MACHINE LEARNING APPLICATION IN REGIME SWITCHING

Group 5

Khor WeiSheng
Li Jiahang
Pan Haolin
Zhao Yudong
Zhu Zhizhong

Abstract

Project Introduction

In this project, we are trying to use machine learning models to predict market regime, or market movement of US S&P 500 index where market regime is market movement of up, down or middle, in total of 3 classes.
The point of predicting market regime is for investors to adjust trading strategy accordingly and maximize return. And the predicted market movement can also be considered as a feature in other related analysis, such as predicting particular stock movement or predicting index return.

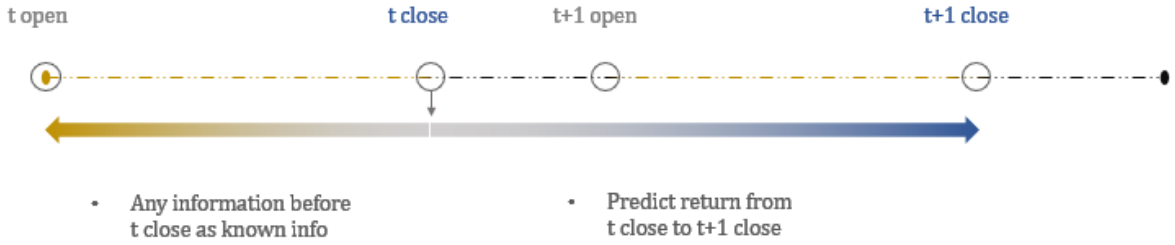

Figure 1 Information Set and Prediction Framework

This problem is a classification problem, and above picture is a visualization of our predicting process. The output(y) to our model is market regime of up, down and middle, which is relevant to return from t-close to t+1 close. The input(X) in our model are relevant financial data or engineered features observable in market before the market close at time t which is the time we make trading decisions.
This problem is special from many other classification problems in that

1) Our class label(y) is self-defined while in most other cases, it's given.
2) Instead of using only classification score to evaluate the model, we can also look at portfolio returns utilizing on prediction.

In other researches, researchers have used HMM and LSTM to predict market movements. But in most cases, they are looking at market positive return and negative return. But we choose to look at 3 class for model stability in that index returns clustered at middle, i.e. +0.1%, a lot, while this does not give much guidance to traders. Therefore, in our model validation, we will not trade direction for market regimes fell into middle class. And another benefit of this method is that middle class can act as safety cushion for wrong predictions within up and down groups when they are not too far away.

Analyzing Process

In our report, we start from collecting data of various asset classes which could affect S&P500 in part1. Then we conducted feature engineering to extract more relevant features including trading indicators, n-day returns and etc in part2. Following on, we looked at various definitions of y, found the best y and selected features(X) to build our model in part3 and finally tested various machine learning models in part4. In part5, we concluded our main findings and pointed out potential improvements.

Part1 Data Collection and Cleaning

Data Collection

We collected data of major asset classes, including equity, bonds(yields), FX, commodity and reference indices such as VIX and Fama-French investment factors.
In equity market, besides our analysis target S&P500 index, we find it may be useful to also look at foreign market index, we selected actively traded market indices from Asia market: Japan Nikkei225 and HongKong HangSeng Index, and Euro market: FTSE100 index. History financial analysis has pointed out that futures can provide some guidance of future market movement, therefore we included S&P500 Futures.
In bond market, we obtained actively traded US treasury bonds of different tenor. In order to reference FX, we choose to look at Dollar index. In commodity market, we obtained major products from agriculture, energy, metals and precious metals.
Detailed information are presented in below table.

			open high low close	volume	begin time
Equity	SP500 Futures		√	√	1997/9/10
	SPY		√	√	1993/1/29
	JP Nikkei225(N225)		√	√	1965/1/5
	UK FTSE100		√	√	2001/1/3
	HK Hang Seng(HSI)		√	√	1986/12/31
Yield	2Y Treasury Yield		√		1988/2/26
	10Y Treasury Yield		√		1988/2/27
	30Y Treasury Yield		√		1988/2/28
FX	Dollar Index		√		1990/1/1
Commodity	Agriculture	Coffee, Soybean and so on. 17 in total	√	√	1979-2000
	Energy	Oil, Natural Gas and so on. 7 in total	√	√	
	Industrial Metals	Copper	√	√	
	Precious Metals	Gold, Silver	√	√	
Others	VIX		√		1990/1/3
	FF5	Mkt-RF,SMB,HML,RMW,CMA			1963/7/1

We obtained above data form WRDS, Yahoo finance and investing.com.

Data Timeline

Then next step is to clarify the timeline of the data to make sure that we don't have look ahead bias. From a practical perspective, after doing the prediction for t+1, we need to make the trading decision before market close at time t. Any data after US stock market close at day t is not usable. Fortunately, both Asian market and European market close before American market close, so before US market close at time t, we already have all the data of Asian market and open price from European market for time t, so they can be used directly.
One key assumption we made in using data is that within US market, we took US close price on day t as known information when we made trading decisions at time t, which is slightly before close, i.e. 5 minutes before close. But this approximation is justified in that during the last several minutes of trading day, the market normally do not have large movements, therefore in real trading, we can take the current price as close price to compute feature engineering as well as do the prediction. It's optimal for us to also use the price 5 minutes before trading close to reflect the reality, however, due to data limitation, we could only obtain daily ticks.

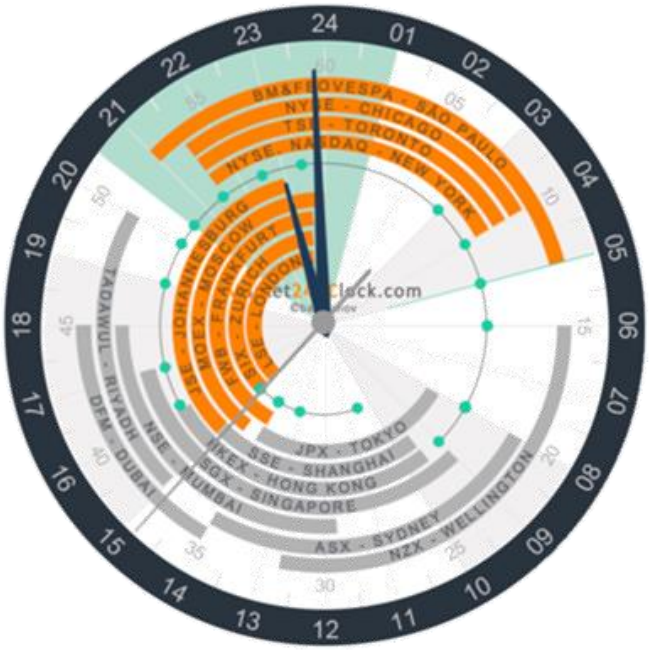

Figure 2 US Market Trading Time is Not Covered by Asian Market

Data Cleaning

In building model, we focused on data from last 20 years to reduce the impact form market behavior changes since trading techniques has developed a lot compared with 20 years ago due to advances in computational power and information availability.
Data Cleaning is composed of below steps.

- Convert all data to number
- Check the completeness of data, if there is too much NA, then drop that attribute.
- For the rest, forward fillna to avoid look ahead bias;
- Use the index of Y as basis, shift down X for 1 day and merge with Y;

- To keep the length for all X and Y to be same, cut data from 1998-01-01 to 2020-06-19. We kept some additional data apart from the last 20 years for computing feature engineering.

	Price_Yield2y	Open_Yield2y	High_Yield2y	Low_Yield2y	Change_Yield2y	...	High_VIX	Low_VIX	Change_VIX	Mkt-RF	SMB	HML	RMW	CMA	RF	Y
2020-06-17	0.197	0.199	0.205	0.187	0.0212	...	37.45	31.73	-0.0212	-1.18	-2.06	-1.63	0.71	-0.10	0.000	-0.003600
2020-06-18	0.195	0.205	0.205	0.193	-0.0096	...	35.17	32.25	-0.0059	-1.18	-2.06	-1.63	0.71	-0.10	0.000	0.000594
2020-06-19	0.195	0.195	0.199	0.189	0.0000	...	36.25	32.24	-0.0158	-1.18	-2.06	-1.63	0.71	-0.10	0.000	-0.005649

5653 rows × 216 columns

Finally, we have a data table as above, there are in total 5653 rows and 216 columns data. The rightmost column is Y, and the others are X.

Part2 Feature Engineering

The purpose of feature engineering is to construct features for model based on the raw data. There are three methods to do this.
Firstly, we used statistical measures. We calculated three statistical measures such as return, volatility and spread. We used 1 day, 5 days, 10 days, 30 days and 60 days return and we considered 20 days standard deviation of the return as volatility. We also calculated open to close spread, low to high spread and cross asset price spread as well.
Secondly, we used technical indicators. We calculated SMA(simple moving average) and CCI(commodity channel index) for measuring market trend, ATR(average true range) for measuring volatility and RSI(relative strength index) for measuring market reversal. Additionally, we also created dummy variables which are used to indicate the status of these technical indicators and the change of the status.
Lastly, we used PCA to downsize the features we created to 20 principle components. Unlike usual PCA using the whole data to do so, at this part, in order to avoid look ahead bias, we used rolling covariance matrix to calculate rolling PCA and set the values of 20 principle components at the last day as the features of that day. After that, we confirmed that 20 principle components can represent 99% information of all features by looking at the rolling explained variance.

Part3 Features and Predicted Classes

Appreciate the effort from previous part, we do not need to worry the time period overlap and data information richness. We have basic daily pricing data, various technical indicator and advanced engineering features (PCA).

Features(X)

For the independent variable (features), se consider few asset classes, i.e. Bonds, Commodity, Dollar Index, Stock Index and Fama-French Factors. Well-known to the practitioners, information did persist in a short period of time, so we included features mentioned above lag up to 3 days since this is a predictive model.

Product of Asset Class:

Bonds	: 2y Treasury yield, 10y Treasury yield, 30y Treasury yield
Commodity	: Gold, Silver, Crude Oil WTI, Natural Gas, US_Corn, US_wheat
Stock Index	: Hang Seng, Nikki225, FTSE100, VIX Index, S&P500 Futures
Fama-French[1]	: SMB, HML, RMW, SMA (no feature engineering applied)
Others	: US Dollar Index, RF

Feature engineering of each of the product:

(High-Low) / Open	Volume	% Change	ATR	LH_spread
Volatility	CCI	OC_spread		

Total potential features = 3 Bonds + 6 Commodity + 5 Stock Index + 2 Others =16
Total features from each available product[2] = 8
Potential usable features = (16 * 8 * + 4) * 3 = 396
After dropping invalid data and perform fulfill, we have total 324 features to train our model. Most of the dropped features are volume related as volume of some stock index is not available in the early 2000s.

Predicted Class(y)

In the project, we want to build a model to study the market situation in near future. In order to have clearer future market direction, we want to have 3 class, i.e. Up, Mid and Down.
Class definition:

Up	: We expect a reasonable profit when we go long a position for one day.
Mid	: We expect a small movement in the stock market and there will have no sufficient to profit in either long or short position for 1 day holding period.
Down	: We expect a reasonable profit when we go short a position for one day.

[1] Classical Fama-French 5 factors to explain market return. Detail explanation can be found at https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/Data_Library/f-f_5_factors_2x3.html
[2] Refers to Feature engineering of each of the product.

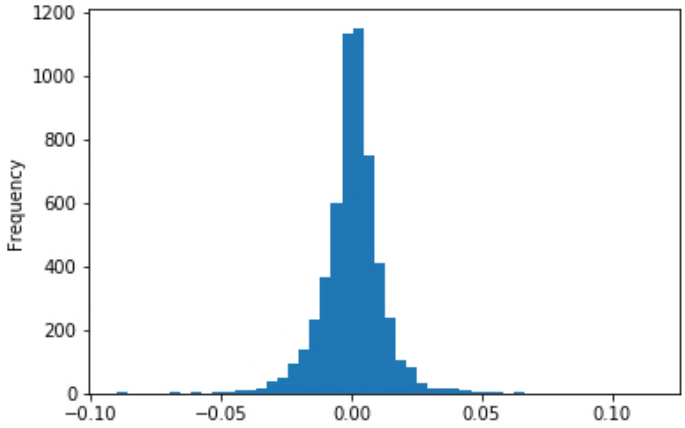

Figure 3 Return Distributions

Standard statistical separation

First, we apply standard statistical separation, where:

$$\text{Up: } r_t > \mu + 1.65 \times \sigma;$$
$$\text{Down: } r_t < \mu - 1.65 \times \sigma;$$
$$\text{Mid: Otherwise.}$$

Then, we discover serious data imbalance where each of class up and class down have only around 200 sample point out of around 5000 sample (20-year daily data). Not to mention the model can't capture the class as few sample sizes, it is unreasonable to classify as above because the market is dynamic. Taking 20 years data and treat them as equal cannot different risk-return in stable market and risk-return in fluctuate market.

Equal Separation

Second, simply separate them equally where number of samples in class up, class mid and class down are same. The drawback here is that class mid has very narrow range which is bound at (-0.2%, 0.2%). We argue that a return should be at least cover commission, spread and other costs, therefore, it is not a favorable method to define the class by this method. Also, this method does not take the dynamic of market into account.

Dynamic Statistical Separation

Lastly, we want to ensure the dynamic classification which capture the market dynamic and also make statistical sense. We propose a rolling volatility method. It is setup as follow:

Lookback period = 1 year

$$\mu^* = 1\ year\ mean, \sigma^* = 1\ year\ mean$$
$$\text{Up: } r_t > \mu^* + 1 \times \sigma^*;$$
$$\text{Down: } r_t < \mu^* + 1 \times \sigma^*;$$
$$\text{Mid: Otherwise.}$$

This method classifies the market situation by lookback past 1-year market return and its standard deviation. The advantage is that it captures recent market volatility as stable market tend to have stable return (typical lower) and volatile market typical generate abnormal return (typical higher). This also better explain investor risk averse in predicted return (higher risk requires higher return). Data imbalance remain exist

but much better for model training, both class up and class down has around 15% of the total sample size.

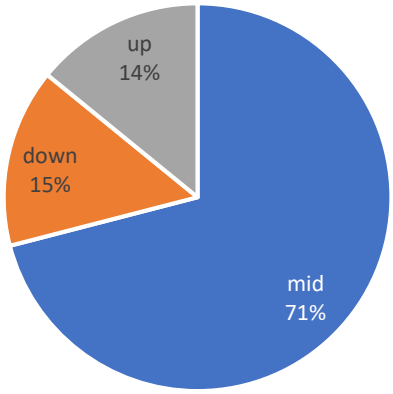

Figure 4 Class Distribution for Rolling Volatility

We will proceed the model training with this method as it is best method among three.

Part4 Model training

We employ Decision Tree, MLP, Logistic, Boosting, Stacking to train and test.
To make comparison, we have the setting as below:

Train_test split(test_size = 0.2, random state = 0)

As we want the model to correctly classify positive or negative (precision > recall) as we want to maximize our return by investing capital based on the predicted result. We go to long the market if class up and short in class down. Also, we want the model to have good F_β score (Macro) in each classification rather than weighted F_β or Micro F_β because all the market situations are equally important for us to invest or to position ourselves. We tune the parameter of model as below:

Scoring = fbeta(beta = 0.5, average = Macro)

Decision Tree Model:

We fitted decision tree classifier with 298 features and no hyperparameters tuning to establish baseline.

	Precision	Recall	F1	F0.5	ROC AUC
DecisionTree_Base	51%	52%	51%	51%	75%

In order to look at most important features, we selected features according to importance. The most important products relevant to S&P500 are: VIX, bonds, Crude Oil, Foreign Index

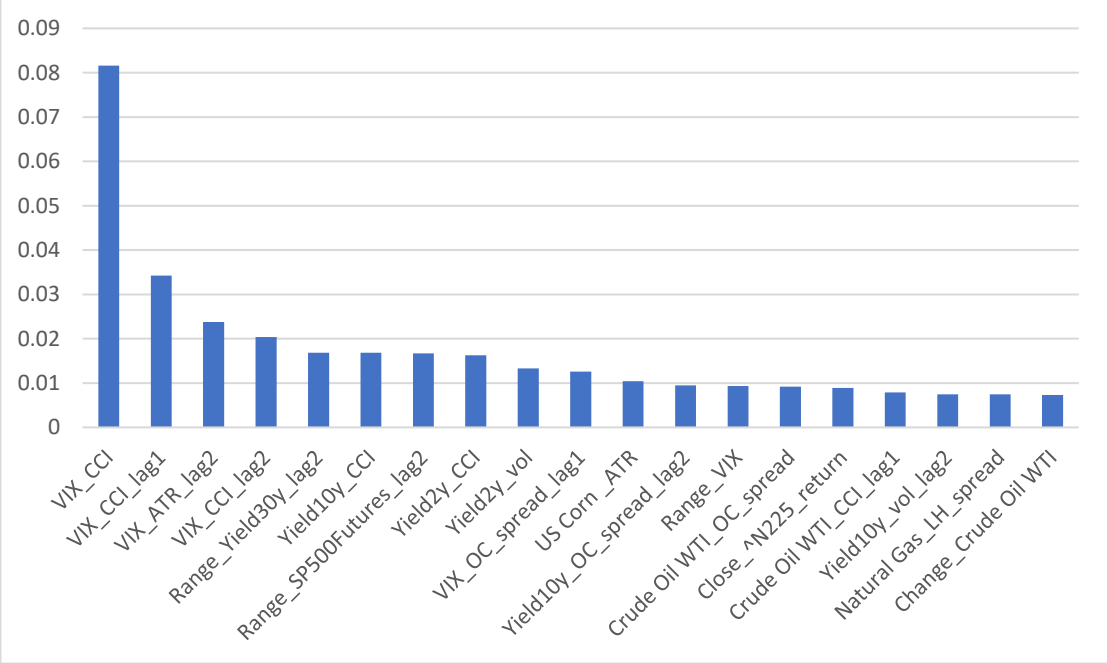

Figure 5 Feature Importance from DecisionTree Model

Grid Search on Decision Tree

Decision tree model is highly susceptible to overfitting training data without parameter tuning. In order to find the best decision tree model, we used 10-fold cross validation to reduce over fitting, grid searched on splitter, max depth and min sample split to find the best combination for the model and tried different feature numbers.

	Precision	Recall	F1	F0.5	ROC AUC
Full Feature	60.7%	49.3%	51.2%	54.9%	61.7%
Top 50 Feature	60.6%	49.3%	51.2%	54.7%	61.7%
Top 20 Feature	61.3%	50.8%	53.1%	56.6%	62.7%

The best decision tree model we have is a tree with max depth of 6 and min sample split of 5 and it works best under 20 features.

Comparison With Other Simple Models

Similarly, we also built models using other models and obtained below results.

Full + PCA : Include the available feature[3] + feature generated from PCA
Full : Include the available feature[4]
Top 50 : Top 50 import feature suggested by DecisionTreeClassifier.
Top 20 : Top 20 import feature suggested by DecisionTreeClassifier.

[3] Features mentioned in Section Features
[4] Features mentioned in Section Features

Test Sample F 0.5 Score (Macro)

	Top 20	Top 50	Full	Full + PCA
MPL	62.9%	62.9%	26.2%	26.3%
Logistic	67.5%	68.0%	35.7%	34.3%
AdaBoost - Log	67.0%	67.0%	60.8%	58.6%
AdaBoost - Tree	64.5%	64.2%	65.3%	63.7%

Analysis on Ensemble Models

Moving on, we tested the model under framework of ensemble models.
The 3 different ensemble methods have different characteristics:

- Bagging gives variance reduction effect by increasing sample size through resampling.
- Boosting methods gives bias reduction effect through putting more weight on wrongly classified samples.
- Stacking combines benefits from each stacked model.

Below table presents bagging and boosting model results.

	Bagging	RandomForest	AdaBoost	GradientBoost
Precision	62.6%	72.0%	58.9%	69.8%
Recall	60.7%	60.7%	66.2%	66.8%
F1	60.3%	63.3%	61.6%	67.6%
F0.5	61.2%	67.1%	59.8%	68.7%
ROC AUC	69.6%	69.9%	73.2%	74.3%

Both of the 2 ensemble models out-perform single decision tree models in F0.5 and ROC AUC score. It means that these models did improve classification power in variance or bias as compared with single ones.
By looking at ROC AUC scores, boosting methods on average out-perform bagging methods in that boosting methods gives better classification on edges. It's because our data is highly sensitive to edges, a single change in one feature may indicate huge change in predicted market return. Therefore, bagging did not work well in ROC AUC.

Following on bagging and boosting methods, we also tried stacking methods. We used decision tree, SVC and Gaussian Naïve Bayes as LV1 model and tested LV2 model using decision tree, Gaussian Naïve Bayes and Logistic model and further grid searched for best parameters.

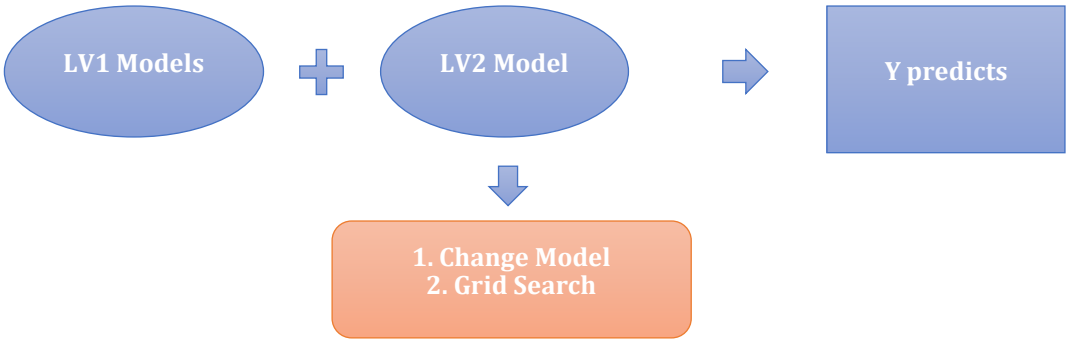

Figure 6 Working Flow of Stacking Model

	DecisionTree	NB	Logist
Precision	60.8%	59.5%	69.5%
Recall	72.8%	71.3%	52.2%
F1	63.6%	63.0%	53.7%
F0.5	61.4%	60.4%	57.8%
ROC AUC	77.8%	76.7%	63.7%

Our stacking model did not improve compared with bagging and boosting methods. Normally stacking can obtain improved performance from good performing models with edge in different aspects. However, in our case, it could be that the models we input did not have too much difference in predicted results.

AdaBoosting is best Ensemble Model from NAV

Furthermore, if we look into the 2 boosting methods, GradientBoosting gives higher score in both F0.5 and ROC AUC. But does it mean GradientBoosting has better classification power than AdaBoosting?
From perspective of investment, we are more sensitive to market dramatic changes, especially market surge. Therefore, it makes more sense to look at net asset value(NAV) of our portfolio. In the constructed portfolio, if the predicted market move is up, then we will long 1 unit of S&P500; if predict to be down, then we short 1 unit of S&P500; otherwise, we don't hold any S&P500 position.
Looking from prospective of NAV, if 2 classifier has same classification scores, the one with higher NAV should be the better classifier since it gives better prediction on dramatic market movements. Alternatively, this NAV method can also be replaced by building another classification model to predict extreme movements. Since NAV is an more intuitive approach, in our report, we go with NAV approach.

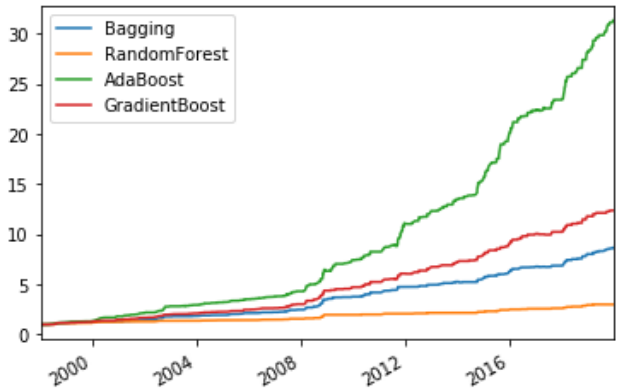

Figure 7 NAV of Ensemble Models

Above NAV figure shows that AdaBoosting out-perform GradientBoosting methods and bagging methods, suggesting AdaBoosting is the best prediction model out of the 3 ensemble models even tough it's not as good as GradientBoosting in F0.5 and AUC. The almost up-only moving trend in AdaBoosting suggests that it not only makes most of the predictions right, it also captures market dramatic moves where the other 3 models fail to achieve.

Resolving Concerns on Model Stability

In part4 we built our models using training data sliced from total data without specifying tenor. Therefore, the training set we obtained sprays within the whole 20 years from 2000 to 2020, representing the regime pattern for the 20 years. This method is well justified for now since our machine learning model is trying to find the general pattern across a long period of time, and assuming this pattern will still work in the future.

But, in real investment, we don't know whether market style would change, therefore we switch to look at results if we split data by time. And here we are also trying to understand whether our previous NAV comes mostly from utilizing future pattern information and the high NAV is not achievable in real trading execution.

Aiming to replicate the real investment environment, we tested the model by using the previous 80% of years as training data and test on the latest 20% years.

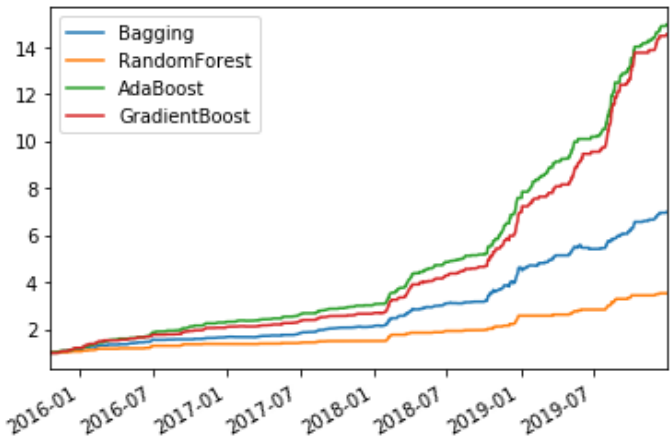

Figure 8 Time Series Testing NAV

	Bagging	RandomForest	AdaBoost	GradientBoost
Precision	53.9%	74.4%	62.4%	69.8%
Recall	55.8%	50.5%	64.4%	68.9%
F1	54.7%	53.8%	63.2%	69.2%
F0.5	54.2%	60.6%	62.7%	69.5%
ROC AUC	65.8%	62.5%	72.1%	75.9%

The result is as good as our result before. And the investment performance is moving monotonously up, indicating stable consistent predictions.
It's worthy to note that:

1. Even though total return during the testing period dropped from 3000% to 1400%, it does not mean a bad performance since total return is highly relying on market volatility. Since we changed testing period, the final best performance changes accordingly;
2. Boosting model still over-perform bagging models, boosting model made less significantly wrong predictions (up label predicted as down, or other way round). And captured most predictions right in the surge.

Part5 Conclusion:

In our report, we found that our problem is unique in that y is self-defined. By trying out several common market regime definitions, we notices that our model is sensitive to different kinds of market regime definitions and selected rolling-volatility market regime definition for building model for its economic validity.
The best model in our prediction is boosting in both scoring system and profitability system. In scoring system, the best model is Gradient Boosting model and in profitability system, the best one is AdaBoosting. Compared with other classifiers, the 2 boosting models give less significantly wrong classifications, thus gives decent monotony characteristic in profitability test.
In the test of trying different number of features, our model works good with around 20 features instead of all hundreds of features. By using only the most important features, we may have reduced noise we input into the models.

Potential Improvements

There are also some improvements from our report:

1. We tested limited features due to both computation power limitation and time constraints. It's worthy to exploit more relevant products and features to obtain better performance.
2. Our model is t day close- t+1 day close(t+1 trading) prediction. However, if we switch the trading framework to intraday, t day open-t day close(t+0 trading), and utilize foreign index intraday price, i.e. utilize FTSE100 index growth right before US stock market opens, we believe it may give some additional information and improve

classification score at the cost of missing the price change from t-1 close to t open.