Statistical Arbitrage on S&P 500 by Machine Learning Models

1. Introduction

1.1 Background

S&P 500 is one of the most followed equity indices which measures the stock performance of largest 500 companies listed on stock exchanges in US. The average annual total return of the index, including dividends, since inception in 1926 has been 9.8% (Wikipedia). From Berkshire Hathaway's annual meeting 2020, one of the most famous investors Warren Buffet indicated "In my view, for most people, the best thing is to do is owning the S&P 500 index fund". Besides directly investing on S&P 500 index, statistical arbitrage trading on composites of S&P 500 index is also a hot topic.

1.2 Literature Review

Krauss et al. (2017) developed a statistical arbitrage strategy based on deep neural networks (DNN), gradient-boosted trees (GBT), random forests (RAF), and their simple ensample and deployed it on the S&P 500 constituents from 1992 to 2015. The daily returns of 0.25 percent with annualized Sharpe ratio of 1.81 proves their methods performed better than general market. Using the same data as Krauss et al. (2017), Fischer and Krauss (2018) deployed long short-term memory (LSTM) networks for predicting out-of-sample directional movements for the constituent stocks of the S&P 500. Their daily returns of 0.46 percent and a Sharpe Ratio of 5.8 prior to transaction costs showed LSTM outperformed DNN, GBT and RAF. Fischer et al. (2019) performed a statistical arbitrage strategy based on logistic regression (LG) and RAF with 40 cryptocurrency coins on minute-binned data from 18 June 2018 to 17 September 2018. They achieved an excess return 0.038%/round-trip trade after transaction cost with 0.15%/half-turn.

1.3 Goal

This project will compare different machine learning methods in S&P 500 trading with an extension data (start from year 2006 to the end of year 2019) different from Krauss et al. (2017). We also use grid-search to tune the parameters in machine learning methods. The goal is to compare the recent results of machine learning methods to the previous results and check the hypothesis that “profits are declining in recent years and there is a severe challenge to the semi-strong form of market efficiency”.

2. Datasets Extract, Transform and Load (ETL)

2.1 List of historical S&P 500 components

Download from Wharton Research Data Services (WRDS) and Wikipedia, clean them and get a mapping table indicates which stock will be included in S&P 500 at the end of the specific time.

2.2 Daily data of historical S&P 500 components

Based on the components from section 2.1, use Python yahoo finance module yahoo-finance-1.0.4 to download daily data of all components and S&P 500 index from 2005/01/01 to 2020/01/01 (15 years data). Since yahoo finance will not keep the delisted stock’s information, several delisted stock’s data in historical S&P 500 will not be available. This will not affect the prediction too much because most of the stock’s information will be arrived.

3. Exploration Data Analysis (EDA)

3.1 Data split for formation and trading

Like Krauss (2017), we set the length of the in-sample formation window to 750 days (approximately three years) and the length of the subsequent out-of-sample trading window to 250 days (approximately one year). We move the formation-trading set forward by 250 days in a sliding-window approach, resulting in 12 non-overlapping trading batches (Table 1) to loop over our entire data set from begin of year 2005 until end of 2019.

Subset	form_start	form_start	trad_end	Subset	form_start	form_start	trad_end
0	1-1-2005	1-1-2008	1-1-2009	6	1-1-2011	1-1-2014	1-1-2015
1	1-1-2006	1-1-2009	1-1-2010	7	1-1-2012	1-1-2015	1-1-2016
2	1-1-2007	1-1-2010	1-1-2011	8	1-1-2013	1-1-2016	1-1-2017
3	1-1-2008	1-1-2011	1-1-2012	9	1-1-2014	1-1-2017	1-1-2018
4	1-1-2009	1-1-2012	1-1-2013	10	1-1-2015	1-1-2018	1-1-2019
5	1-1-2010	1-1-2013	1-1-2014	11	1-1-2016	1-1-2019	1-1-2020

Table 1: Formation and Trading periods for 12 non-overlapping trading batches

3.2 Data Cleaning

From list of historical S&P 500 components, there are 711 stocks, only 636 can be available from Yahoo Finance since delisting and information removed. I remove stocks with extremely large price (e.g. share price>5000) by considering the problems from Yahoo Finance database. The final whole dataset is 626 stocks with 3774 adjusted closed prices from the corresponding trading days. By splitting as table 1, number of stocks in each training subset (Figure 1) are increasing with the movement of the time period.

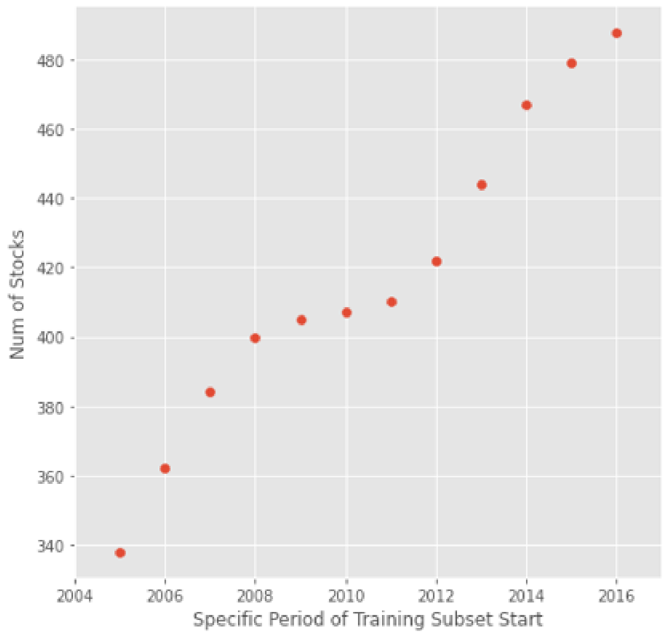


Figure 1. Number of stocks in different training subset

3.2 Data Description

Figure 2 displays the return for S&P 500 and randomly picked 5 stocks in whole period. Different stock performs differently, after taking the normality test, we conclude returns of stocks “SWN”, “SPG” and S&P 500 are not normally distributed, returns of stocks “FTV”, “LYB” and “NWSA” are normally distributed.

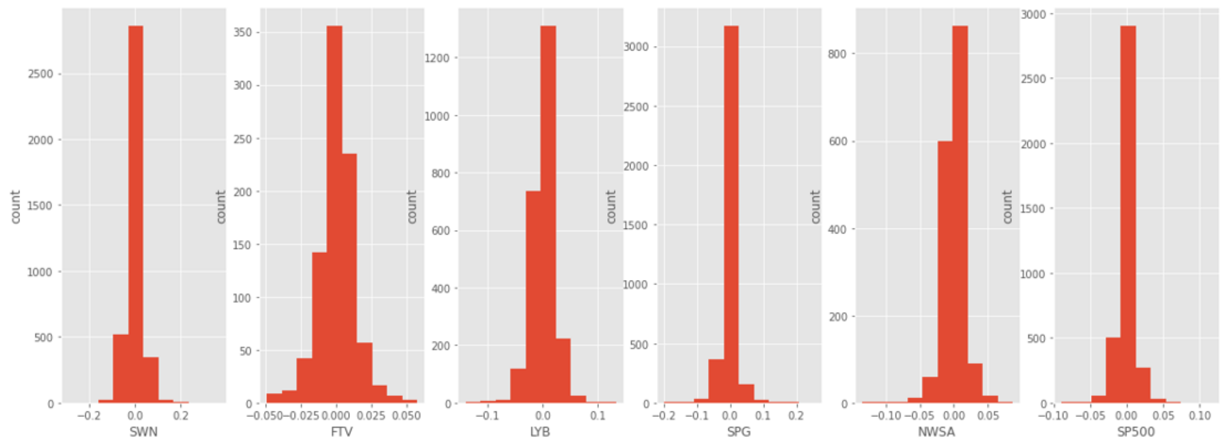


Figure 2. Returns from 5 randomly picked stocks and S&P 500

3.3 Feature generation

For each formation-trading set, the feature space (input) and the response variable (output) are created as follows:

Input:
Let $P^s = (P^s_t)_{t\in T}$ denote the price process of stock s or financial indicator, with $s \in \{1, \dots, n\}$.
Define the simple return $R^s_{t,m}$ for each stock or indicator over m periods as
$R^s_{t,m} = \frac{P^s_t}{P^s_{t-m}} - 1$
Where $m \in \{\{1, \dots, 20\} \cup \{40, 60, \dots, 240\}\}$

The notation of features is f_m, m is defined as above which represents returns of different lags of days.

Output:

Construct a binary response variable $Y^s_{t+j,j} \in \{0,1\}$ for each stock s. The response $Y^s_{t+j,j}$ equals to one (class 1), if the j-period return $R^s_{t+j,j}$ of stock s is larger than the corresponding cross-sectional median return computed over all stocks and zero otherwise (class 0). Here we can try different j(s). For example, if j=1 it means one-period return prediction. We try to forecast probability $\mathcal{P}^s_{t+j}$ for each stock s to outperform the cross-sectional median in period t+j. Please note the binary response also can be extended to multinomial response in the future work.

3.4 Feature Description

Since there are 12 subsets, I pick up one (subset 10 in formation set) as an example for EDA. From figure 3, although some adjacent (such as one period or two periods) features may correlated, I keep all features since this creates the same features as Krauss (2017) and Fischer and Krauss (2018). Some highly correlated features can be detected in figure 4.

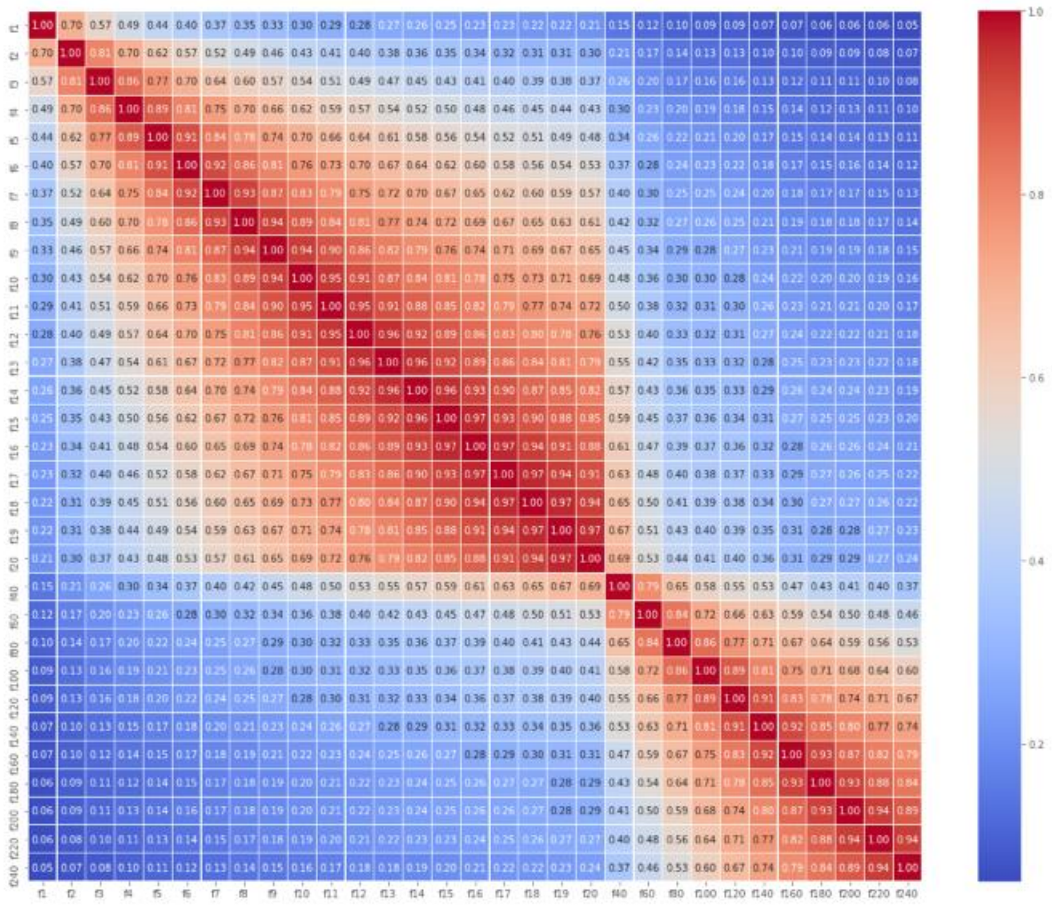
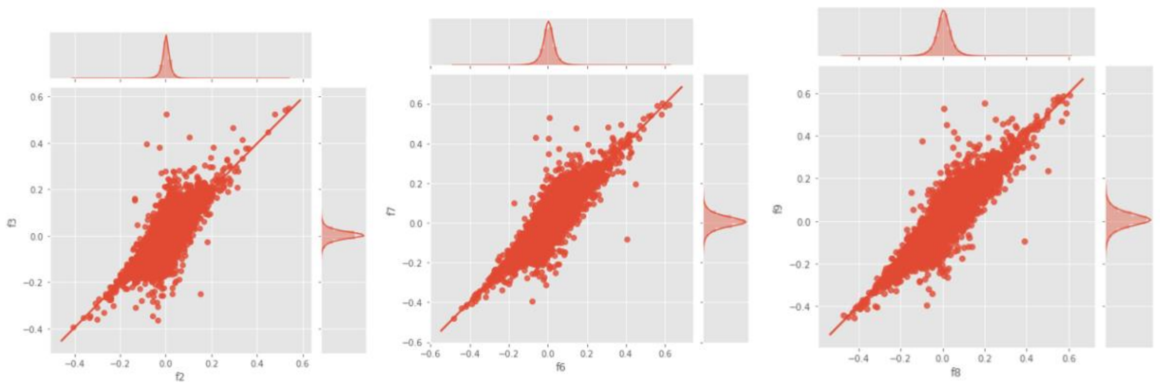


Figure 3. Feature Correlation Heatmap from Subset 10



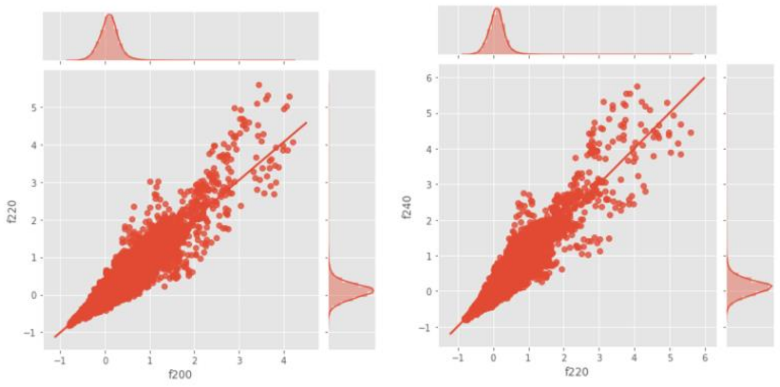


Figure 4. Highly Correlated Features from Formation Subset 10

4. Machine Learning models

4.1 Tuning Parameters

I used machine learning models – k-nearest neighbor (KNN), random forest (RAF), logistic regression (LG), XGBoost, ensemble (average) of KNN, RAF, LG and XGBoost, ensemble (stacking) of KNN-XGboost, multiple layer propagation (MLP) in classification process. Table 2 shows the selected parameters to be tuned in each method.

Methods	Parameters to be tuned	Be selected values
knn	Number of neighbors	{3,5,7,11}
	Weight function used in prediction	{'uniform','distance'}
	Metric	{'euclidean', 'manhattan'}
raf	The number of trees	{1500,2000,2500}
	The maximum depth of the tree	{20,25,30}
	The number of features for the best split	{log2(n_features), sqrt(n_features)}
	The number of samples to draw in bootstrap	{500, 700, 1000}
lg	Inverse of regularization strength	{10^-3, 10^-2,...,10^2,10^3}
	Penalization (learning rate)	{'l2'}
	Optimization solver	{'newton-cg','lbfgs','sag'}
	Maximum iterations	{10000,20000}
xgboost	Maximum depth of a tree	{3,5,7}
	Minimum sum of instance weight needed in a child	{5,10}
	Number of trees	{500, 1000}
	Penalization (learning rate)	{0.01, 0.02, 0.05, 0.1}
ensemble 1	Average of knn-xgboost	Average
ensemble 2	Stacking of knn-xgboost	Meta-function
mlp	structures, early stopping, activate function	31-10-5-1, 256-128-64-32-1

Table 2. The Selected Parameters to Be Tuned in Each Method

4.2 Results for Machine Learning

I split each formation subset into 80% as training set and 20% testing set. In each training set, I use 5 folder cross-validation to find the optimal parameters (table 3) with highest accuracy rate. Figure 5 displays the comparison of train and test accuracy for machine learning methods with optimal parameters in all subsets. For train subsets, we ignore knn method since train accuracy for knn is meaningless, the result shows stacking ensemble performs best. Besides stacking ensemble, for subset 0, 1, 3, 4 and 5, xgboost performs best, for the remaining subsets, average ensemble performs best. Overall, the range of best training accuracy is from 67.9% to 100%. For test subsets, stacking ensemble also performs best. Besides stacking ensemble, for subset 0, 1, 2, 3 and 4, xgboost performs best, for subset 7, average ensemble performs best, for the remaining subsets, knn performs best. MLP method doesn't perform well in all subsets. Overall, range of best testing accuracy is from 52.14% to 54.55% which is obviously higher than 50%, we can conclude that machine learning methods give us some effective prediction for one day stock's movement.

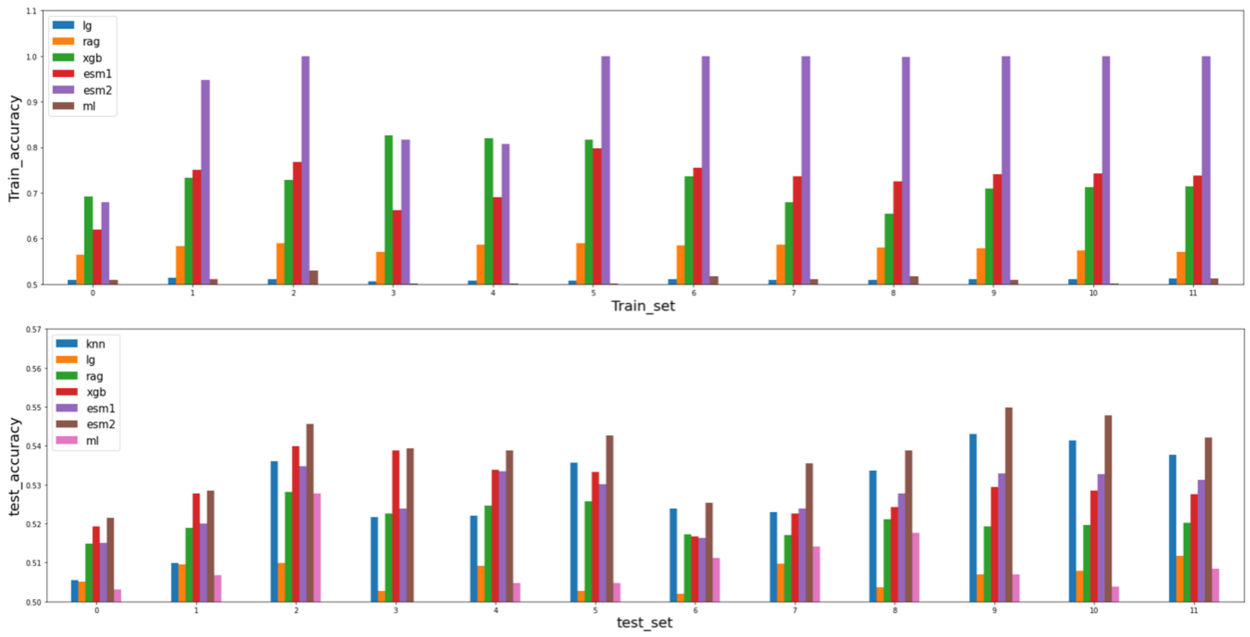


Figure 5. Train and Test Accuracy for machine learning methods with optimal parameters in all subsets

4.3 Optimal Selected Parameters

Table 3 shows the optimal values for all subsets in each method. These values are kept in the algorithm for trading part.

Methods	Optimal values for Dataset											
	0	1	2	3	4	5	6	7	8	9	10	11
knn	11							5	3		5	11
	'uniform'	'distance'		'uniform'		'distance'						
	'manhattan'	'euclidean'		'manhattan'		'euclidean'		'euclidean'				
raf	2500			2000	2500							
	20		30		25		30		25	20		
	"sqrt"	"log2"		"sqrt"		"log2"						
	700	1000		700	1000							
lg	0.01	0.1	0.001		1000	0.01	0.1	0.001			10	0.001
	l2'											
	"sag"	"newton-cg"	"lbfgs"	"newton-cg"	"lgfgs"	"newton-cg"	"sag"	"newton-cg"		"sag"	"lbfgs"	"newton-cg"
	1000											
xgboost	7											
	5								10		5	
	1000											
	0.01	0.02		0.05			0.02	0.01		0.02		
ensemble 1	Average output probabilities from knn, raf, lg and xgboost											
ensemble 2	Meta function with output probabilities from knn, raf, lg and xgboost											
mlp	256-128-64-32-1											

Table 3. The optimal values for all subsets in each method

5. Trading

5.1 Trading Methods

Momentum method was studied by Jegadeesh and Titman (1993) earlier in the trading process. The main idea is to buy the past winners and short the losers. I sort all stocks over the cross-section in descending order with the prediction probabilities which are reached by stacking ensemble method with the optimal parameters in table 3. Trading process is to long the highest ("expected winner") k probabilities and short the lowest ("expected loser") k probabilities to make profits.

5.2 Trading Result (K=3)

The comparison of cumulated daily trading returns to S&P 500 for 3 stocks/pair (figure 6) indicates in year 2008, 2009, 2011, 2015 and 2018, my method (stacking ensemble) outperformed the benchmark of S&P 500 without transaction cost. Beside year 2014, 2016 and 2017, my method achieved positive cumulated daily returns in other 9 years.

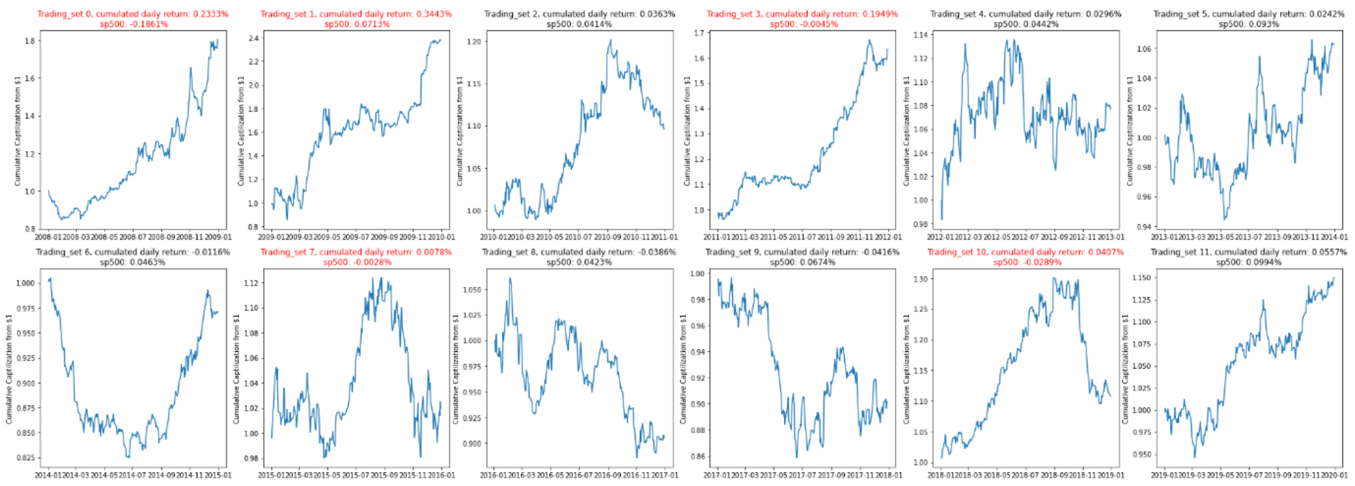


Figure 6. Cumulated Daily Trading Returns vs S&P 500 for 3 stocks/pair

5.3 More Trading Result (K=3, 5, 10, 15, 20)

To check if the number of stocks in each pair will affect the trading performance, I increase k from 3 to 5, 10, 15 and 20. In years 2008, 2009 and 2011, most of my method with different stocks in each pair performed better than S&P 500. In years 2013, 2014, 2016, 2017 and 2019, even the best performance of my method got less return than S&P 500. Comparing to Krauss et al. (2017), although they got 0.25% daily return, most of their returns are realized before year 2008 (especially from year 1993 to 2000). In year 2010 to Oct 2015, loss of their capitalization was over 50% with after taking the transaction cost. Although Fischer and Krauss (2018) used a LSTM model and got better results, they could not avoid negative daily return in year 2013, 2014 and 2015 with 10 selected stocks after transaction cost. Positive daily returns still could be achieved by my method (although I didn't take transaction cost). My result also confirms our hypothesis that "profits are declining in recent years and there is a severe challenge to the semi-strong form of market efficiency."

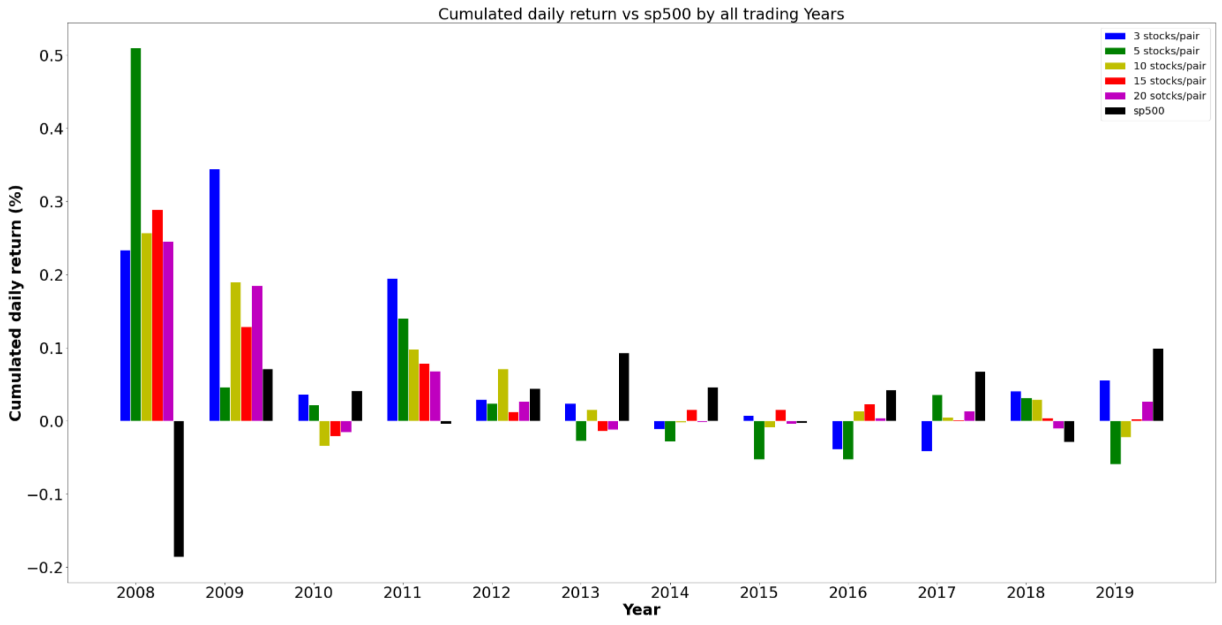


Figure 7. Result of Momentum with Stacking Ensemble by 3,5,10,15 and 20 stocks/pair

6. Future Work

There are lots of work can be considered in the future:

- Check the data quality. The data stored in Yahoo Finance may not be 100% accurate and we may try purchase the high-quality data (daily or minute).
- Try other models such as recurrent neural network (RNN) and long short-term memory (LSTM).
- For prediction stock's movement, not only count if outperforms median or not, try multi-classification
- Use different datasets such the Financial Times Stock Exchange (FTSE) 100 or Asian market data.
- Analysis detailly with more criteria such as maximum drawdown and reasons why in the years 2013, 2014, 2016, 2017 and 2019, my method didn't perform well.
- Continue to tune and find the "optimal" parameters to replace table 3 and do the momentum trading again.

Reference

Jegadeesh, N., & Titman, S. (1993). Returns to buying winners and selling losers: Implications for stock market efficiency. The Journal of finance, 48(1), 65-91.

Krauss, C., Do, X. A., & Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operational Research, 259(2), 689-702.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Fischer, T. G., Krauss, C., & Deinert, A. (2019). Statistical arbitrage in cryptocurrency markets. Journal of Risk and Financial Management, 12(1), 31.