Udacity Investment and Trading Capstone Project

- Stefan Dulman -

Table of Contents

Project overview

Investment and trading are activities that generate lots of interest in our society. In this project we will explore the application of machine learning to financial transactions, in order to help investors and portfolio managers to take informed decisions. We will characterize the chosen stock/portfolio with a variety of metrics and make predictions with respect to their evolution. The effects of the predictions will be measured and results analyzed.

A plethora of data exists already, as financial transactions on the stock market are well regulated and documented. We will make use of data extracted from Yahoo Finance, which allows very easy access to transaction records. Our analysis will be based on data recorded in the last six years for a few well-known companies (Google, Yahoo, Microsoft, Amazon, Apple and SPDR S&P 500 ETF). We provide an interface via which the user can download data for other companies and have access to all results of this project on a portfolio of her own choosing.

Making buy/sells decisions is difficult and, to the best of my understanding, will always have a chance variable attached to it. Nevertheless, the mechanism exemplified in this project will help understand and manage the risk level for investments. These tools help visualize and understand the tradeoffs between risk and expected gains.

Problem statement

In this project we focus on exploring and understanding the relationship between taking a risk versus expected gains. The main problem comes down to a simple question: will a particular stock go up or down in the future? We will try to answer this question by creating a few features that expose trends in the data and use machine learning regression to predict the next value of a particular stock.

The problem can be generalized to several stocks forming a portfolio. We will make use of notions from the modern portfolio theory developed by Harry Markowitz. Based on the predicted evolution of stocks, we will suggest how the portfolio manager should allocate his resources to maximize gain while minimizing risk.

Analysis

Data exploration

We downloaded the data for the six stocks from Yahoo Finance for the period 01-01-2011 until 31-12-2016 and performed simple statistics and visualizations on it. The time evolution of the stocks is presented in Figure 1. We notice that the evolution of stocks takes place at different scales, with Amazon and Google stocks dominating the others price-wise.

In order to be able to be able to properly assess the relative values of stock we need to normalize these values with respect to the first data entry. The normalized evolution of stocks is presented in Figure 2. The relative comparison already shows a slightly different story. The amazon stock dominates all the others in terms of performance. Its value has increased four times in the considered time interval. Despite having a similar high price per share, the Google stock is much more similar in performance with the other stocks considered.

We present simple statistics of the data in Table 1. We notice that the data has no gaps for the considered interval. All the symbols have 1510 samples over the considered six years. The kurtosis values are quite different than the standard one of 3 for normal distributions. We expect the distributions to be asymmetric or even multimodal.

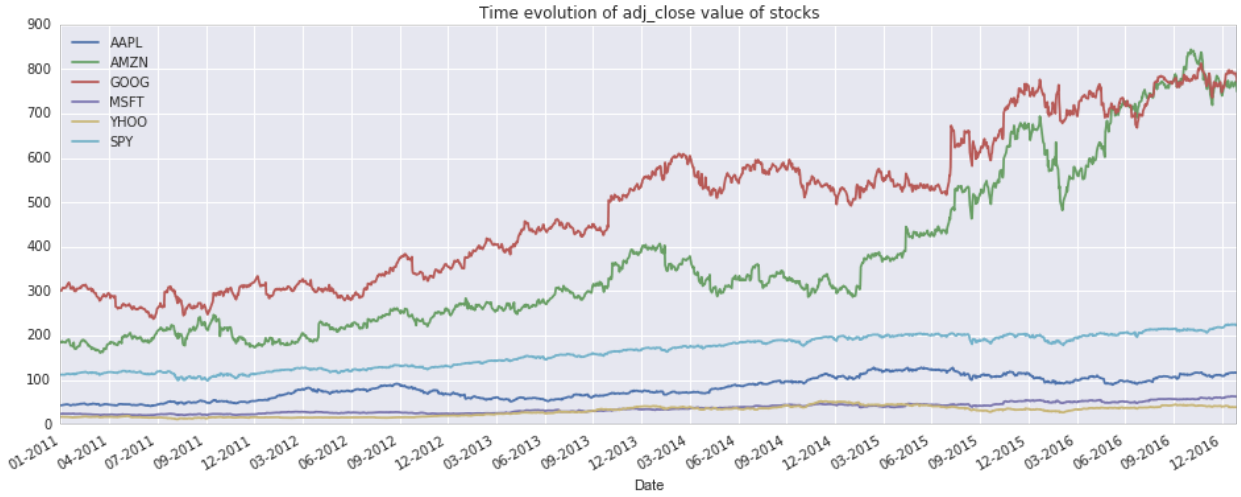

Figure 1 Time evolution of stocks

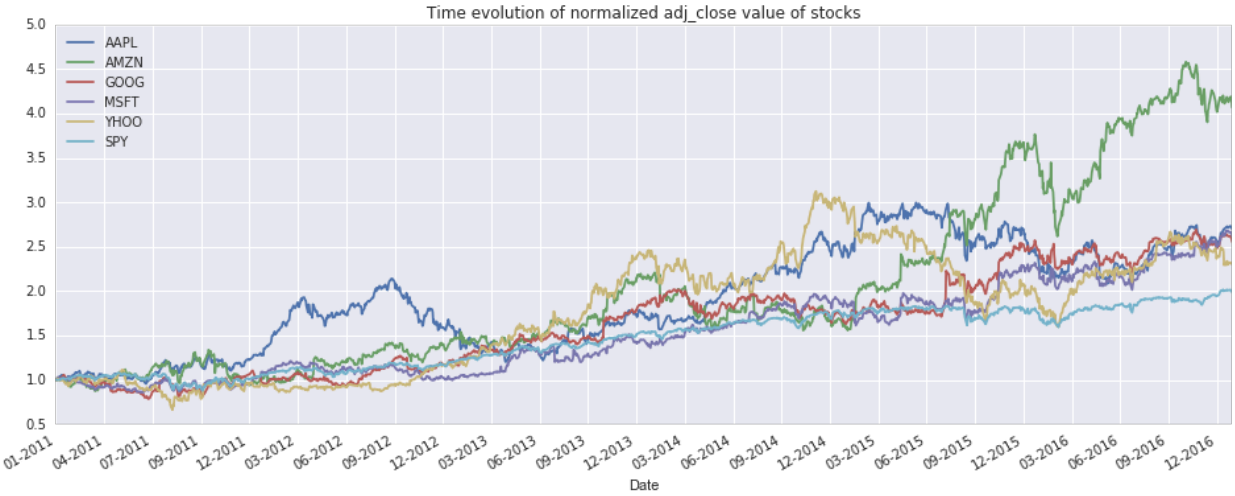

Figure 2 Time evolution of normalized stocks

Table 1 Statistics of the data

	AAPL	AMZN	GOOG	MSFT	YHOO	SPY
Count	1510	1510	1510	1510	1510	1510
Mean	81.90	371.07	492.60	36.03	29.24	162.84
Std	24.82	184.30	166.98	11.66	11.18	36.40
Min	40.85	160.97	237.20	20.23	11.09	98.12
25%	59.54	227.83	317.36	25.53	16.35	125.60
50%	78.77	310.04	518.34	34.09	32.13	168.69
75%	105.04	439.92	601.17	44.73	38.32	197.49
Max	127.97	844.36	813.11	63.24	52.37	225.44
Kurtosis	-1.24	-0.11	-1.20	-1.03	-1.38	-1.48

This last hypothesis is confirmed by plotting the histograms of two of the distributions. As shown in Figure 3, the Amazon stock distribution is heavily skewed to the left. Figure 4 shows that the distribution corresponding to the Yahoo stock is non-standard, having two modes, one very sharp and one quite flat.

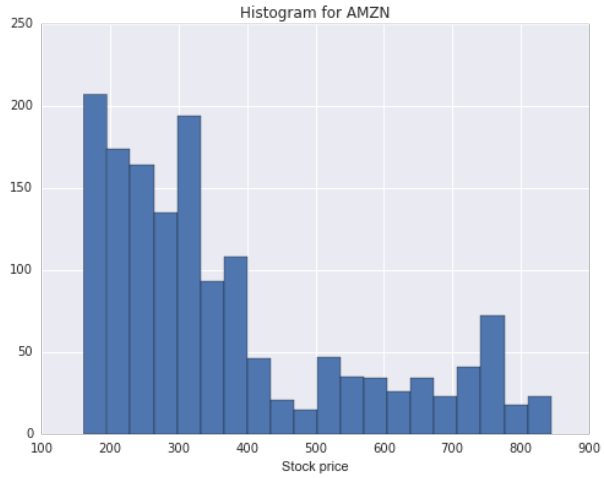

Figure 3 Histogram for the Amazon stock

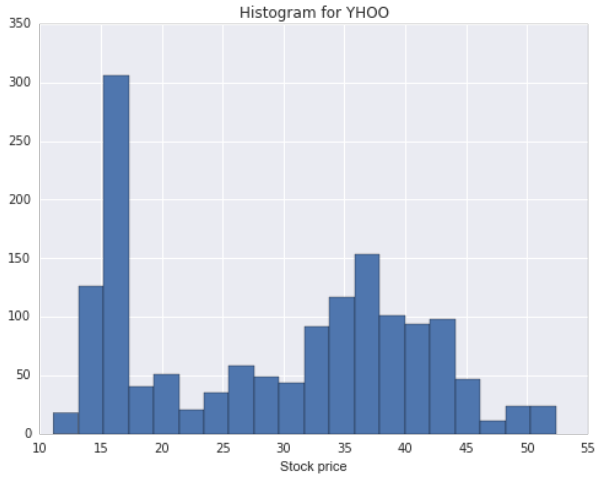

Figure 4 Histogram for the Yahoo stock

Algorithms

We will make use of two regression techniques in this report (random forests and support vector regressions), briefly described below.

Support vector regressor (SVR)

SVR regression is a variation of the concepts of support vector machines (SVM). SVM is a technique developed primarily for clustering. For example, suppose that the task at hand is to classify two sets of points as the white and black circles shown in Figure 5. The way in which SVM works is determining the points that bound the two clusters and then create a curve (red line) separating them, such that this curve is as far away from the margins of the two clusters as possible. A good solution for SVM classification will be a curve that allows ample amounts of space around it, as shown in the figure below. SVM is not limited to the 2D space – it can find a hyper plane separating data in any number of dimensions.

When we move from the discrete case presented above to the continuous one (where each point is associated with a continuous target value), we obtain the regression version of SVM, called SVR. The algorithm works similarly, only that the optimization procedure includes also the target value in its mathematical formula.

SVM is an established technique for which a lot of theory has been developed. The most notable result is the employment of various *kernels*. A kernel is a function which replaces the dot product used in the original algorithm with virtually any kind of function. Some of these functions can be highly nonlinear in the original space – hence SVM provides a lot of flexibility. The flexibility of SVMs come at a cost – the need to choose a suitable kernel for each problem. This is where we will be employing grid search methods to find the best function for our problem from the few most used ones.

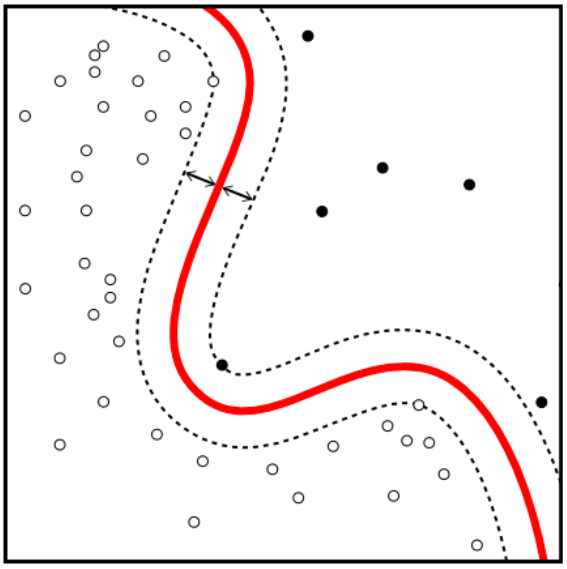

Figure 5 SVM classification (taken from [1])

Random forest regressor (RFR)

To understand random forests, we need to cover the concept of decision trees first. Assume some data needs to be classified in one of two classes, as shown in Figure 6. We could create a set of "if…then" rules with respect to the two axis (x1 and x2) and basically split the space into rectangles as shown below. While preferred for their simplicity (drawing the decision tree offers a straight-forward explanation of what it does), decision trees are quite sensitive to even small amounts of noise. Noisy data can lead to the wrong constants used in the comparisons in the tree, leading to wrong results.

A solution to this problem is the generalization of the decision tree into random forest [2]. Instead of using a single decision tree, we can train several (say 100). For each data point that needs classification, we will run each decision tree and do majority voting on the results. Random forests are common mechanisms in machine learning and can cope with noisy data pretty well. Also, they are easy to train, the main disadvantage being the time taken – we need to train each tree individually. As in the previous case, the random forest regressor is the generalization of the classifier to a continuous case of the target variable.

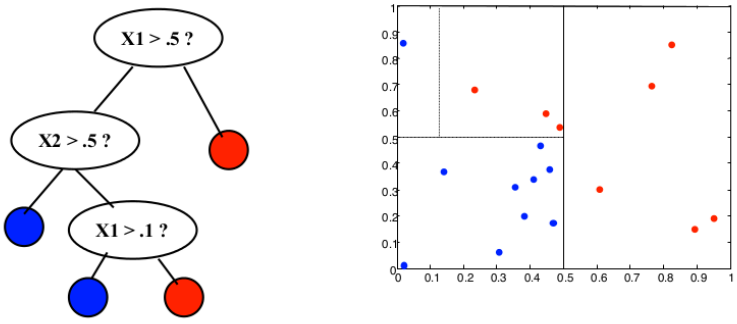

Figure 6 Decision tree (taken from [3])

Single stock management

As previously mentioned, we added a number of variables to the simple data set. We begin this section by detailing these additions:

Bollinger bands [4]: we are using the contrarian version of Bollinger bands [5]. We chose a band of 2 sigma-s around the mean of 12 weeks. Regarding the generation of buy, sell signals: if close price exceeds the upper band, we have a sell signal and if close price falls below the lower band, we have a buy price.

Figure 7 shows the Bollinger bands and the buy/sell points for the Google symbol. First of all, we notice that we can have successions of buy decisions or successions of sell decisions. We filtered the decision duplicates (retained only the first value) and obtained the results presented in Figure 8. This seems to work for the buy signals, but outputs bad results on the sell signals. In other words, delaying the sell decision to the last value in a succession will provide significant gains. In line with our estimator (which can only look ahead for a limited number of steps in the future), we hold to this kind of filtering and apply it when needed to all metrics buy/sell decisions.

Figure 7 Bollinger Bands for Google stock – unfiltered buy/sell decisions

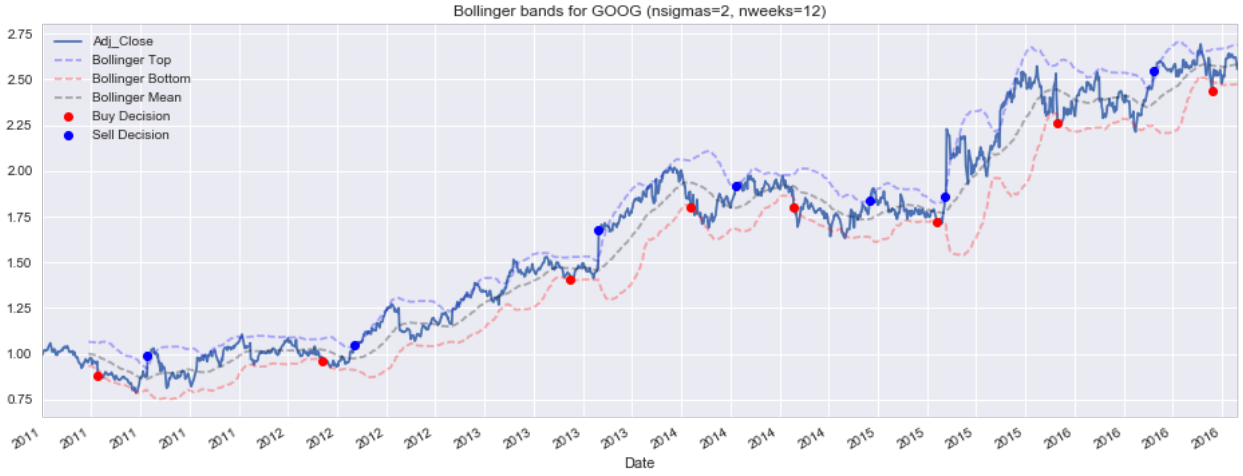

Figure 8 Bollinger Bands for Google stock – filtered buy/sell decisions

Dual moving average crossover [6] – compares two moving averages. Again, we choose the contrarian approach: selling (buying) immediately after the short-term moving average exceeds (falls below) the long-term moving average by at least 1% (should hold in highly volatile stock markets).

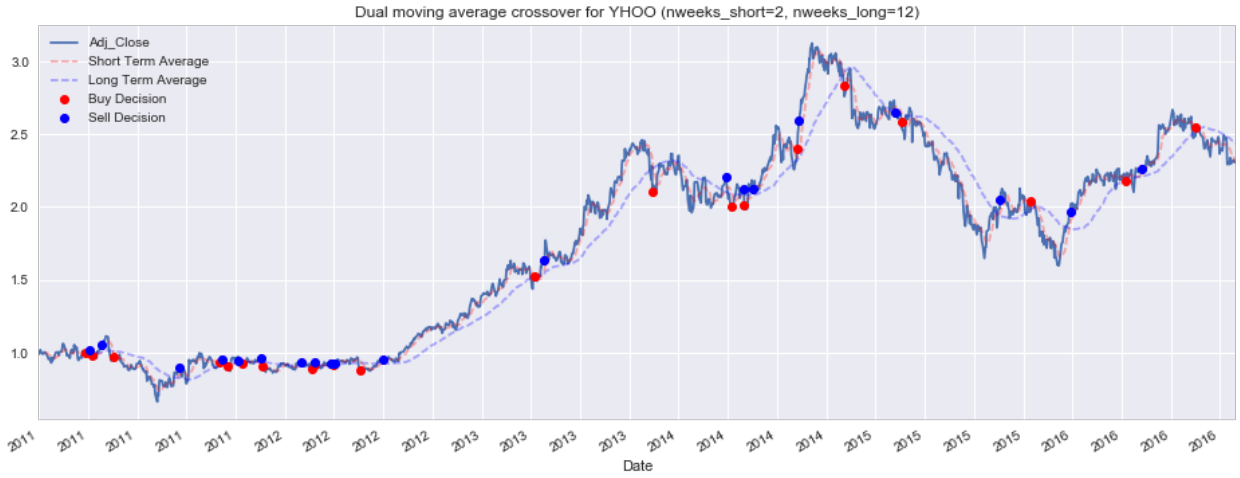

Figure 9 Dual moving average crossover for Yahoo

Contrarian channel trading - a buy (sell) signal is generated for the next day trading if the close price on any given day falls below (exceeds) the lowest (highest) close price of the past n weeks (4 in our case) by at least 1%. Figure 10 presents this metric on the Amazon stock.

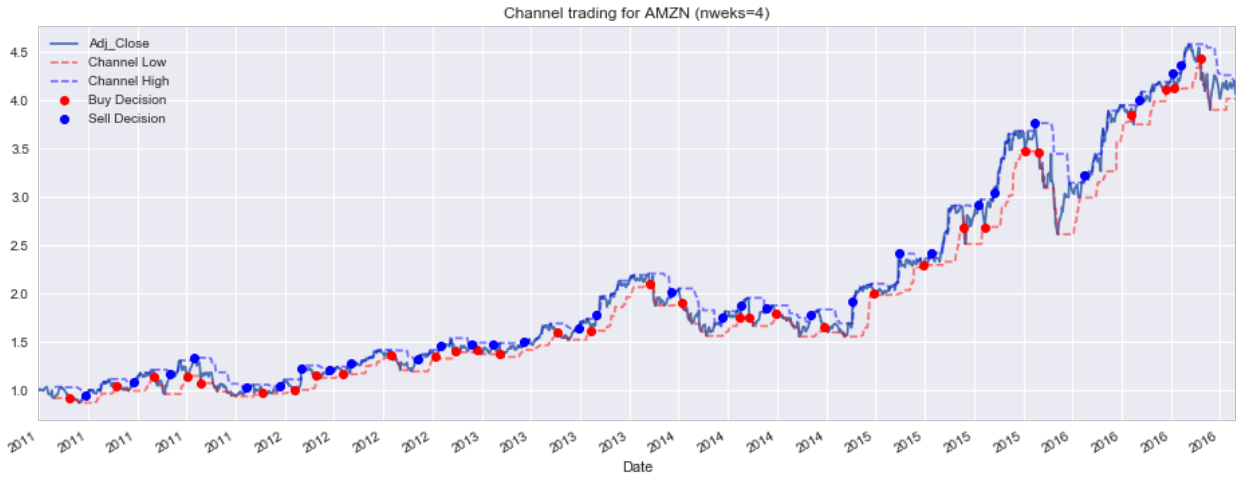

Figure 10 Chanel trading for Amazon

Portfolio management theory

Efficient frontier for no risk-free assets

Conform to modern portfolio theory [7], we can compute a minimum amount of risk as a function of a given expected return. This function takes graphically the form of a hyperbola, known also as Markowitz bullet. Assume that ω_i are the weights associated with the assets (their sum is 1), Σ the covariance matrix for the returns of the assets in portfolio, R the vector of expected returns [8] and q a positive variable. We will choose the vector ω which minimizes the expression:

$$\omega^T \Sigma \omega - q R^T \omega \quad (1)$$

for a given value of q (risk tolerance). The first term represents the variance of the portfolio return, while the second represents the expected return on the portfolio. If we use the notation $\mu = R^T \omega$ then we can minimize $\omega^T \Sigma \omega$ (risk) for a given value of μ (given return). This coincides with the definition of the the efficient frontier given above.

We perform this computation using an optimizer which searches for the minimum value of the risk under the constraints that the given return equals μ and the weights add to 1. We also bound the weights to the (0,1) interval.

Assuming $\rho_{i,j}$ being the correlation between two assets, the covariance between the same assets is $\sigma_{i,j} = \sigma_i \sigma_j \rho_{i,j}$ where σ_i is the standard deviation of the returns of asset i. In matrix form, if σ is the vector $[\sigma_1, \sigma_2, \ldots \sigma_n]$, ρ is the correlation matrix, and diag() is a function returning a square diagonal matrix with the covariance vector on its diagonal:

$$\text{Covariance matrix} = \text{diag}(\sigma)\, \rho\, \text{diag}(\sigma) \quad (2)$$

Figure 11 presents an example of efficient frontier [9] for our portfolio computed with the supplied code.

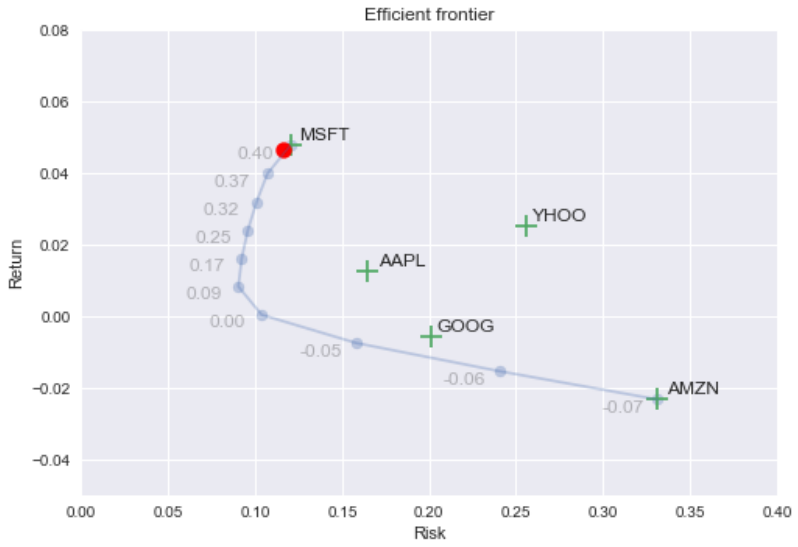

Figure 11 Efficient frontier for portfolio on 11 Nov 2015

Maximum Sharpe ratio portfolio

Figure 11 also shows various values of the Sharpe ratio for portfolios along the efficient frontier. We are interested in determining the portfolio which maximizes the ratio return versus risk, or the maximum Sharpe portfolio. This is indicated with a red dot and is situated at the point where the capital allocation line is tangent to the efficient frontier. In Figure 11 the allocations are 0 for all, except 0.943 for MSFT and 0.056 for YHOO.

We will perform an optimization similar to the one described in the previous section, only this time we will maximize the ratio:

$$\text{Sharpe Ratio} = R^T\omega \,/\, \text{sqrt}(\omega^T\Sigma\omega) \tag{3}$$

subject to the weights ω_i summing to 1 and taking values in the interval (0, 1).

Methodology

Data preprocessing

We used a package called yahoo-finance to retrieve the data for this project. It has a straight-forwards interface returning data from Yahoo Finance based on a query that includes the name of the symbol and a range of dates. From the returned data we only held the “Adj_Close” and “Date” columns. As basic pre-processing we converted the “Adj_Close” value from string to float and the “Date” from string to a date object. We replaced the index of the data frames with the “Date” column data. Also, we sorted the index in ascending order (data is retrieved in descending order).

In order to make processing and interpretation easier, we normalized the data by dividing all stock values by the first value in the dataset. This way, all stocks start at the value 1 and their evolution is easier to compare to each-others.

No additional preprocessing was needed as the data for the selected stocks was in good format. No data was corrupt or missing.

Our estimators use the following rolling window variables, which were added to the dataset:

- Bollinger lower and upper band (together with the mean and standard deviation values computed on the moving window). The default window size was 12 weeks.
- Dual cross-over means – the default window size for the slowly moving mean was 16 weeks, while the default window size for the fast moving mean was 2 weeks
- Channel-trading mean – the default window size was set to 4 weeks
- Daily return – defined as the gain/loss made every day with respect to the previous day

Each day in our data set gets a value for all variables expressed above. As the estimator needs a rolling window of data, we create a data frame where for each day, we assemble all the above variables in a window of 20 previous weeks. Thus, for every prediction, data from the past twenty weeks is taken into account, leading to 180 features in the feature space.

Estimator implementation

The core functionality on which this project relies is the ability of estimating the close price for a stock in the future. There are a lot of parameters to consider in order to perform a good prediction. Some of them are linked to the data (relevant features, derived variables such as daily returns and moving averages), some to the history aspects (size of time windows) while some relate to the estimators being used (kind of estimators, parameters of estimators). While searching all the possible combinations is infeasible we need to take informed decisions about fixing constant values for most of them and limiting the search to as few as possible.

The decisions we took about fixing parameters are:

- Data related: we will perform training and testing using two data sets. One is based solely on ‘Adj_Close’ values in the past (we call this the ‘simple dataset’). The second includes Bollinger bounds, dual cross-over and channel trading means as well as daily returns. We will call this data set (‘full dataset’).

- Time related: for the full dataset, we set the window variables such that there are no overlapping parameters. Bollinger bounds are computed using a 12-week interval, dual cross-over means computed using 16 and 2-week intervals, while channel return uses 4-week intervals. Deciding on the amount of data in the past to consider for the overall simulation was more difficult. We settled for a four-week interval based on the graph below (we trained an estimator for increasing values for the history window for all selected stocks – any value from one to four weeks seems to be a good trade-off for all of them).
- Estimator related: we tried several estimators and decided to include in the report Support Vector Regression (SVR) and Random Forest Regressor (RFR). Details about their parameters are provided later in the report.

Training estimators requires careful design of the training and test data sets. For this project, we are dealing with time series, which involves additional complexity. Samples cannot be used at random as trends and time correlations need to be taken into account. Also, testing data should always be in the future with respect to the training data.

Simple cross validation cannot be employed as it would use testing data which is in the past with respect to the training data. Hence, we employed an adapted version from sklearn library, which works with time series, called TimeSeriesSplit(). It works as follows: first it splits the data in several consecutive sets (s0, s1, etc.). The train set is an increased at each step, while the test set is considered to be the immediate following set. Notice that the train set includes all previous data:

Table 2 Partitioning with TimeSeriesSplit()

Partition number	Train set	Test set
1	[s0]	[s1]
2	[s0, s1]	[s2]
3	[s0, s1, s2]	[s3]
4	[s0, s1, s2, s3]	[s4]
…	…	…

In order to search for the best predictor, we used a technique called grid search. It considers sets of values for the parameters and it searches through all their combination. For the two classes of estimators we considered, these were the parameters used:

- SVR: 'C': [0.5, 0.75, 1, 1.5, 2, 5, 10], 'kernel': ['linear', 'rbf', 'sigmoid']
- RFR: 'n_estimators': [50, 100, 200], 'min_samples_split': [2,3,5,10], 'min_samples_leaf': [2,5, 10]

Results

Model evaluation and validation

We implemented the grid search procedure described in the previous section. As an example, some of the best parameters found for different time windows and different stocks are:

Table 3 Parameters for SVR and RVR via grid search

Time window size	Stock name	SVR	RVR
1	AAPL	'kernel': 'linear' 'C': 10	'min_samples_split': 10, 'n_estimators': 200, 'min_samples_leaf': 5
2	AMZN	'kernel': 'linear' 'C': 0.5	'min_samples_split': 3, 'n_estimators': 200, 'min_samples_leaf': 5
3	GOOG	'kernel': 'linear' 'C': 5	'min_samples_split': 2, 'n_estimators': 200, 'min_samples_leaf': 5
4	YHOO	'kernel': 'linear' 'C': 1.5	'min_samples_split': 2, 'n_estimators': 200, 'min_samples_leaf': 5

The overall performance of the considered estimators over the portfolio is shown in the following four figures. Figure 12 and Figure 13 show the performance of SVR. Training the estimator on simple data leads to much better results, leading to the conclusion that the added metrics have a high correlation and actually leading the estimator away from the desired performance.

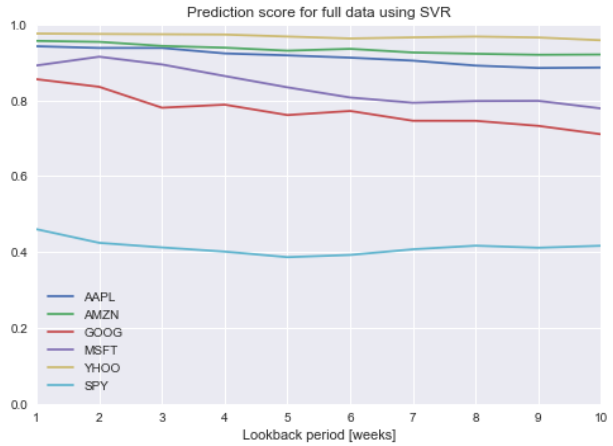

Figure 12 Prediction score for full data using SVR

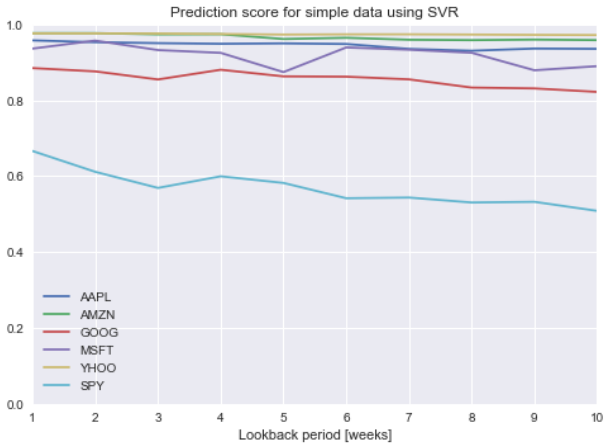

Figure 13 Prediction score for simple data using SVR

The Random forest regressor performs substantially worst than SVR. As shown in Figure 14 and Figure 15, most of the stocks don't even have positive prediction scores (we chose to show only the positive values on the graph). There are several reasons for this. The main one is that the predictions are just a constant for ascending data that has not been seen before (common trend in the current portfolio).

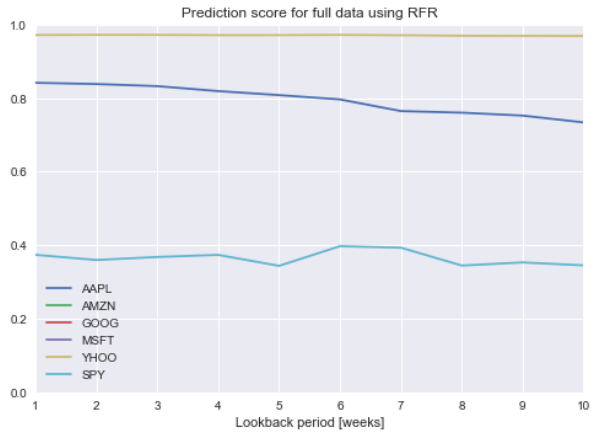

Figure 14 Prediction score for full data using RFR

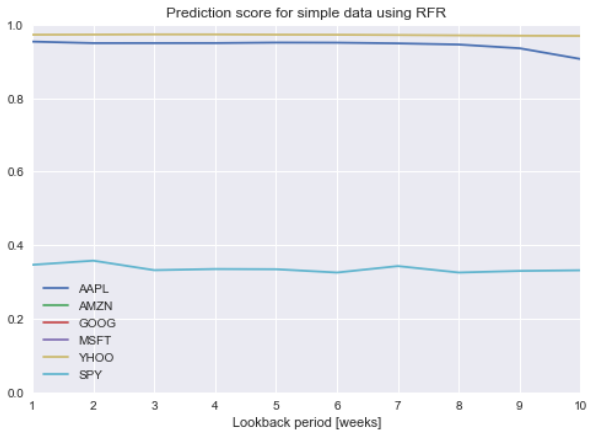

Figure 15 Prediction score for simple data using RFR

Overall, SVR performs much better than random forests estimators. In the following we will only use it for performing predictions related to the stock evolutions. SPY stock is the hardest to predict. This is because it shows a constant increasing trend for the selecting period. The chosen predictors do not handle well constant trends in the data (the recommendation for SVR is that the data is actually normalized). Such trends could be captured by using, for example, a neural network.

Single stock management results

Using the SVR predictor we can define several heuristics for generating sell/buy signals and trying to trade automatically. The results in this section are based on back testing, meaning that we used the historical data we had as input for the automatic trading simulation. We consider the data normalized to the first entry in all the remainder of this report.

First, we consider the ideal situation in which all data is known before hand. This can be used as a benchmark, as it outputs the maximum profit feasible on the stock. Assuming that one buys one share when the buy signal is issued (at each local minimum in the stock) and sells the share when the sell signal is issued (at each local maximum in the stock), the following numbers for total sells and buys are obtained:

Table 4 Local extremes based buys/sells results

Stock	Total sells	Total buys	Predicted sells	Predicted buys	Real profit	Predicted profit
AAPL	533.44	519.89	285.15	285.54	13.55	-0.39
AMZN	491.90	482.08	198.53	196.37	9.82	2.16
GOOG	424.82	416.72	95.02	93.73	8.10	1.28
MSFT	393.02	383.04	89.16	89.72	9.98	-0.56
YHOO	469.25	458.96	239.37	238.29	10.29	1.08
SPY	394.12	387.44	25.30	24.10	6.67	1.19

As expected, the predicted values are much smaller than the real ones, as a direct consequence of the predictor limited precision. The errors for buy/sell signals vary with each stock.

Finally, we took the sell/buy decisions based on Bollinger bands and Channel trading approaches (described above). Back testing the predictors led to the following results:

Table 5 Profits for Bollinger bands and Channel trading approaches

Method	AAPL	AMZN	GOOG	MSFT	YHOO	SPY
Bollinger bands	1.38	1.10	-0.97	1.38	-2.29	0.99
Channel trading	1.68	1.89	2.41	1.99	0.22	0.42

It looks that the channel trading approach performs the best out of the three tested methods. It output profit for each of the stocks involved.

Portfolio management results

We integrated the previous piece of theory into our project and ran back testing of the automatic portfolio management. We start with an equal allocation for all stocks (5 stocks – 0.2 allocation share for each). We predict the next day price evolutions and change the ratios to match the new Sharpe portfolio. At the same time, we record how much expensive the transaction was based on the amount of change in ratios and real price variation.

As benchmark we chose the SPY symbol, representing the SPDR S&P 500 ETF stock. It appears to be the most popular mutual fund today, with good performance and a huge amount of daily transactions.

The results are shown in Figure 16. As we notice, our estimator failures are influencing the portfolio management quite a lot. At times, our algorithm is approaching SPY but most often than not it fails.

Figure 16 Automatic portfolio management vs SPY

Justification

Before judging the performance of the estimators we need to take a look at the given data. The time aspect makes it quite different than the data traditionally used in a basic classification or regression problem.

Consider, for example, a stock with a rising trend for the considered period, such as Amazon in Figure 2. In reality, the future evolution of the stock is not determined by the past evolution, thus what looks as a trend line might be a simple "bump" when plotting the data for the next six years. Removing the trend line algorithmically would be just an artifact that will not be applicable to any other stock and despite providing (maybe) an improved result for a few predictions, will fail completely when looking at a larger time range.

Once we agree that the trend line should not be adjusted, then the estimator is left with the problem that it should generalize on data outside of the domain it was trained with. Consider again Amazon stock, for example the period June 2012 – December 2012 in Figure 2. This data is at the end of the time interval considered so it can only be used for testing. As visually obvious, this data is significantly different than the other period for the stock (data has a high plateau, not previously seen). An estimator like random forest will predict a constant value for this whole period, as it did not see anything similar before. SVR performs slightly better, but it suffers from the trend issue described above.

We tried to alleviate the problem by using derived variables in the training set which remove the trend (such as daily returns). Training the estimators on a data set formed only by such variables and normalized variables with respect to the first data entry for the whole time window considered led to worst results. This is because, the data transformation performed led to significant loss of information (the trends in the data are basically the only mechanism on which we can extrapolate).

Given these considerations, I appreciate that SVR performs quite well. As shown in Figure 13, it's score is larger than 0.9 for most of the stocks on simple data. This leads to a quite good correlation of predicted data (see for example Figure 17 in the Conclusion section). The difficulty of the stock prediction

problem lies in the fact that a good correlation is not enough – we need to be able to estimate the daily return in order to obtain profit in the automatic trading process.

Nevertheless, using an adapted technique which combines estimation with trends decisions, we are able to obtain profit for the given set of stocks – see the channel trading results in Table 5. The results give us confidence regarding the generalization of our solution, as profits are positive for *all* considered stocks.

Conclusion

Free-form visualization

An example of SVR prediction with simple data for the GOOG symbol on the test data is presented in Figure 17. We used a look-back period of 4 weeks. The predicted values follow the real values quite well (the overall score on the testing set is 0.88, while the correlation coefficient between the real and predicted series is 0.95).

A certain bias is noticeable – the predicted values are in general above the real data. This varies across the stocks in our considered portfolio and, as shown below, will lead to errors in the automatically generated Buy and Sell signals.

Figure 17 Prediction on the test data using SVR (normalized stock value as a function of time)

Despite the fact that historical data about stocks contains only limited amount of information, Figure 17 shows that, at least at the trend level, we are able to give a good prediction. This gives a good starting point for further exploration for decisions taken based on combined metrics such as channel trading, as explained in the previous section.

The second visualization we address in this section refers to the amount of correct buy/sell decisions based on the predicted data. Figure 18 presents the percentage of correct sell decisions versus the percentage of correct buy decisions for each stock. The decisions buy/sell were based on estimating if next day closing price will result in a local minimum/maximum on the graph.

The results are surprising. As we used different estimators, optimized for each stock, we expected less spread of these data points. The consistent bias the estimators output for each stock are part of the explanation for this phenomenon.

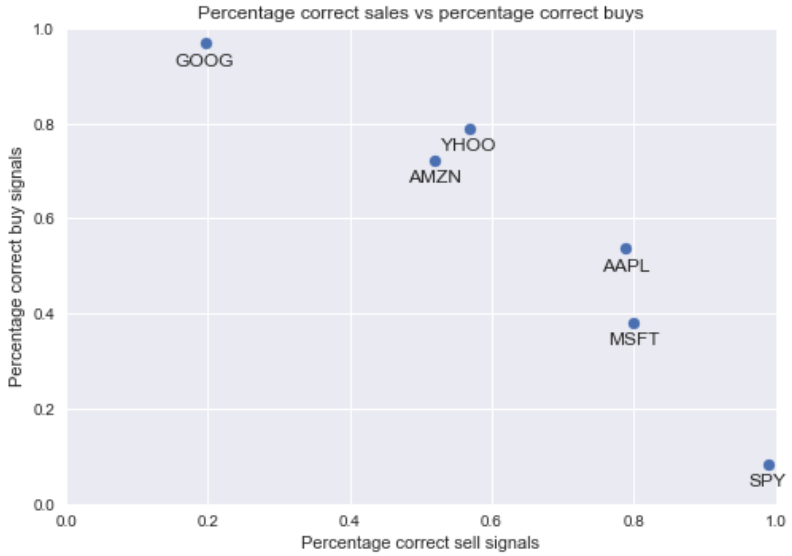

Figure 18 Percentage correct sells signals versus percentage correct buys signals

The figure shows that there is a trade-off between the accuracy for the buy and sell signals, with Google and SPY being at the extremes, while the other stocks fall somewhere in the middle. Based on this figure we can propose additional processing, or a different one for the stocks that fall in the extremes.

Reflections and improvement

To summarize this project: we started with the problem of single stock prediction based on already existing historical transactions data. We preprocessed the data and proposed several estimators for it. After evaluating their performances, we decided to use SVR and proposed an automatic grid search procedure which outputs the best parameters for a given stock. The last part of the problem was to generalize the results to the management of a portfolio. We followed the theory of modern portfolio management and proposed a procedure that computes the Sharpe portfolio. We back-tested this procedure on the historical data and compared its results with the SPY index.

I found this project very interesting actually. It was my first contact with the financial prediction community and it encouraged me to follow the “Machine learning for trading” course on Udacity. Having actually to code the predictors and understand how to interface them with the sell/buy decisions for single stocks and portfolio management made me aware of the many subtleties of trading tools.

It is obvious from the results that one should not run immediately and invest money based solely on such a tool. The first question that comes to mind is: could we make this tool better?

Based on the given implementation, the answer is yes, a few directions coming to mind:

- Combine several stock data together and train a single model that understands the concept of a stock, instead of models for each stock.
- When projecting the multidimensional training sets and test sets for a given stock on two-dimensions using PCA, the two sets overlap only partially. We could transform the data removing implied trends, like ascending one in ‘SPY’. We tried to normalize each window in time with

respect to the last value in the window. The train/test sets overlapped in 2D space, but the results were worst than the presented approach.

- Feature extraction plays an important role in machine learning and one can dedicate significantly more attention to it. Ideas like using an auto-encoder to refine the feature set or using an LSTM for prediction could be followed.
- The combination of a predictor that ignores trends with a metric that is trend based (such as channel prediction) leads to the best result. We have shown in Table 5 that we obtain a profit for each individual stock. Changing the portfolio management to adjust its weights only at these points in time (where we obtained guaranteed profit) is an idea for future work.

As a high-up view, I noticed that the stock closing values are not an ergodic process, meaning that traditional signal processing and statistical techniques will fall short on their results. The closing price contains only a small amount of information about the market, state and trends of the economy, social aspects, strategy and competition, etc. Hence, I do not expect major improvements in performance only based on the given data. I see this predictor tool useful in the sense of adding one more dimension to the time series, presenting the existing information in a slightly different refined way.

Bibliography

[1] Support_Vector_Machines. [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine.

[2] Random_forest. [Online]. Available: https://en.wikipedia.org/wiki/Random_forest.

[3] Random_Forest_Regressor_explained. [Online]. Available: https://www.quora.com/How-does-random-forest-work-for-regression-1.

[4] Bollinger_Bands. [Online]. Available: https://en.wikipedia.org/wiki/Bollinger_Bands.

[5] N. J. Balsara, G. Cheng and L. Zheng, "The chinese stock market: an examination of the random walk model and technical trading rules," *Quarterly journal of business and economics,* vol. 46, no. 2, pp. 43-63, 2007.

[6] Dual_Moving_Average_Crossover. [Online]. Available: https://faculty.fuqua.duke.edu/~charvey/Teaching/BA453_2002/CCAM/CCAM.htm.

[7] Modern_portfolio_theory. [Online]. Available: https://en.wikipedia.org/wiki/Modern_portfolio_theory .

[8] Expected_Return_Variance_And_Standard_Deviation_Of_A_Portfolio. [Online]. Available: http://www.investopedia.com/walkthrough/corporate-finance/4/return-risk/expected-return.aspx.

[9] Capital_allocation_line. [Online]. Available: https://en.wikipedia.org/wiki/Modern_portfolio_theory#Risk-free_asset_and_the_capital_allocation_line.