Machine Learning Engineer Nanodegree
Capstone Project

Jaehyoung Yoo 03/22/2017

I. Definition

Project Overview

In finance, portfolio means combination of investment assets such as stocks, bonds, and cash. When determining a proper asset allocation, one aims at maximizing the expected return while minimizing the risk[1]. Many investment firms, and hedge funds seek for optimal allocation which always comes with the trade-off between risk and return. On the other side, there are people who insist that market index, or ETF[2] is the best portfolio. Hence, managers should do nothing but just follow the market. In other words, they claim passive management[3] over active management[4] of portfolio. EMH[5] and CAPM[6] provides theoretical background for this passive management strategy. It is still a controversial issue whether it is possible to beat the market or not.

Problem Statement

In this project, we will try to refute Efficient Market Hypothesis by outperforming[7] the market index. Following active management, We will construct a portfolio of stocks with help of machine learning to simply do better than the market index, S&P500[8].
Followings are step by step strategies to achieve our desired goal:
1) Predict S&P500 future return from current input features: daily, weekly S&P500 return, other market return index(DAX[9], NIK[10]), trade volume, price of commodities, and so on.
2) For every stocks in S&P500, run regression to fit a line where x is S&P500 and y is an individual stock. Gradient of this fitted line is called beta in CAPM theory and it signals how stock move relative to the market.
3) We now have to select proper stocks depending on our S&P500 future forecast. Our solution guides us to select high beta stocks when market is expected to rise and select low beta stocks vice versa. Detail explanation will be given later for this logic.
4) After selecting bucket of stocks, we seek for optimal allocation weight for each stock that

[1] In finance, risk refers to volatility
[2] An ETF, or exchange traded fund, is a marketable security that tracks an index
[3] Passive management is an investing strategy that tracks a market-weighted index or portfolio
[4] Active management is a strategy where manager makes specific investments to outperform a benchmark index.
[5] Efficient Market Hypothesis
[6] Capital Asset Pricing Model
[7] Higher profit and less loss than benchmark
[8] The Standard ˙ Poor's 500, an American stock market index based on the market cap of 500 large companies
[9] Deutscher Aktienindex (German stock index)
[10] Nikkei 255, a stock market index for the Tokyo Stock Exchange

maximize portfolio's return or risk adjusted return.
5) In simulation, we will iterate step 1) - 4) for daily basis.

Metrics

Machine Learning techniques will be used in forecasting S&P500 return. For model evaluation and validation, we will be calculating the coefficient of determination, R^2, to quantify our model's performance. The coefficient of determination for a model is a useful statistic in regression analysis, as it often describes how "good" that model is at making predictions. The values for R^2 range from 0 to 1, which captures the percentage of squared correlation between the predicted and actual values of the target variable. A model with an R^2 of 0 always fails to predict the target variable, whereas a model with an R^2 of 1 perfectly predicts the target variable. Any value between 0 and 1 indicates what percentage of the target variable, using this model, can be explained by the features. A model can be given a negative R^2 as well, which indicates that the model is no better than one that naively predicts the mean of the target variable.
Furthermore, dealing with time series data, we will use roll forward cross validation which guarantees training data always being before testing data. This is to avoid data leakage or looking into future which can lead to unrealistically optimistic result. This is implemented as TimeSeriesSplit in sklearn library.
When deciding optimal allocation for each stocks in our portfolio, we will compare among three metric function (1) Function that maximize Sharpe ratio[11] (2) Function that maximize return (3) Function that minimize risk/variance and select the best among three metrics.

[11] Sharpe Ratio is a way to examine the performance of an investment by adjusting for its risk.

II. Analysis

Data Exploration & Exploratory Visualization

For historical stock price data(2010 - 2016), we will source it from: Quandl API
It provides various access to financial data in python. There are many options for representing stock price data. In this project we will use adjusted close price, because this compensates for any price anomaly resulting from dividends or stock split.

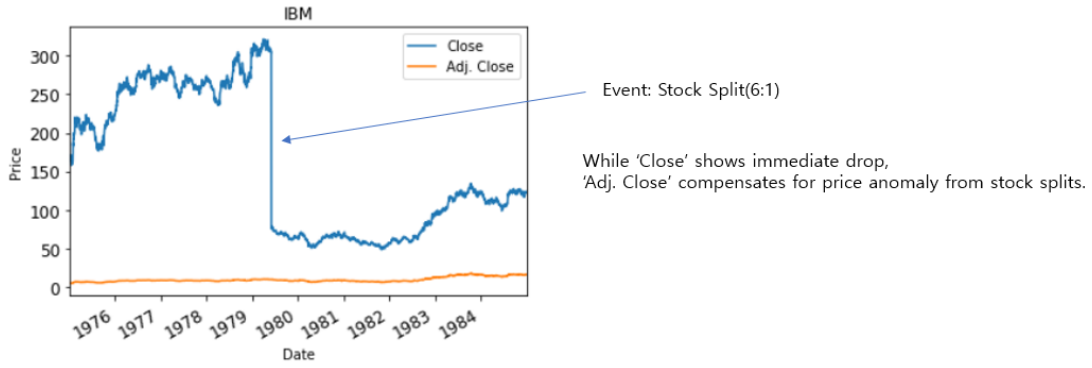


Figure 1: Close vs Adj. Close in IBM's price

Since this is financial data from market, there might be some missing values for each stocks. There are two approach for dealing with these missing data. One is interpolation[12] and other is filling forward/backward with last known value. We will choose to fill with last known value. If we choose to interpolate, we might give information about future to our learning model and get optimistic result.

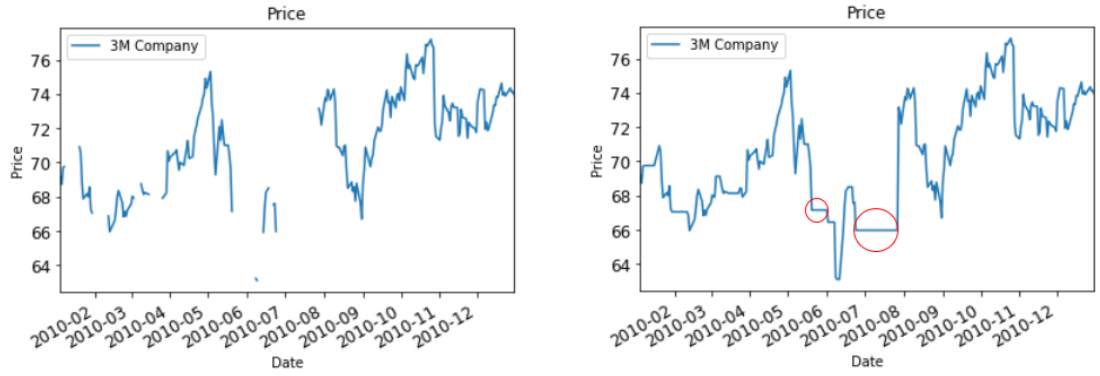


Figure 2: Filling missing data with last known value

Now let's explore some input price and volume data. Figure 3 plots stock price and prints statistics. Distribution of price here seems as bimodal, hence statistic here is not so meaningful since it is far from normal distribution.
Figure 4 plots stock volume and its statistics. Distribution of price here seems as much more like normal distribution than price. In addition, plot somewhat looks like a noise from i.i.d[13] distribution. However, its kurtosis value is high meaning fat tail or some outliers existing.

[12] Interpolation is a method of constructing new data points within the range of a discrete set of known data points.
[13] Independent and identically distributed random variables

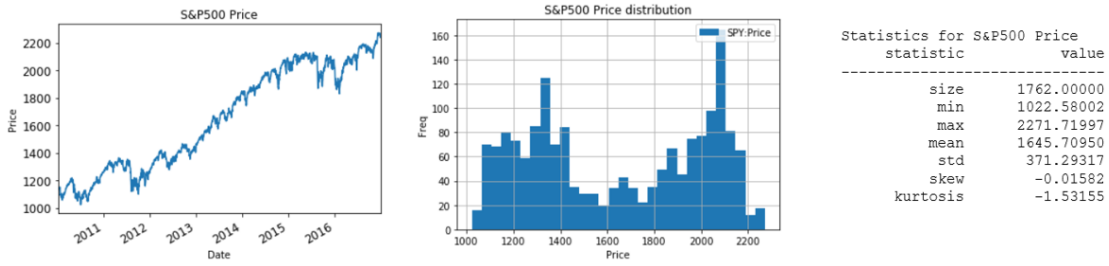


Figure 3: Plot and distribution for stock price

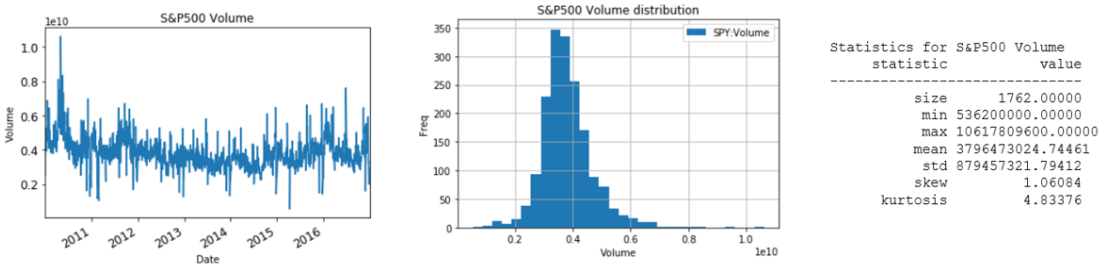


Figure 4: Plot and distribution for stock trade volume

Algorithms and Techniques

We will divide our market beating strategy into three main stream:
1) Forecasting S&P500 return from current and past data using machine learning.
2) Selecting appropriate stocks(based on CAPM) depending on market forecast.
3) Finding optimal weights for stocks that maximize portfolio's return, using optimizer.

1) Forecasting S&P500 return from current and past data:

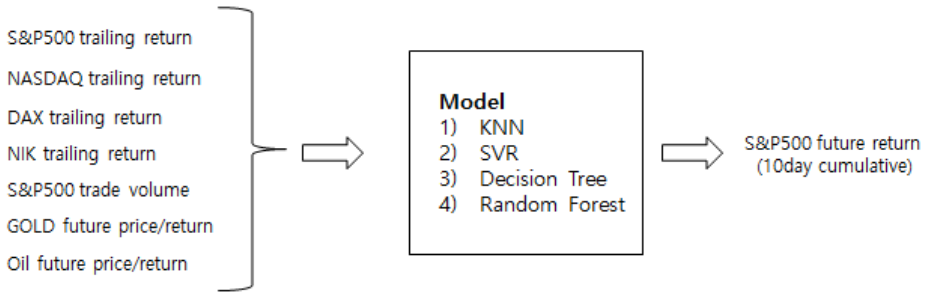


Figure 5: Description and mechanism for how learning model is used in this project

We will use regression model learned from selected input features. Various models will evaluated on R^2 score metrics. Candidates are KNN Regressor, Decision Tree Regressor, SVR[14], and Random Forest Regressor. KNN regression can be used for it's ease of interpretation, low calculation time, and also when underlying model is nonparametric. We can use parameter

[14] Support Vector Regressor

such as number of neighbors or weights to generalize better and avoid over fitting. SVR also generalizes well in high dimensional space with help of kernel trick. This is suitable for our case where we have many input features. We can also tune parameter such as 'C', 'epsilon', 'gamma' to cope with any outliers or over fitting. Furthermore, SVR will help us generalize better than other algorithm thanks to it's concept of maximizing margin. We can also use decision tree and random forest. Decision tree is known to be less sensitive to feature selection, and scale difference among features. Also decision tree seems to work well even with some outliers in high dimensional space. We would have to select good choice for depth of tree dealing with bias-variance trade off. Ensemble method usually provide better result than single estimator. Hence, we will also run simulation with random forest algorithm and see how it results. This random forest algorithm will help us to generalize better by tuning number of estimators and depth of tree. Also, it is robust to noise and outliers thank to it's randomness. Rather than testing all the parameters manually, we will exploit grid search algorithm to automate tuning according parameters for each model.

2) Selecting appropriate stocks depending on market forecast:
To construct market beating portfolio, we should select appropriate stocks depending on market. For instance, we would select high beta stocks if we are in bull market[15], and low beta stock if in bear market[16]. Refer to Figure 6 for detail explanation on this logic.

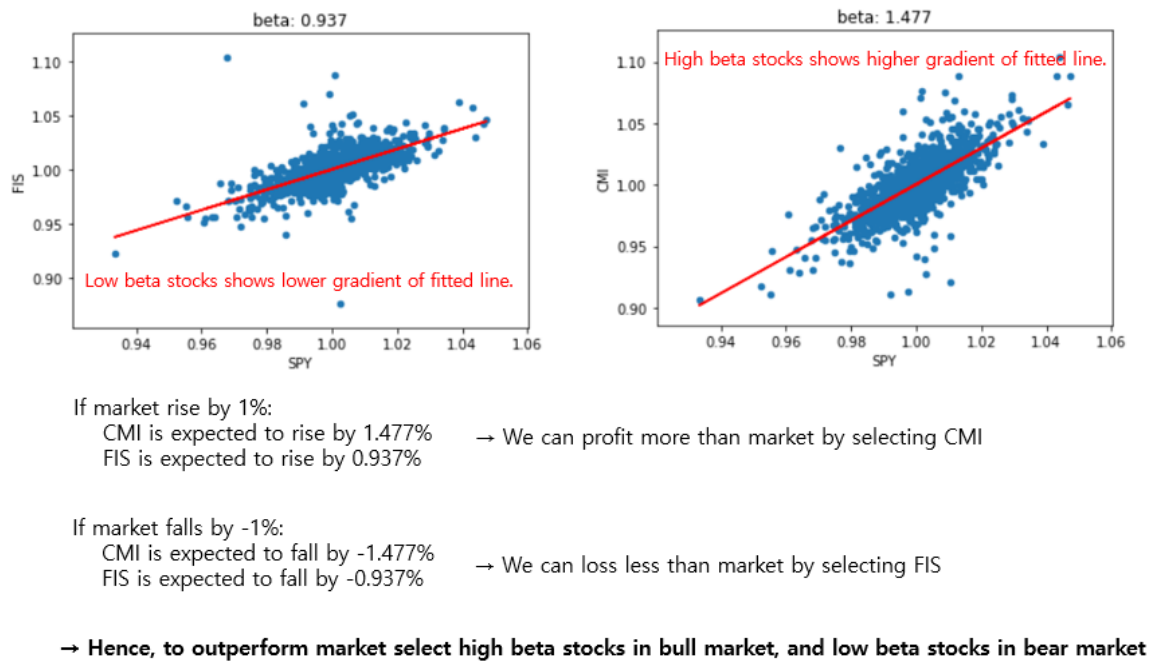


Figure 6: How to select stocks depending on market based on CAPM

To derive beta value for every stock, we have to run linear regression where x domain is market and y domain is specific stock we want to calculate. We do not want to use stocks that have poor correlation with the market because this means that price movement is somewhat random when compared to the market. We want stocks that show strong relationship to market.

[15] A bull market is a period of generally rising prices.
[16] A bear market is a general decline in the stock market over a period of time

3) Final step is to decide optimal allocation for each stocks that would be in our portfolio. Point here is that even though we have same stocks in our portfolio, how we allocate weights to them can greatly affect in overall performance of our portfolio. In this project, we optimize this weights using historical data. We use optimizer which maximize our given function. We will compare between three metric function. (1) Function that maximize Sharpe ratio (2) Function that maximize return (3) Function that minimize risk/variance

Benchmark

Our benchmark will be S&P500 index since S&P500 index is one of the most commonly followed equity index, and many consider it the best representations of the U.S. stock market and a bellwether for the U.S economy. Since S&P500 index is based on the market cap. of 500 large companies listed on NYSE/NASDAQ, it narrows our stock selection to 500 large companies. We might had to look over all companies listed on NASDAQ otherwise. We will plot and compare daily portfolio value of S&P500 versus our portfolio and see how learned portfolio can outperform the S&P500 index. Simulation range will be from 2016-01-01 to 2016-12-31.

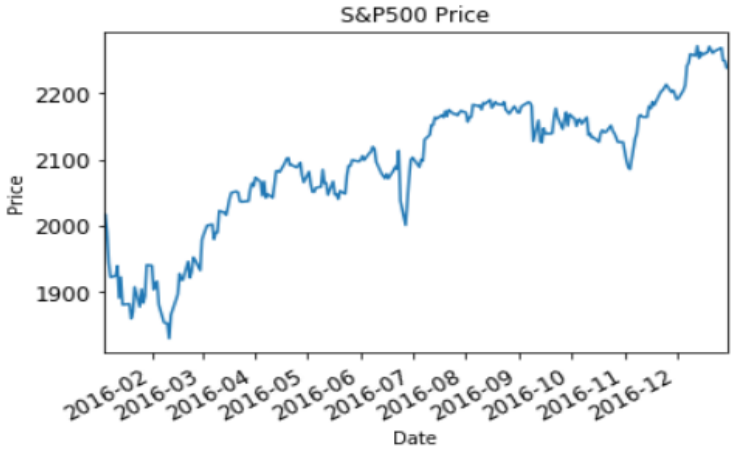


Figure 7: S&P500 performance in 2016

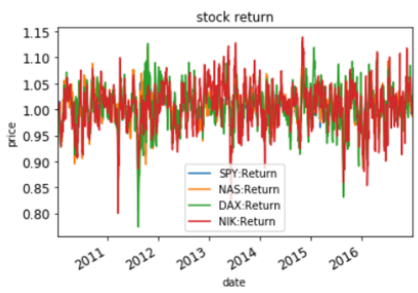
III. Methodology

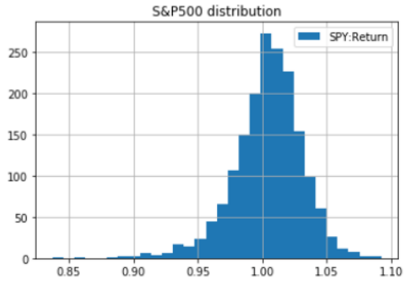
Data Preprocessing

As we have seen in Data Exploration, price of stock data is not normal distribution. Although, it is not mandatory for it to be normal distribution, many of portfolio theory assumes underlying normal distribution. Hence, we should better transform price data to return data as mentioned below.

$$n - day\ cumulative\ return = \frac{p_t}{p_{t-n}} \tag{1}$$

As you can see from Figure 8, distribution now look similar to normal distribution. However statistics show high kurtosis meaning that distribution has heavier tails and a sharper peak than the normal distribution. It is generally known that stock return shows fat tail. 2008 financial crisis is a good example of this fat tail. Some can argue that this fat tail as outliers, however in this project we won't remove any outliers since it does not seen to deviate too much from the mean value.

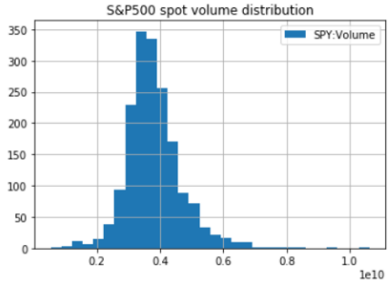


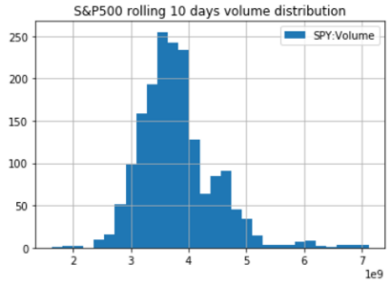



```
Statistics for SPY:Return
     statistic           value
-------------------------------
          size      1762.00000
           min         0.83702
           max         1.09246
          mean         1.00425
           std         0.02691
          skew        -0.78328
      kurtosis         2.49443
```

Figure 8: Plot and distribution for stock return

On the other hand, trade volume is similar to normal distribution except for some outliers and fat tail from high kurtosis. It was difficult to use this volume as an input data because it looked like i.i.d distribution. Trade volume for tomorrow is not so much affected by today's trade volume. Since we are trying to forecast 10-day future, I thought that some trend of volume movement






```
Statistics for SPY:Volume(10D)
     statistic                value
-----------------------------------
          size           1762.00000
           min 1626553000.00000000
           max 7121765000.00000000
          mean 3801428562.76958466
           std  655254810.13675213
          skew          1.15130
      kurtosis          3.04685
```

Figure 9: Plot and distribution for rolling mean volume data

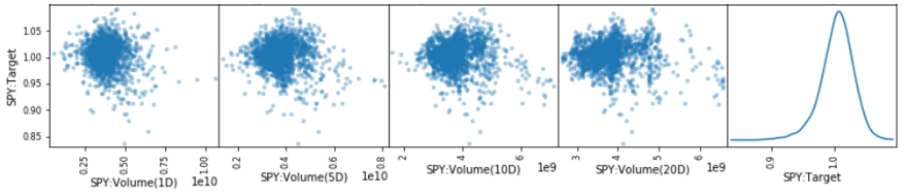
might help in describing market movement. So, I added rolling mean(5day, 10day, 20day) and sought for any relationship.

$$n - day\ rolling\ mean = \frac{v_t + v_{t-1} + ... + v_{t-n}}{n} \tag{2}$$

In addition, when selecting stocks for our portfolio, we removed some potential candidates list from S&P500. The reason behind this is to avoid survival bias[17]. We might end up with optimistic result if we select survived companies from S&P500 2016 constituent. Hence, we carefully select list of stocks from 2010 S&P500's list.

Selecting Features & Refinements

Before teaching our model, I wanted to see relationship between our target value and input features. For any transform that I have applied to volume feature, I could not find any meaningful correlation with target value. So, I dropped trade volume from input feature.



	SPY:Target
SPY:Volume(1D)	-0.121729
SPY:Volume(5D)	-0.084534
SPY:Volume(10D)	-0.040558
SPY:Volume(20D)	-0.045456
SPY:Target	1.000000

Figure 10: Scatter plot between target value and volume input features

	SPY:Target	GLF:Return(10D)	SPY:Return(10D)	NAS:Return(10D)
SPY:Target	1.000000	-0.033123	0.429326	0.393964
GLF:Return(10D)	-0.033123	1.000000	-0.003888	-0.009722
SPY:Return(10D)	0.429326	-0.003888	1.000000	0.951875
NAS:Return(10D)	0.393964	-0.009722	0.951875	1.000000

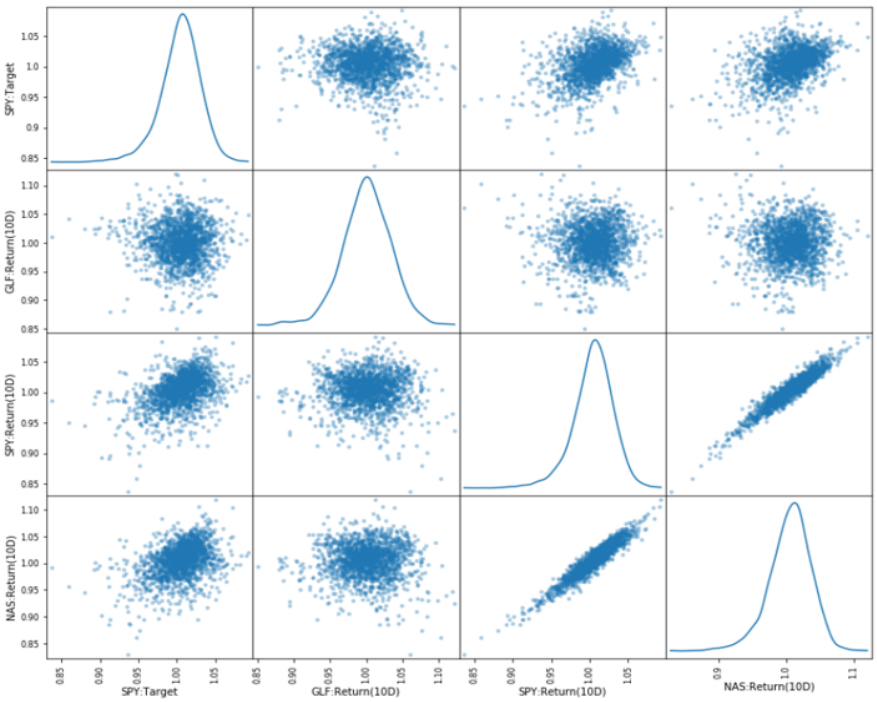


Figure 11: Scatter plot between target value and price input features

[17] Survival bias, is the logical error of concentrating on the people or things that "survived" some process and inadvertently overlooking those that did not because of their lack of visibility which can lead to false conclusions in several different ways

Also if you see from Figure 11, Gold Futures is not helpful in explaining S&P500 movement and NASDAQ return is redundant since it is almost similar to S&P500 itself. Hence, I removed those two features from input sets. Finally, I have add S&P500 1day, 5day, 20day ago cumulative return to our feature sets because many technical analyst use this n-day trailing data looking for signal in market.

Implementation

1) Forecasting S&P500 return from current and past data:
For implementing model to predict S&P500 future return, comparisons were made between KNN Regressor, DecisionTree Regressor, SVR, and RandomForest Regressor. R^2 score was used for performance metric, grid search techniques, and time series split cross validation scheme was used to search optimal parameters. It was shown that random forest regressor had the best performance with r2 score of 0.51.

```
def fit_regression(reg, X, y, params):
    """ Performs grid search over passed parameter for a
        passed regressor trained on the input data [X, y]. """
    """Cross Validation Scheme"""
    tscv = TimeSeriesSplit(n_splits=10)
    """Transform 'performance_metric' into a scoring function using 'make_scorer'"""
    scoring_fnc = make_scorer(r2_metric)
    """Create the grid search object"""
    reg = GridSearchCV(reg, params, scoring = scoring_fnc, cv = tscv)
    """Fit the grid search object to the data to compute the optimal model"""
    reg = reg.fit(X, y)
    """Return the optimal model after fitting the data"""
    return reg.best_estimator_
```

2) Selecting appropriate stocks depending on market forecast:
For stock selection based on capital asset pricing model, we had to get beta value for every stock. Below is how we ran linear regression and also correlation value which we use later as a threshold.

```
def get_stocks_CAPM(stocks):
    df = pd.DataFrame(columns=['beta','alpha', 'corr'])
    daily_return = proc.get_daily_returns(stocks)
    corr = daily_return.corr(method='pearson')

    for col in daily_return:
        """linear regression to fit"""
        beta, alpha = np.polyfit(daily_return['SPY'], daily_return[col], 1)
        df.ix[col] = [beta, alpha, corr['SPY'][col]]

    return df
```

3) Finding optimal weights for stocks that maximize portfolio's return:
Finally, below is how we got optimized weight that maximize given function. Sharpe ratio maximizing is for an example. See how it use negative sign to change maximization to minimization.

```
    def statistics(weights):
        weights = np.array(weights)
        pret = np.sum(ret.mean() * weights)
        pvol = np.sqrt(np.dot(weights.T, np.dot(ret.cov(), weights)))
        return np.array([pret, pvol, pret/pvol])

    def min_func_sharpe(weights):
        return -statistics(weights)[2]

    ..
    """get optimized weight that minimize min_func_sharper function"""
    cons = ({'type':'eq', 'fun':lambda x: np.sum(x) - 1})
    bnds = tuple((0,1) for x in range(noa))

    opts = sco.minimize(min_func_sharpe, noa * [1./noa,],
                        method='SLSQP', bounds=bnds, constraints=cons)
```

IV. Results

Model Evaluation and Validation

For predicting S&P500 future return, various models were used. Candidates were KNN regressor, SVR regressor, Decision Tree regressor, and random foreset regressor. By running grid search techniques we were able to find optimized parameter with ease. Below are tables for each model with parameters and their final scores. Highest score is from random forest regressor. If goal of this project was to just predict S&P500 market return, we will use random forest for our model. However, since we have see how our constructed portfolio outperforms S&P500 index, we would try others model to when we actually run simulation.

Model	R2 score	max_depth	n_estimator	n_neighbor	weights	Kernel	C	epsilon
Random Forest	0.512	3	20	-	-	-	-	-
KNN	0.274	-	-	27	'distance'	-	-	-
Decision Tree	0.492	3	-	-	-	-	-	-
SVR	0.484	-	-	-	-	'rbf'	10.0	0.001

Figure 12: Table for score and optimal parameter in learning models

In addition, to validate the robustness of this model, I removed some input features(return for DAX, return NIK), added some new input features(SPY cumulative return(3-day, 30-day, 50-day). Such perturbation in training data did not affect the result greatly.

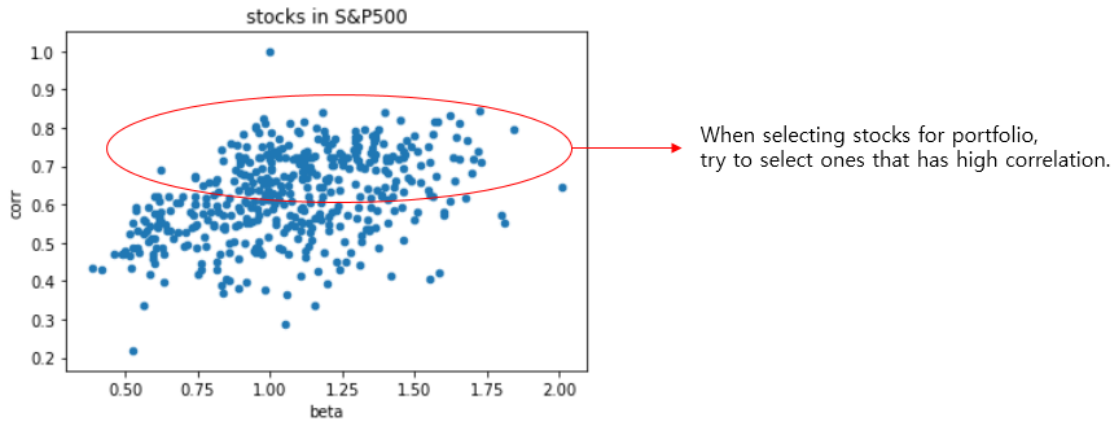


Figure 13: Scatterplot for beta and correlation for stocks in S&P500

For selecting our stocks in portfolio based on beta, there are two parameters we should consider. (1) Number of company in portfolio(default = 20) (2) Correlation Threshold(default = 0.7). High beta stocks tends to increase or decrease greater than the market, and we want to be more sure about this movement. Stocks with lower correlation to market might show more random movement compared to market. By running simulation, we would later find out optimal parameters for stock selection.

Finally, when we optimize allocation for stocks in portfolio, we said that three approaches will be considered: 1) maximize Sharpe ratio 2) maximize return 3) minimize risk. By running sim-

ulation, we would later find out what minimize function would result in greater performance.

Justification

Now let's run simulation and see how our constructed portfolio outperform market index.
1) Forecasting S&P500 return from current and past data:
We will use data from 2010 to 2015 for training. Models are (1)KNN(neighbors=27, weights='distance'), (2)DecisionTree(maxdepth=3), (3)SVR(kernel='rbf', C=10.0, epsilon='0.001'), (4)Random Forest(maxdepth=3, estimator='40')
2) Selecting appropriate stocks depending on market forecast:
(1) Number of company is portfolio(10, 20, 30) (2) Correlation Threshold(default = 0.7)
3) Finding optimal weights for stocks in portfolio:
(1) Maximizing Sharpe Ratio (2) Maximizing Return (3) Minimizing Risk

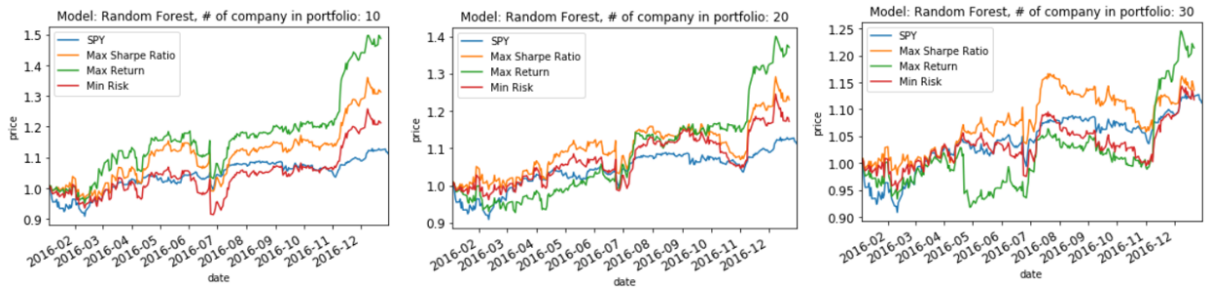


Figure 14: Result: Our portfolio versus S&P500 index(Model: Random Forest)

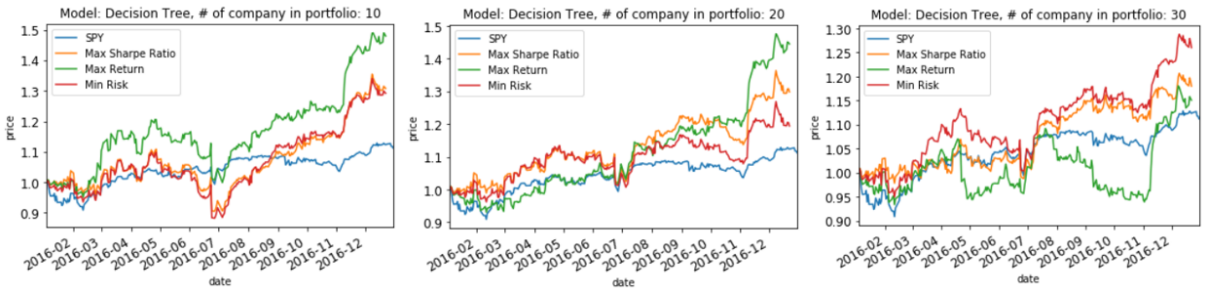


Figure 15: Result: Our portfolio versus S&P500 index(Model: Decision Tree)

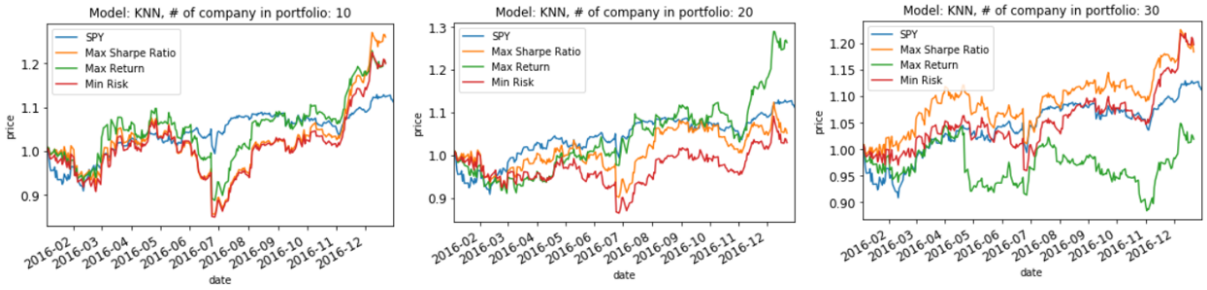


Figure 16: Result: Our portfolio versus S&P500 index(Model: KNN)

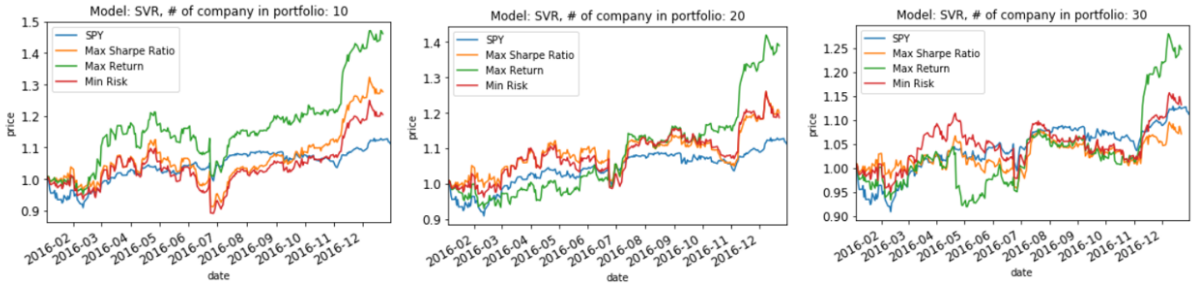


Figure 17: Result: Our portfolio versus S&P500 index(Model: SVR)

Learning Model: Portfolio constructed with Decision Tree, Random Forest, and SVR generally outperforms S&P500 index. However, portfolio constructed with KNN model shows similar or worse performance to S&P500 index. This coincides with our model evaluation and validation that Random Forest, Decision Tree, SVR scored high, and KNN scored low. KNN's poor return/performance is from it's poor prediction.

Number of company in portfolio: In general, portfolio's performance go down as we increase number of companies in our portfolio. This maybe because assuming in bear market, as we include more and more stocks, there is possibility of including low beta stocks as we just take sorted 20 or 30 items from the top.

Optimizing Function: In general, when maximizing Sharpe ratio in optimizer for getting weights, portfolio's performance is somewhat stable and it is always over S&P500 index. Using maximizing return function, we really do great in return perspective, but there are some case where it dramatically drop below S&P500 index. This maybe result of not considering any risk factor.

V. Conclusion

Free-Form Explanation & Visualization on Simulation

Our strategy in beating market was to 1) forecast market 2) High-beta stocks for bull market or Low-beta stocks for bear market 3) Optimize allocation that maximize Sharpe ratio of portfolio. We ran simulation based on above three strategies. Our simulation ran from 2016-01-04 to 2016-12-31. We make 5 day ahead SP500 forecast everyday. Whenever there is signal of changing from bull to bear or bear to bull, we re-select our stocks depending on market forecast. By default, we include top 20 high/low beta companies that have correlation value greater than threshold.

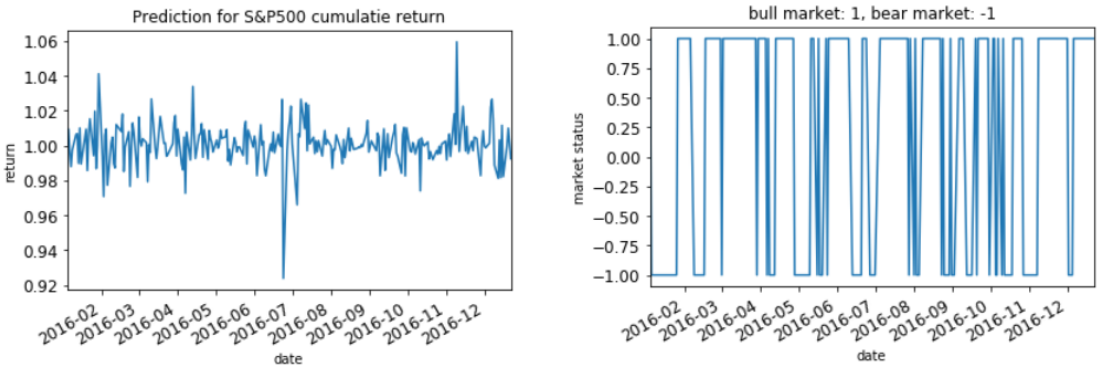


Figure 18: 2016 bull market and bear market prediction

Reflection

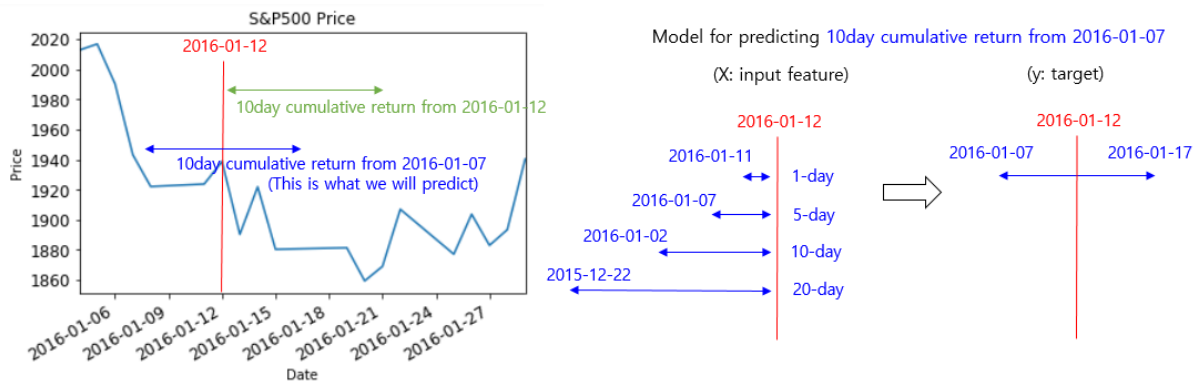


Figure 19: Scheme for predicting S&P500 future cumulative return

When I started this project, I intended to predict tomorrow's SP500 price/return and this was particularly difficult for me. I have tried various financial data such as options, commodity, interest rates and others. Suddenly, rather than forecasting daily story, I thought it would make more sense if we predict weekly or monthly atmosphere of the market. And this made more sense to our project since our strategy depends on market status. Hence, I tried to predict 10-day cumulative S&P500 return from today using 1-day, 5-day, 10-day, and 20-day trailing data.

Although, it got better than before, the result was not so meaningful as expected. Still, I think this is really difficult to predict how price will be after ten day. Alternatively, rather than forecasting 10-day forward return from today, I tried to forecast 10-day forward return from price of 5-day before. This brought us more meaningful result, since we could use past 5day information in predicting future return. Although, this is not exactly predicting future from current spot, I think this is meaningful approach, because at least this prediction signal you how market will move in after 5-day.

Improvement

In this project, we predicted 10 day future cumulative S&P500 return. If we were to predict monthly market movement, I think there is some chance that we might have done better. Also, we divided market prediction to two segments: Bull, Bear. What if we divide it into more, such as strong bull, strong bear, and maybe let's just call it a steady market.
When optimizing weight for each stock that maximize our portfolio's Sharpe Ratio, we used historical mean, and covariance data for each stock. I think we could have achieved better solution, if we could have use good prediction value. However, predicting price and covariance for 500 stocks included in S&P500 is very difficult and time-consuming task.