Machine Learning for Portfolio Selection

1. Momentum and reversal effect detection extension:

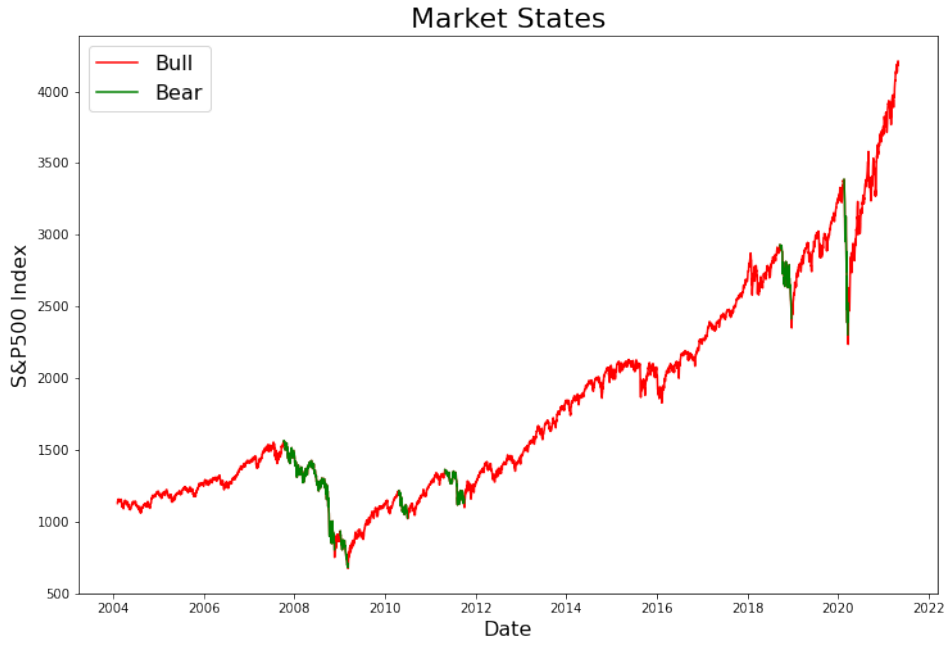

Figure 1. Bull and Bear Market

By the algorithm mentioned in previous report, for US market, most time is bull market, but in fact, there are some periods have no effect at all or not very clear momentum effect. Therefore, this algorithm may be not suitable for our later research. Here, I introduce another method which also consider the no effect situation:

First, the algorithm defines an indicator Simple Moving Average as MA, which is a pretty useful factor to observe the market movements. Then, it uses the follows rules to determine the uptrend and downtrend.

(1). If closing price value leads its MA15 and MA15 is rising for last 5 days then trend is **Uptrend**, i.e., trend signal is 1.

(2). If closing price value lags its MA15 and MA15 is falling for last 5 days then trend is **Downtrend**, i.e., trend signal is -1.

(3). If none of these rules are satisfied then stock market is said to have **no trend**, i.e., trend signal is 0.

The Figure 2 shows the results for applying this algorithm on US market (S&P500), we can observe that by this method, the change between different market states is more frequent, which is more proper.

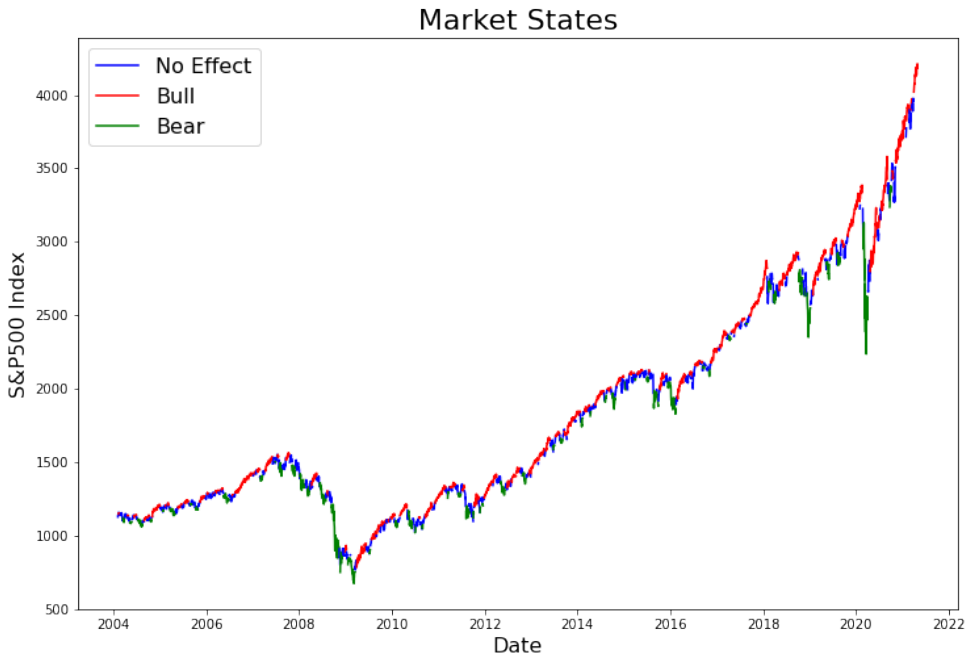

Figure 2. Bull and Bear Market (version 2)

Feature Selection

1. Filter method

Filter methods pick up the intrinsic properties of the features measured via univariate statistics instead of cross-validation performance. These methods are faster and less computationally expensive than wrapper methods. When dealing with high-dimensional data, it is computationally cheaper to use filter methods. There are multiple filter methods, but they are appropriate for different cases. For example, the Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with the best Chi-square scores. And correlation-based method like using **ANOVA** correlation coefficient is widely used for **numerical features and categorical target**, so this method is the most suitable one for our case.

In order to test the impact of this feature selection methods, Support Vector Machine is firstly used to detect the market states, then apply this filter method to select only 30 features from the original features and do the prediction again to compare the performance. Here, training set is from 2005 to 2015, and test set is from 2016/01/01 to 2016/05/01. The results are showing as Figure 3.

	precision	recall	f1-score	support
Bear	0.00	0.00	0.00	29
No Effect	0.03	0.12	0.04	8
Bull	0.77	0.76	0.76	45
accuracy			0.43	82
macro avg	0.27	0.29	0.27	82
weighted avg	0.43	0.43	0.42	82

	precision	recall	f1-score	support
Bear	1.00	0.86	0.93	29
No Effect	0.40	0.25	0.31	8
Bull	0.85	0.98	0.91	45
accuracy			0.87	82
macro avg	0.75	0.70	0.71	82
weighted avg	0.86	0.87	0.86	82

Figure 3. Classification report (Left: Plain SVM Right: SVM With Filter)

From the classification report, we can observe that the total accuracy for plain SVM is only around 42% and it perform poorly on detecting the Bear and No effect periods. While for the SVM model using filtered features, the performance increases significantly. The total accuracy rises to 86%, and this model can detect the Bull and Bear markets very well. For the No effect detection, it still performs badly, but one reason is that the data is imbalanced, which means there are very few points for no effect class in the test set.

2. Wrapper Method

Wrappers require some method to search the space of all possible subsets of features, assessing their quality by learning and evaluating a classifier with that feature subset. The feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. The wrapper methods usually result in better predictive accuracy than filter methods, but have more computation cost.

There are several wrapper method techniques (Forward Feature Selection, Backward Feature). **Forward Feature Selection** is an iterative method wherein we start with one best performing variable against the target. Next, select another variable that gives the best performance in combination with the first selected variable. This process continues until the preset criterion is achieved. While **Backward Feature Selection** works exactly opposite to the Forward Feature Selection method. It starts with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.

One of the best ways for implementing feature selection with wrapper methods is to use Boruta package in python that finds the importance of a feature by creating shadow features. Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features). Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important. At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e., whether the feature has a higher Z-score than the maximum Z-score of its shadow features) and constantly removes features which are deemed highly unimportant. Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs.

By applying this wrapper method, 61 features are selected from the original 68 features based on random forest model. From the follows figure 4, we can observe that the total accuracy increases sightly for the random forest model with wrapper features, and it also performs a little bit better on Bear and No effect market states, which is useful for the later on strategy building.

	precision	recall	f1-score	support
Bear	0.96	0.90	0.93	29
No Effect	0.50	0.88	0.64	8
Bull	1.00	0.91	0.95	45
accuracy			0.90	82
macro avg	0.82	0.89	0.84	82
weighted avg	0.94	0.90	0.91	82

	precision	recall	f1-score	support
Bear	0.96	0.93	0.95	29
No Effect	0.54	0.88	0.67	8
Bull	1.00	0.91	0.95	45
accuracy			0.91	82
macro avg	0.83	0.91	0.86	82
weighted avg	0.94	0.91	0.92	82

Figure 4. Classification report (Left: Plain RF Right: RF With Wrapper)

3. Embedded Method

These methods encompass the benefits of both the wrapper and filter methods, by including interactions of features but also maintaining reasonable computational cost. Embedded methods are iterative in the sense that takes care of each iteration of the model training process and carefully extracts those features which contribute the most to the training for a particular iteration.

In this project, Random Forest Importance is chosen to select the most important features. It is a kind of a Bagging Algorithm that aggregates a specified number of decision trees. The tree-based strategies used by random forests naturally rank by how well they improve the purity of the node, or in other words a decrease in the impurity (Gini impurity) over all trees. Nodes with the greatest decrease in impurity happen at the start of the trees, while notes with the least decrease in impurity occur at the end of trees. Thus, by pruning trees below a particular node, we can create a subset of the most important features.

Then by the random forest importance method, 28 features are selected, which are able to explain more than 95% importance of all the features. These selected features are used on SVM and random forest model to test the performance. From the figure 5, we can see although the performance of random forest keeps the same, it's better to reduce dimension. After we feed the selected features to SVM, the model performs pretty well, whose total accuracy increases to around 93%.

	precision	recall	f1-score	support
Bear	0.96	0.93	0.95	29
No Effect	0.50	0.88	0.64	8
Bull	1.00	0.89	0.94	45
accuracy			0.90	82
macro avg	0.82	0.90	0.84	82
weighted avg	0.94	0.90	0.91	82

	precision	recall	f1-score	support
Bear	0.96	0.93	0.95	29
No Effect	0.64	0.88	0.74	8
Bull	1.00	0.96	0.98	45
accuracy			0.94	82
macro avg	0.87	0.92	0.89	82
weighted avg	0.95	0.94	0.94	82

Figure 5. Classification report (Left: RF with Embedded Right: SVM With Embedded)

In this project, Principal Component Analysis is also used to reduce dimension, but the performance is even worse, and seems lose a lot information. Therefore, after compare the above methods, this project decides to use the embedded method (random forest importance) to select the features, which is the most efficient one.