Machine Learning for Portfolio Selection

QF5401

Zhang Zijie

A0207133U

1. Project Description:

There are a lot of papers have researched on stock technical analysis by machine learning, and this project's idea comes from them but will have some improvements. In stock markets, there are two common market state: bull and bear market. When the market is under the bull market, the uptrend is dominating, and the past winners will probably continue to outperform the benchmark in the future, while under bear market, the reversal effect is common. Based on this fact, this project will use several machine learning models to predict the market states and build a corresponding trading strategy.

The codes and other sources for this project has been pushed to GitHub, and can be access to through: https://github.com/zhangzijie-lab/QF5401.git

This project will research on US market and choose S&P 500 components as the trading universe, and use the history data from 2005 to 2021-04. To study the bull and bear market, here we define **observation and holding periods**, which can be in hours, days or any time horizons depending on the trading preference. Here, we used daily bars, so the time horizon is days. Because different observation and holding periods will have significantly different impacts for the results, we will back test several pairs of the periods. The stocks in the universe will be ordered by their returns in the observation periods, and the top N stocks will be defined as past winners, and the bottom N stocks will be past losers. Then machine learning techniques will be used to predict the future market state (bull, bear, fluctuation) or the trend (uptrend, downtrend, or sideways). Based on the predicted future trend, different trading operation will be applied. Lastly, back test this trading strategy to examine the applicability of it.

2. Bull and Bear market detection:

As we will use supervised machine learning to do prediction, we should pre-define our target, which is the market states (bullish and bearish) for S&P500. But there is no generally accepted formal definition of bull and bear markets in the finance literature, after researching on several popular algorithms, this project tested the following two algorithm to determine the market state. The first one is from Lunde and Timmermann (2004)[1]:

In general, this algorithm imposes a minimum on the price change since the last peak or trough. Let λ_{bull} be a scalar defining the threshold of the movement in stock prices that triggers a switch from a bear state to a bull state and similarly, let λ_{bear} be the threshold for shifts from a bull state to a bear state. The algorithm first finds the maximum close price on the time interval [t_0, t], then computes the relative change:

$$\delta = \frac{P_{max} - P_t}{P_{max}}$$

If $\delta > \lambda_{bear}$, then a new peak is detected at time t_{peak} at which close price attains a maximum on [t_0, t]. The period [$t_0 + 1$, t_{peak}] is labelled as a bull state. A bear state begins from t_{peak}+ 1, which is the new t_0.

Similarly, the logic is used for the trough detection. If $\delta = \frac{P_t - P_{min}}{P_{max}} > \lambda_{bull}$, then a new trough is detected at time t_{trough} at which close price attains a minimum on [t_0, t].

After applying this algorithm, we get the following results as figure 1 shows for bull and bear market states.

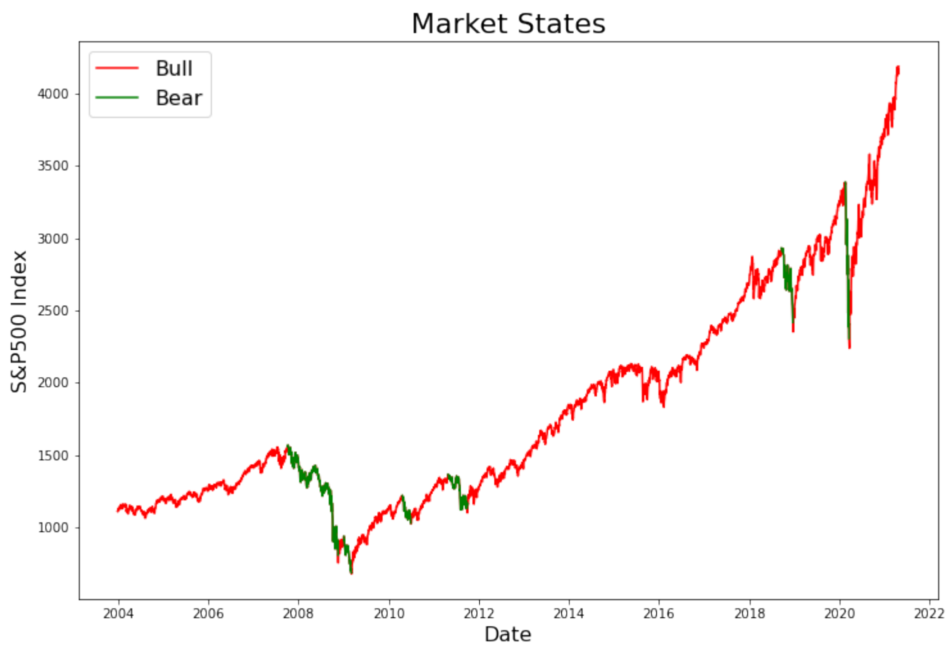


Figure 1. Bull and Bear Market

From Figure 1 we can observe that, for US market, the most time is bull market, but in fact, there are some periods that are fluctuating or do not have very clear trends. Therefore, this algorithm may be not suitable for our later research. Here, this project considers another method[2] which also consider the sideways 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 MA25 and MA25 is rising for last 5 days then trend is **Uptrend**, i.e., trend signal is 1.

(2). If closing price value lags its MA25 and MA25 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 **sideways**, 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 switch between different market states is more frequent, which is more proper.

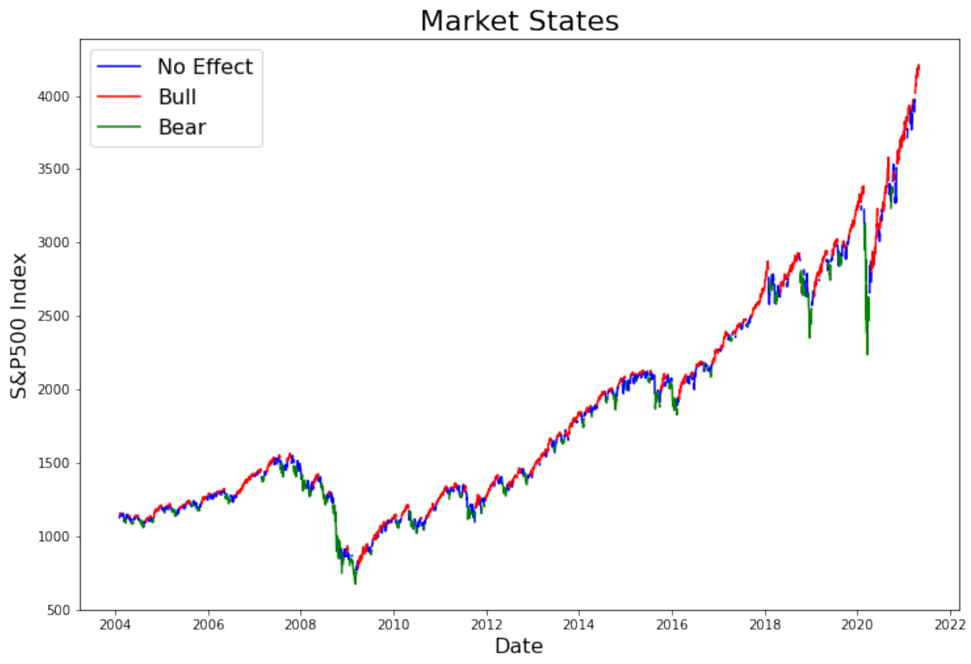


Figure 2. Bull and Bear Market (version 2)

3. Feature Engineering

Features play a pretty important role in almost all machine learning problems. Meaningful features will increase the model's performance significantly, while useless features will ruin the model, which is "garbage in, garbage out". When inputting features into a model, the goal is to feed the model with features that are relevant for predicting a class. Including irrelevant features poses the problem of unnecessary noise in the data, resulting in lower model accuracy. Other than the regular market quotes (e.g. prices (open, high, low, close), volume, etc.), we will calculate several kinds of technical indicators including volume indicators (e.g. Money Flow Index, On-Balance Volume, etc.), volatility indicators (e.g. volatility of return, Bollinger Bands, etc.), trend indicators (e.g. Moving Average, Moving Average Convergence Divergence, Average Directional Movement Index, etc.), and momentum indicators (e.g. Relative Strength Index, True Strength Index, Stochastic Oscillator, etc.)

But some technical factors have high correlation as the Figure 3 shows, and too many features will meet the curse of dimensionality, so feature selection is necessary in this part. This project will apply several techniques to reduce the dimension, such as filter-based feature selection methods, Principal component analysis.

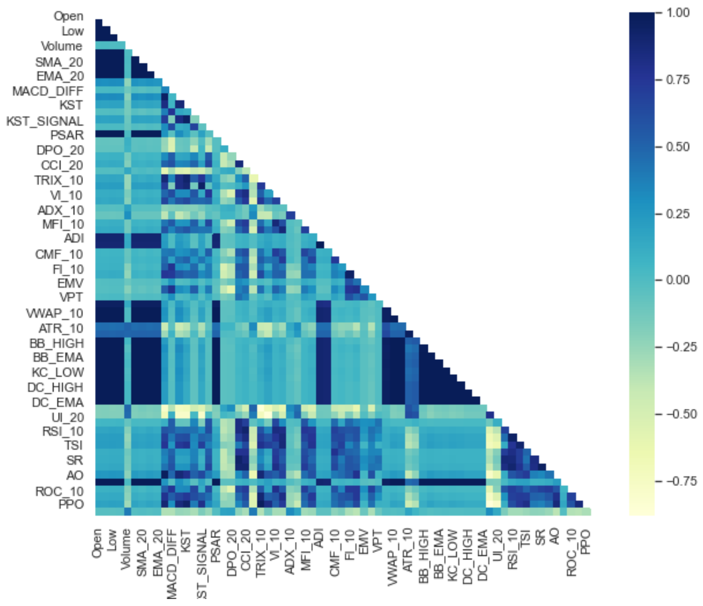


Figure 3. Correlation between features

4. Feature Selection

There are three reasons why feature selection is very important:

Curse of dimensionality — Overfitting: If we have more columns in the data than the number of rows, we will be able to fit our training data perfectly, but that won't generalize to the new samples.

Occam's Razor: We want our models to be simple and explainable. We lose explain ability when we have a lot of features.

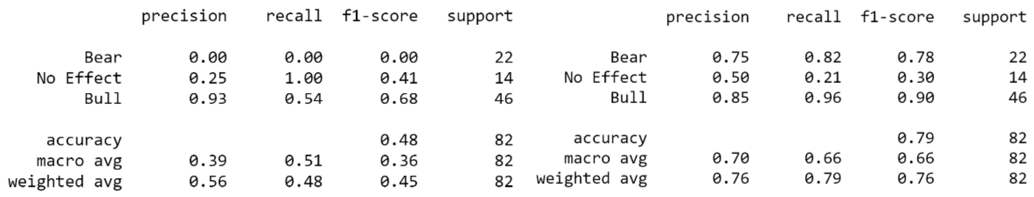
Garbage In Garbage out: Most of the times, we will have many non-informative features. For Example, Name or ID variables. Poor-quality input will produce Poor-Quality output.

Overall, there are three feature selection methods, this project tested all these methods to find a most suitable one for this case.

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, and choose K as 15. The results are showing as Figure 4.



```
              precision    recall  f1-score   support

        Bear       0.00      0.00      0.00        22
   No Effect       0.25      1.00      0.41        14
        Bull       0.93      0.54      0.68        46

    accuracy                           0.48        82
   macro avg       0.39      0.51      0.36        82
weighted avg       0.56      0.48      0.45        82
```

```
              precision    recall  f1-score   support

        Bear       0.75      0.82      0.78        22
   No Effect       0.50      0.21      0.30        14
        Bull       0.85      0.96      0.90        46

    accuracy                           0.79        82
   macro avg       0.70      0.66      0.66        82
weighted avg       0.76      0.79      0.76        82
```

Figure 4. 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 45% 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 76%, 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 Selection). **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.

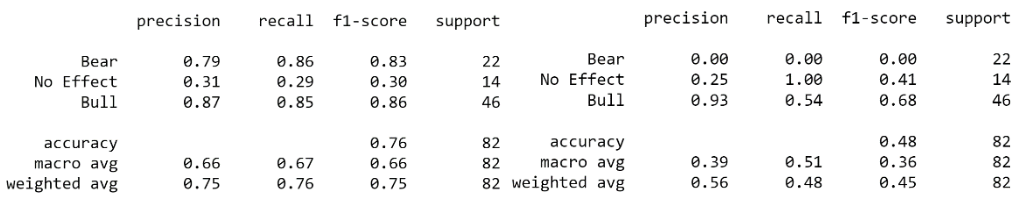
By applying this wrapper method, there are no features filtered out based on random forest model, so this method is not suitable for this case.

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, 48 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 the performance of random forest keeps the same, it's better to reduce dimension. After we feed the selected features to SVM, the model's performance does not change, so this method is also not proper for this problem.



```
              precision    recall  f1-score   support

        Bear       0.79      0.86      0.83        22
   No Effect       0.31      0.29      0.30        14
        Bull       0.87      0.85      0.86        46

    accuracy                           0.76        82
   macro avg       0.66      0.67      0.66        82
weighted avg       0.75      0.76      0.75        82
```

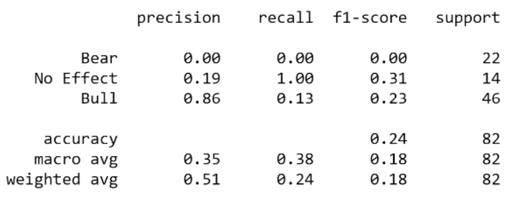
```
              precision    recall  f1-score   support

        Bear       0.00      0.00      0.00        22
   No Effect       0.25      1.00      0.41        14
        Bull       0.93      0.54      0.68        46

    accuracy                           0.48        82
   macro avg       0.39      0.51      0.36        82
weighted avg       0.56      0.48      0.45        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 (Shown in Figure 6). Therefore, after compare the above methods, this project decides to use the **filter method** to select the features, which is the most efficient one.



```
              precision    recall  f1-score   support

        Bear       0.00      0.00      0.00        22
   No Effect       0.19      1.00      0.31        14
        Bull       0.86      0.13      0.23        46

    accuracy                           0.24        82
   macro avg       0.35      0.38      0.18        82
weighted avg       0.51      0.24      0.18        82
```

Figure 6. Classification report (SVM with PCA)

As previous report mentioned, the later on models will use filter method to select features. But the number of features to select needs to be decided. Here precision measure is used to determine how many features to select, and the result shows as Figure 7. From the plot, we can observe that the precision and total accuracy is highest when number of features is larger than 30. As the curse of dimensionality, only 30 selected features will be used in the later research.

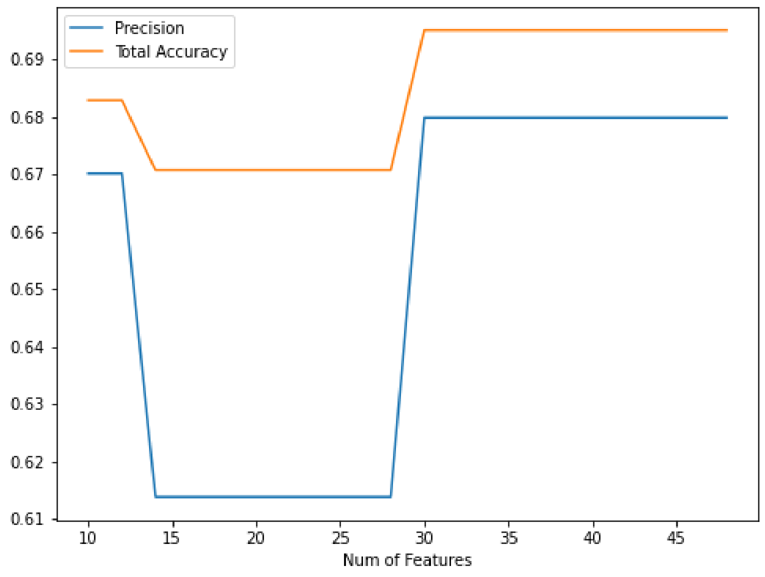
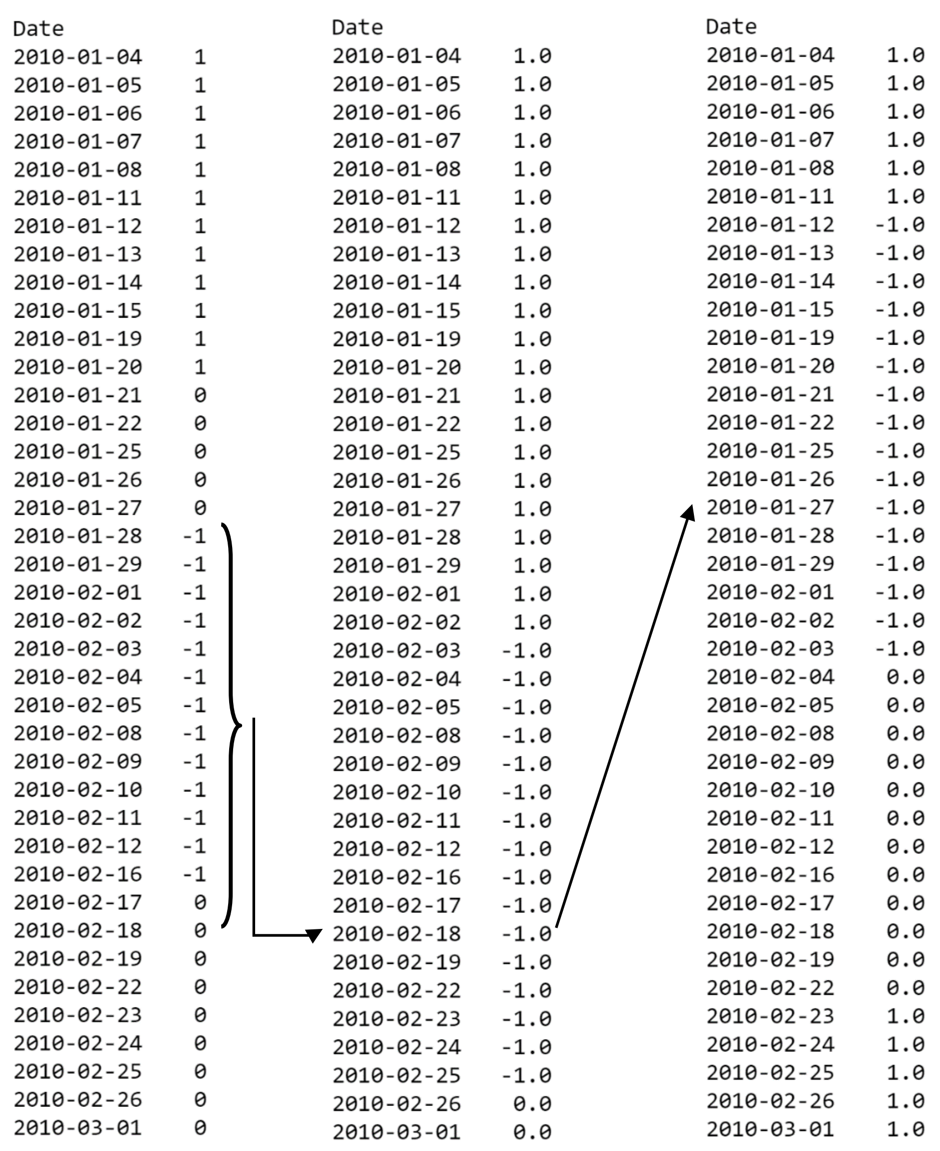


Figure 7. Number of Features versus Performance

5. Data Pre-processing

The results of the market states detection will be the target of this classification problem. But need to do a further modification, because we need to predict the future market state instead of the current one. Here, to study the bull and bear market, observation and holding periods are defined. The observation period is defined as J, while the holding period is defined as K. In practice, they can be in hours, days, or months. During observation period, winners and losers will be detected and future market state will be predicted depend on the data in this period. Holding period is the period holding the selected portfolio.

As the previous market states are for each time points, to get the market state for the period, one way is to choose the majority of a rolling window of length, and then shift K time steps backward to make it become the future prediction. This modification is shown as the Figure 8.



```
Date                      Date                        Date
2010-01-04    1           2010-01-04    1.0           2010-01-04    1.0
2010-01-05    1           2010-01-05    1.0           2010-01-05    1.0
2010-01-06    1           2010-01-06    1.0           2010-01-06    1.0
2010-01-07    1           2010-01-07    1.0           2010-01-07    1.0
2010-01-08    1           2010-01-08    1.0           2010-01-08    1.0
2010-01-11    1           2010-01-11    1.0           2010-01-11    1.0
2010-01-12    1           2010-01-12    1.0           2010-01-12   -1.0
2010-01-13    1           2010-01-13    1.0           2010-01-13   -1.0
2010-01-14    1           2010-01-14    1.0           2010-01-14   -1.0
2010-01-15    1           2010-01-15    1.0           2010-01-15   -1.0
2010-01-19    1           2010-01-19    1.0           2010-01-19   -1.0
2010-01-20    1           2010-01-20    1.0           2010-01-20   -1.0
2010-01-21    0           2010-01-21    1.0           2010-01-21   -1.0
2010-01-22    0           2010-01-22    1.0           2010-01-22   -1.0
2010-01-25    0           2010-01-25    1.0           2010-01-25   -1.0
2010-01-26    0           2010-01-26    1.0           2010-01-26   -1.0
2010-01-27    0           2010-01-27    1.0           2010-01-27   -1.0
2010-01-28   -1           2010-01-28    1.0           2010-01-28   -1.0
2010-01-29   -1           2010-01-29    1.0           2010-01-29   -1.0
2010-02-01   -1           2010-02-01    1.0           2010-02-01   -1.0
2010-02-02   -1           2010-02-02    1.0           2010-02-02   -1.0
2010-02-03   -1           2010-02-03   -1.0           2010-02-03   -1.0
2010-02-04   -1           2010-02-04   -1.0           2010-02-04    0.0
2010-02-05   -1           2010-02-05   -1.0           2010-02-05    0.0
2010-02-08   -1           2010-02-08   -1.0           2010-02-08    0.0
2010-02-09   -1           2010-02-09   -1.0           2010-02-09    0.0
2010-02-10   -1           2010-02-10   -1.0           2010-02-10    0.0
2010-02-11   -1           2010-02-11   -1.0           2010-02-11    0.0
2010-02-12   -1           2010-02-12   -1.0           2010-02-12    0.0
2010-02-16   -1           2010-02-16   -1.0           2010-02-16    0.0
2010-02-17    0           2010-02-17   -1.0           2010-02-17    0.0
2010-02-18    0           2010-02-18   -1.0           2010-02-18    0.0
2010-02-19    0           2010-02-19   -1.0           2010-02-19    0.0
2010-02-22    0           2010-02-22   -1.0           2010-02-22    0.0
2010-02-23    0           2010-02-23   -1.0           2010-02-23    1.0
2010-02-24    0           2010-02-24   -1.0           2010-02-24    1.0
2010-02-25    0           2010-02-25   -1.0           2010-02-25    1.0
2010-02-26    0           2010-02-26    0.0           2010-02-26    1.0
2010-03-01    0           2010-03-01    0.0           2010-03-01    1.0
```

Figure 8. target data (Left: Point Market State Mid: Period Market State Right: Shifted)

But from the Figure 9, we can observe that once the market from one state changes to another one, it will last more than one day, which will last at least one week in general. Therefore, using the majority of states during one period as the period market state may be not suitable, this project applies the follows rule to determine it: if there are more than 10% bear states in this period, then the state for this period will be bear market, else if there are more than 10% no effect then it will be seen as no effect, otherwise, this period state is bull market. This new rule can react faster for the market states changing than the previous majority rule, and make more sense for the later on trading strategy. This modified result will be the final target. Later on, the trading strategy will use the fitted machine learning model to predict the future market states based on the observation period data.

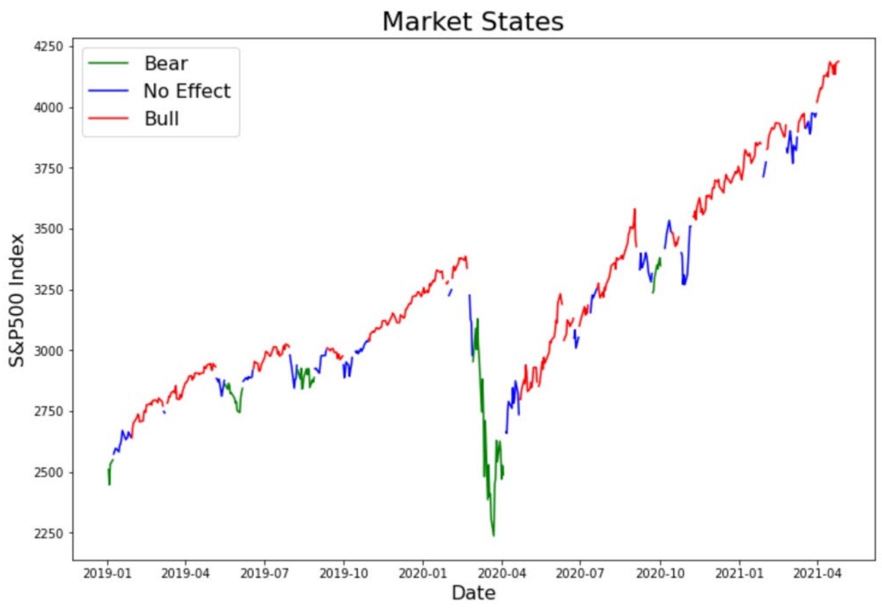


Figure 9. Market States during 2019-2021

After finishing the target generation, the data set also needs to handle the missing value and do Feature Scaling. For missing values, in this project, it's simple to deal with. Here, missing values are filled by previous values, and keep null if no previous value. Because the data is stock price time series, if there is one missing value on a trading date, it's reasonable to fill it with the pre-trading date stock price. For feature scaling, standardization is used to scale the features. Although the total accuracy decreases a little bit, the precision is higher. And from the confusion matrix of these two methods (shown as Figure 10), the misclassifications for class Bear and Bull are more concentrated on No effect class, and fewer misclassification for No effect class, which is better for the later on trading strategy. Because if the misclassification for Bear and Bull markets are all assigned to No effect class, the strategy will just not trade on these periods. In other words, when the model cannot clearly detect the momentum and reversal effect, it just assigns these sample to no effect, which makes more sense. And if there are a lot misclassification for No effect class, the portfolio performance will be ruined significantly. Because these misclassifications are to Bear or Bull class, which will trade the past winners and losers, this strategy will lose money based on the no effect fact. Therefore, the standardization process helps the model perform better and achieve the desired improvement.

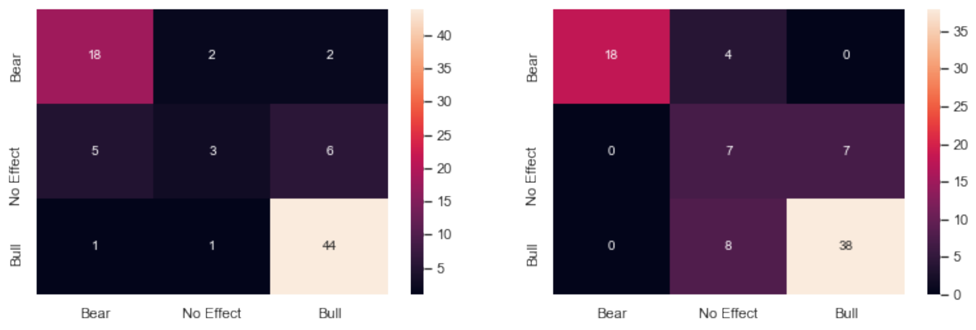


Figure 10. Confusion Matrix (Left: Plain SVM Right: SVM with Scaled)

6. Model Selection:

This market states detection can be seen as a classification problem, so we will choose several machine learning models from simple to complex as follows: Support Vector Machine, Decision Tree, Random Forest, Light GBM, and Multilayer Perceptron Neural Network.

Support Vector Machine (SVM) is very applicable to classification problems, and can overcome overfitting problems. Essentially, SVM uses the kernel function to project the inputs into high-dimensional feature spaces so that SVM can efficiently solve non-linear classification problems. In this research, we selected the radial basis function (RBF) as the kernel function, as described in follows formula:

$$K(X_1, X_2) = \exp\left(-\frac{\|X_1 - X_2\|^2}{2\sigma^2}\right)$$

Decision Tree (DT) is a very famous supervised learning technique. It uses a tree-like graph or model to make decisions. Given training data, DT is able to learn decision rules inferred from the data features during the training process. And DT is simple to understand and to interpret. The generated rules can be visualized easily.
In addition, DT is very fast, and there is little data preparation for DT. DT has been widely applied to operations research to help make decisions. In a word, DT will be one of the useful machine learning techniques to identify the momentum and reversal effects in this research.

Random forests (RF) are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance, so for this problem RF and boosted trees are both used to find the optimal model.

Light GBM is a high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks. But compared to XGBoost, Light GBM is very fast to train a large dataset, and the biggest difference between these two methods is the way of splitting tree. XGBoost uses the level-wise tree growth, while Light GBM uses leaf-wise tree growth. Leaf wise splits lead to increase in complexity and may lead to overfitting and it can be overcome by specifying another parameter max-depth which specifies the depth to which splitting will occur.

Multi-layer Perceptron (MLP) has been widely applied to supervised learning problems. The model is trained to learn the correlations between inputs and outputs. The model adjusts the parameters, weights, and bases from time to time to minimize errors in the training process. Designing a good network topology for the studied problem is a tough task. The numbers of layers and neurons on each layer and the selection of activation functions all affect the performance of the model. In practice, we have to try various topologies to acquire a good one. Therefore, after testing a lot types of structure, the model with hidden layer sizes (64, 128, 64, 32) is selected.

Also, to get the best performance of each model, the parameter tuning is necessary. After applying grid search for each model with 3 folds cross-validation based on precision score, here are the models with parameters:

SVM: gamma = scale, kernel = rbf, C=1

Random Forest: max depth = 3

Decision Tree: max depth = 3

Light GBM: number of estimators = 1000, learning rate = 0.1, max depth = 3

MLP: activation = logistic, solver = Adam, alpha=1e-4

7. Trading Strategy

In this research, rolling past 10 years data will be used to feed the model as training dataset, and predict the future market state based on the observation period J as mentioned before. But the model will predict the future at each time point in observation period, so some techniques should be used to determine the market state for the holding period based on a time series prediction. Here aged weighted technique is used to average the predictions instead of equal weighted, because the recent predictions are much more important than the long past predictions. The formular of aged weighted average is as follows:

$$\omega_i = \frac{\lambda^{i-1}(1-\lambda)}{1-\lambda^n}$$

Where ω_i is the weight of ith data, n is the total number of the data, λ is the rate of decay, and is usually chosen as 0.99, 0.98, 0.97, or 0.96. The smaller the λ is, the more important the recent data is.

Then depending on the period state prediction, winners and losers will be traded.

If the prediction is bull market, then momentum strategy will be applied, which is long past winners and short past losers;

If the prediction is bear market, then reversal strategy will be applied, which is short past winners and long past losers;

If the prediction is fluctuation, then no trade will be done during this period.

This process will repeat during the whole back-test period, and the portfolio is equal weighted, which will re-balance every K (holding period) trading day. The portfolio's performance based on each model is shown as below (Figure 11). In this special case (J=60, K=15), portfolios based on SVM and Decision Tree outperform the benchmark (S&P500), while others are worse than it. Later on, this project will back-test multiple different combination of J and K, to examinate the strategy's stability and performance.

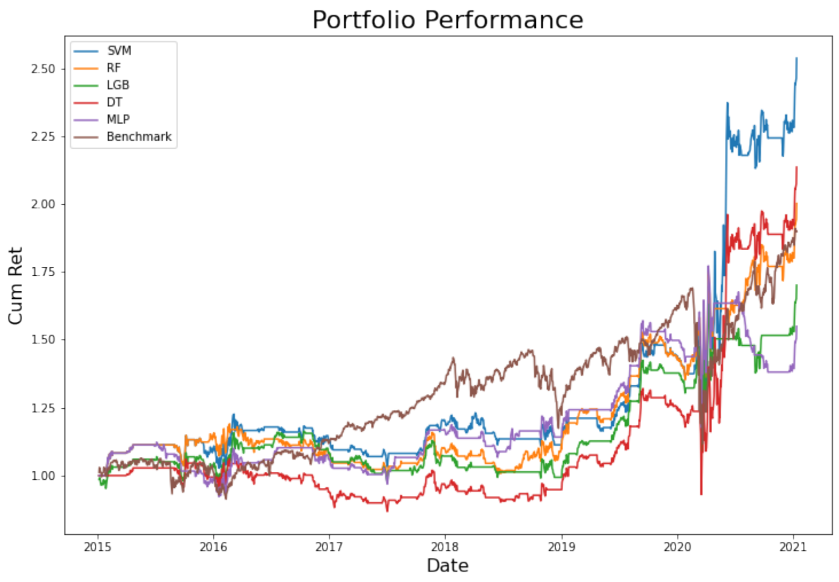
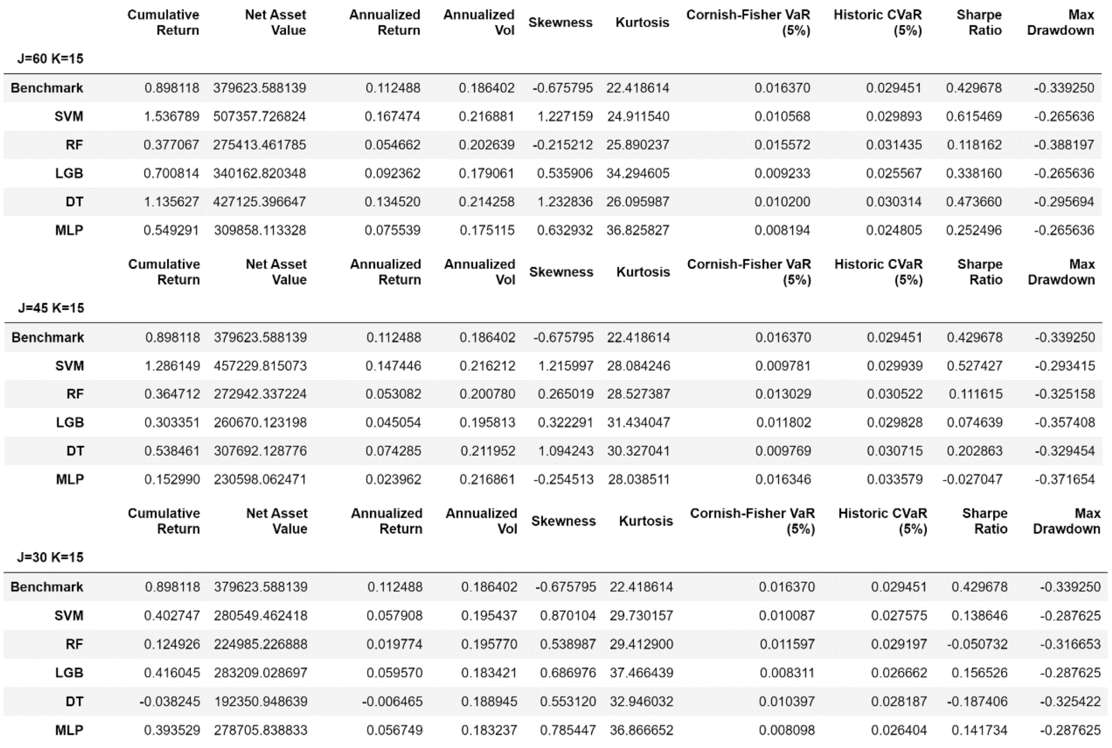


Figure 11. Portfolio Performance (J = 60, K = 15)

Here are the strategy results for using different J (Observation period)-K (Holding period) pairs. The following evaluation metric contains not only the portfolio's return, but also risk related factors, such as VaR, and max drawdown.

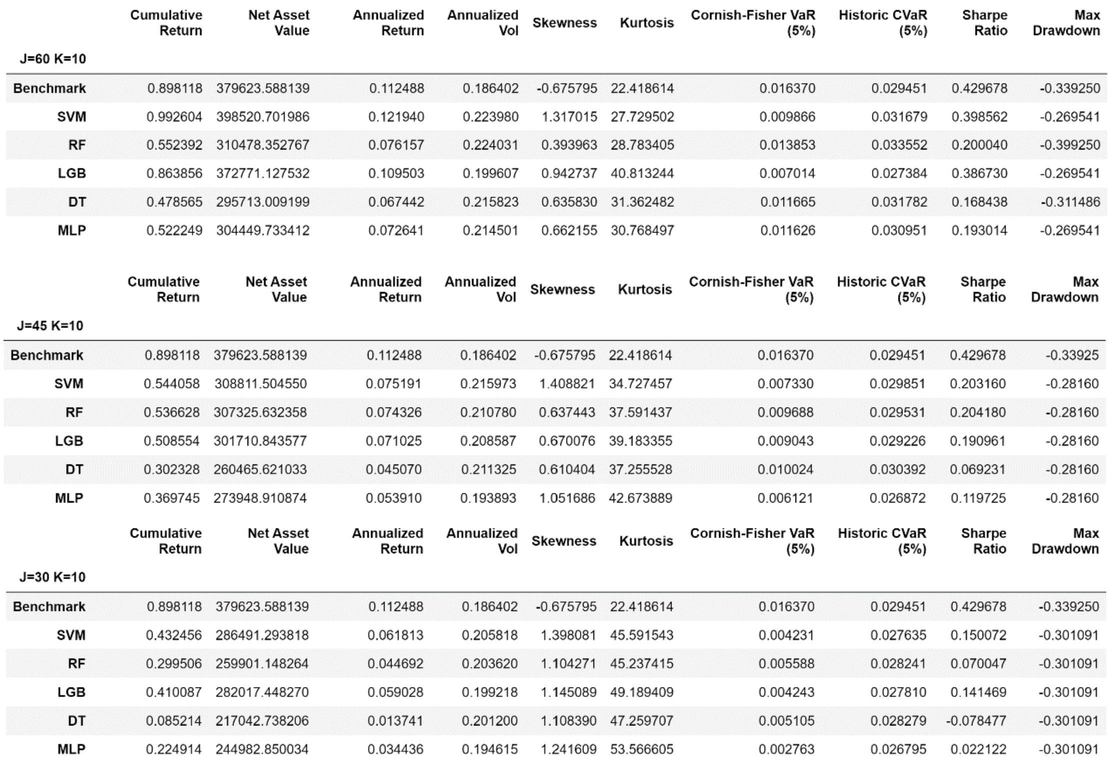


J=60 K=15	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.339250
SVM	1.536789	507357.726824	0.167474	0.216881	1.227159	24.911540	0.010568	0.029893	0.615469	-0.265636
RF	0.377067	275413.461785	0.054662	0.202639	-0.215212	25.890237	0.015572	0.031435	0.118162	-0.388197
LGB	0.700814	340162.820348	0.092362	0.179061	0.535906	34.294605	0.009233	0.025567	0.338160	-0.265636
DT	1.135627	427125.396647	0.134520	0.214258	1.232836	26.095987	0.010200	0.030314	0.473660	-0.295694
MLP	0.549291	309858.113328	0.075539	0.175115	0.632932	36.825827	0.008194	0.024805	0.252496	-0.265636

J=45 K=15	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.339250
SVM	1.286149	457229.815073	0.147446	0.216212	1.215997	28.084246	0.009781	0.029939	0.527427	-0.293415
RF	0.364712	272942.337224	0.053082	0.200780	0.265019	28.527387	0.013029	0.030522	0.111615	-0.325158
LGB	0.303351	260670.123198	0.045054	0.195813	0.322291	31.434047	0.011802	0.029828	0.074639	-0.357408
DT	0.538461	307692.128776	0.074285	0.211952	1.094243	30.327041	0.009769	0.030715	0.202863	-0.329454
MLP	0.152990	230598.062471	0.023962	0.216861	-0.254513	28.038511	0.016346	0.033579	-0.027047	-0.371654

J=30 K=15	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.339250
SVM	0.402747	280549.462418	0.057908	0.195437	0.870104	29.730157	0.010087	0.027575	0.138646	-0.287625
RF	0.124926	224985.226888	0.019774	0.195770	0.538987	29.412900	0.011597	0.029197	-0.050732	-0.316653
LGB	0.416045	283209.028697	0.059570	0.183421	0.686976	37.466439	0.008311	0.026662	0.156526	-0.287625
DT	-0.038245	192350.948639	-0.006465	0.188945	0.553120	32.946032	0.010397	0.028187	-0.187406	-0.325422
MLP	0.393529	278705.838833	0.056749	0.183237	0.785447	36.866652	0.008098	0.026404	0.141734	-0.287625

Figure 12. Portfolio Evaluation Matrix (J = 60, 45, 30, K = 15)

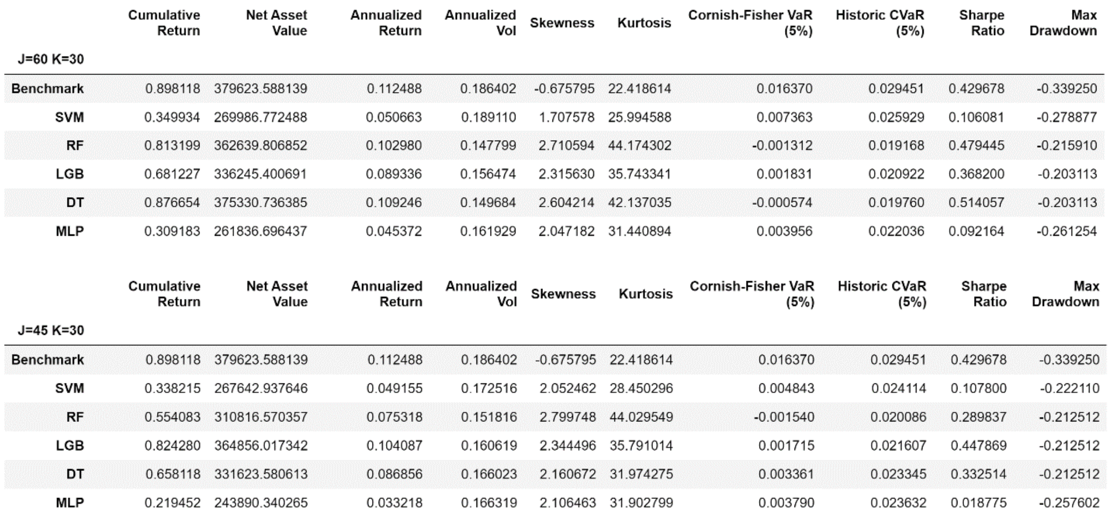


J=60 K=10	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.339250
SVM	0.992604	398520.701986	0.121940	0.223980	1.317015	27.729502	0.009866	0.031679	0.398562	-0.269541
RF	0.552392	310478.352767	0.076157	0.224031	0.393963	28.783405	0.013853	0.033552	0.200040	-0.399250
LGB	0.863856	372771.127532	0.109503	0.199607	0.942737	40.813244	0.007014	0.027384	0.386730	-0.269541
DT	0.478565	295713.009199	0.067442	0.215823	0.635830	31.362482	0.011665	0.031782	0.168438	-0.311486
MLP	0.522249	304449.733412	0.072641	0.214501	0.662155	30.768497	0.011626	0.030951	0.193014	-0.269541

J=45 K=10	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.33925
SVM	0.544058	308811.504550	0.075191	0.215973	1.408821	34.727457	0.007330	0.029851	0.203160	-0.28160
RF	0.536628	307325.632358	0.074326	0.210780	0.637443	37.591437	0.009688	0.029531	0.204180	-0.28160
LGB	0.508554	301710.843577	0.071025	0.208587	0.670076	39.183355	0.009043	0.029226	0.190961	-0.28160
DT	0.302328	260465.621033	0.045070	0.211325	0.610404	37.255528	0.010024	0.030392	0.069231	-0.28160
MLP	0.369745	273948.910874	0.053910	0.193893	1.051686	42.673889	0.006121	0.026872	0.119725	-0.28160

J=30 K=10	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.339250
SVM	0.432456	286491.293818	0.061813	0.205818	1.398081	45.591543	0.004231	0.027635	0.150072	-0.301091
RF	0.299506	259901.148264	0.044692	0.203620	1.104271	45.237415	0.005588	0.028241	0.070047	-0.301091
LGB	0.410087	282017.448270	0.059028	0.199218	1.145089	49.189409	0.004243	0.027810	0.141469	-0.301091
DT	0.085214	217042.738206	0.013741	0.201200	1.108390	47.259707	0.005105	0.028279	-0.078477	-0.301091
MLP	0.224914	244982.850034	0.034436	0.194615	1.241609	53.566605	0.002763	0.026795	0.022122	-0.301091

Figure 13. Portfolio Evaluation Matrix (J = 60, 45, 30, K = 10)



J=60 K=30	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.339250
SVM	0.349934	269986.772488	0.050663	0.189110	1.707578	25.994588	0.007363	0.025929	0.106081	-0.278877
RF	0.813199	362639.806852	0.102980	0.147799	2.710594	44.174302	-0.001312	0.019168	0.479445	-0.215910
LGB	0.681227	336245.400691	0.089336	0.156474	2.315630	35.743341	0.001831	0.020922	0.368200	-0.203113
DT	0.876654	375330.736385	0.109246	0.149684	2.604214	42.137035	-0.000574	0.019760	0.514057	-0.203113
MLP	0.309183	261836.696437	0.045372	0.161929	2.047182	31.440894	0.003956	0.022036	0.092164	-0.261254

J=45 K=30	Cumulative Return	Net Asset Value	Annualized Return	Annualized Vol	Skewness	Kurtosis	Cornish-Fisher VaR (5%)	Historic CVaR (5%)	Sharpe Ratio	Max Drawdown
Benchmark	0.898118	379623.588139	0.112488	0.186402	-0.675795	22.418614	0.016370	0.029451	0.429678	-0.339250
SVM	0.338215	267642.937646	0.049155	0.172516	2.052462	28.450296	0.004843	0.024114	0.107800	-0.222110
RF	0.554083	310816.570357	0.075318	0.151816	2.799748	44.029549	-0.001540	0.020086	0.289837	-0.212512
LGB	0.824280	364856.017342	0.104087	0.160619	2.344496	35.791014	0.001715	0.021607	0.447869	-0.212512
DT	0.658118	331623.580613	0.086856	0.166023	2.160672	31.974275	0.003361	0.023345	0.332514	-0.212512
MLP	0.219452	243890.340265	0.033218	0.166319	2.106463	31.902799	0.003790	0.023632	0.018775	-0.257602

Figure 14. Portfolio Evaluation Matrix (J = 60, 45, K = 30)

8. Conclusion

From testing different JK pairs, we can observe that in general our strategy performs better when the observation period is relatively long comparing with the holding period, while usually underperform the benchmark for the holding period is long. Among these models, SVM is more stable and has higher annualized return and Sharpe ratio. Also, SVM has low risk comparing with other models, because the max drawdown and VaR are lower. For all the combinations of different JK pairs and models, the best one is using SVM and J=60, K=15. This strategy has the highest annualized return and Sharpe ratio, and the risk metrics are also lower than others. Because SVM is able to overcome the overfitting problems, it can do best in this case, while other complex models like Boosting and ANN can only fit training data very well, but cannot predict the test dataset, which meets the overfitting problem. With proper JK pairs, RF and DT can also beat the market, or the portfolio's return is slightly lower than the benchmark, but the max drawdown and VaR are lower.

Here are still some limitations for this strategy. Firstly, this project only tested a few pairs of JK, while the selection of observation and holding periods could be investigated more carefully. Secondly, for different time periods, this strategy performs differently, and when the market is quite fluctuating, this strategy is even worse than the benchmark. Only when the market has significant trends, this strategy performs better. And when it applies the momentum strategy, the leverage is quite high, because it will long the top stocks and also short the bottom stocks. Thirdly, the universe should be the historical S&P500 components during the test period, but it is the current one. Therefore, there is survival bias in this case. Lastly, this project only uses technical factors, while in real world, fundamental factors are also pretty important. If can get more fundamental data, the prediction accuracy will probably be better.

References

[1] Lunde, A. and Timmermann, A., Duration dependence in stock prices: An analysis of bull and bear markets. J. Busi. Econ. Stat., 2004, 22, 253-273.
[2] Rajashree Dash, Pradipta Kishore Dash, A hybrid stock trading framework integrating technical analysis with machine learning techniques. The Journal of Finance and Data Science 2 (2016) 42-57.
[3] Zhixi Li, Vincent Tam, A Machine Learning View on Momentum and Reversal Trading. Algorithms.
[4] Xingyi Li & Valeriy Zakamulin (2020) Stock volatility predictability in bull and bear markets. Quantitative Finance, 20:7, 1149-1167.
[5] Dongdong Lv, Shuhan Yuan, Meizi Li, and Yang Xiang, An Empirical Study of Machine Learning Algorithms for Stock Daily Trading Strategy. Mathematical Problems in Engineering Volume 2019.
[6] Xiao Zhong and David Enke, Predicting the daily return direction of the stock market using hybrid machine learning algorithms. Financial Innovation (2019) 5:24.
[7] Jennifer Conrad1 and M. Deniz Yavuz, Momentum and Reversal: Does What Goes Up Always Come Down? Review of Finance, 2017, 555–581.