. 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. One Grid Search for a particular model thereby counts as one training.

On the average dimension reduction delivered slightly higher accuracy, 73.5% versus 72.5% (Figure 3). 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

Figure : 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 4).



Figure : 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 5).



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

# Simulation

The simulation module picks a start date 5 years in the past, assembles an index portfolio based on the index composition as described in the data retrieval chapter, and an identical simulation portfolio. From there onwards there is a daily update:

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 between the labels and the actual actions (buy, sell, hold).

Figure 6 shows the flow in more detail; The simulation is being performed by index, and by individual stock. The overall performance is then 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 is being used to select, load and execute the best model. For simulation purposes only the features part of the Xy\_all HDF storage is being used.

The model predicts the state of the stock symbol, which is the labels from above; 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.

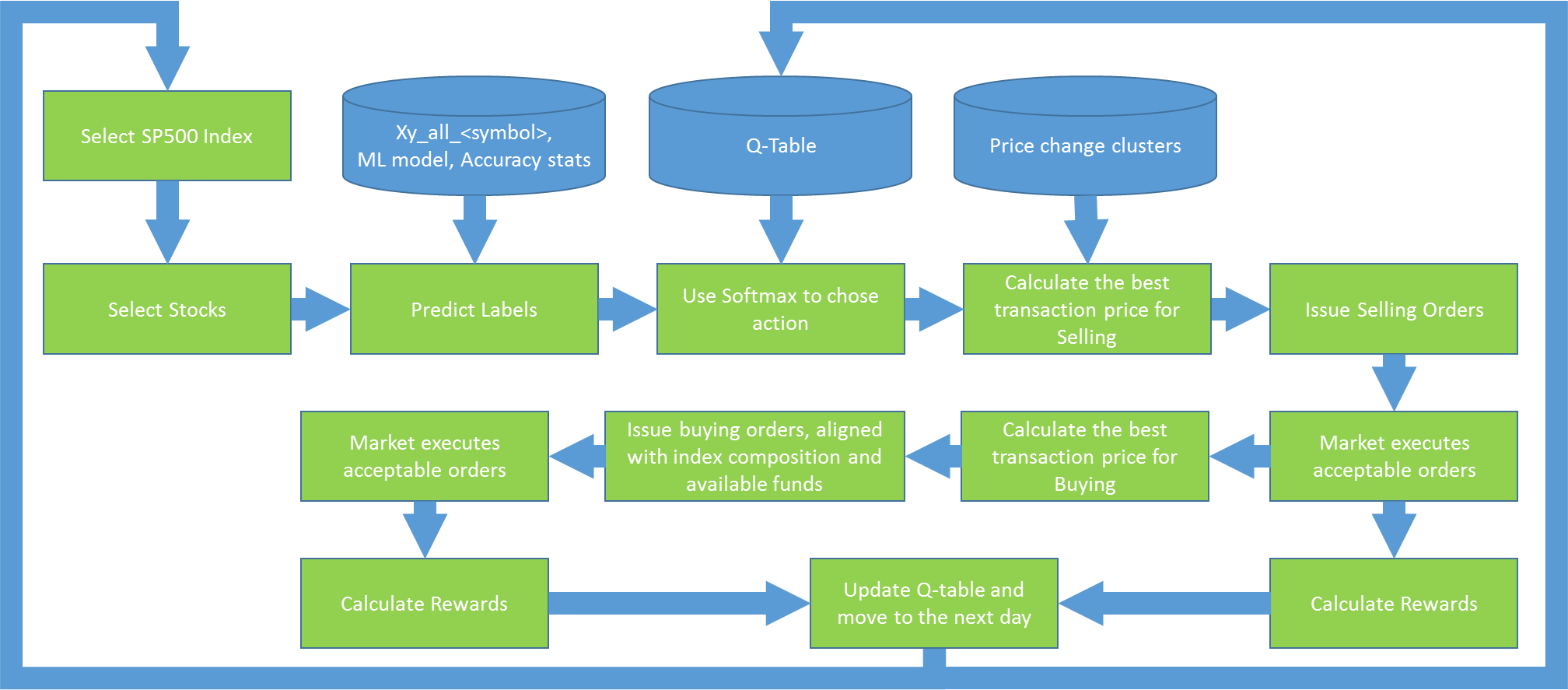


Figure 6: Simulation Flow

With the state, softmax is being used to determine the recommended action, with which then the order price is being calculated, based on the prices 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.

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 calculated. The aim is to have a portfolio that is close to the index. Therefore 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 there can be cash left. That is the cash that was allocated to buying orders that have not been executed due to a too low order price.

The rewards 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=.2 and gamma=.4 the q-table is then updated.[[1]](#footnote-1)

# 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…). This is a time-consuming step that is being executed before the actual user forecasts are being ran. Once done, the user interface will provide stock price recommendations for the day.

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. 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.
2. The system loads the feature data and the according to the training statistics the 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.

References

Pandas DataFrame fillna. (2016, 10 30). *pandas.DataFrame.fillna*. Retrieved 10 30, 2016, from Pandas: http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.fillna.html

scikit AdaBoost. (2016, 10 30). *sklearn.ensemble.AdaBoostClassifier*. Retrieved 10 30, 2016, from scikit Learn: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

scikit Decision Trees. (2016, 10 30). *Decision Trees*. Retrieved 10 30, 2016, from scikit Learn: http://scikit-learn.org/stable/modules/tree.html#tree-classification

scikit kN. (2016, 10 30). *sklearn.neighbors.KNeighborsClassifier*. Retrieved 10 30, 2016, from scikit Learn: http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

scikit MinMaxScaler. (2016, 10 30). *sklearn.preprocessing.MinMaxScaler*. Retrieved 10 30, 2016, from scikit Learn: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

scikit Naive Bayes. (2016, 10 30). *sklearn.naive\_bayes.GaussianNB*. Retrieved 10 30, 2016, from scikit Learn: http://scikit-learn.org/stable/modules/generated/sklearn.naive\_bayes.GaussianNB.html

scikit Ranfom Forrest. (2016, 10 30). *sklearn.ensemble.RandomForestClassifier*. Retrieved 10 30, 2016, from scikit Learn: http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

scikit SVC. (2016, 10 30). *sklearn.svm.SVC*. Retrieved 10 30, 2016, from scikit Learn: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

Wikipedia S&P500. (2016, 10 30). *List of S&P 500 companies*. Retrieved 10 30, 2016, from Wikipedia: https://en.wikipedia.org/wiki/List\_of\_S%26P\_500\_companies

1. The gamma and alpha values are arbitrary; it’s simply the two values that lead to good performance in the Smartcab project. To be more accurate, different values would have to be simulated. [↑](#footnote-ref-1)