Statistical Arbitrage on S&P 500 Components by Machine Learning Models

Springboard Data Science Career Track Capstone 1, April 27th 2020 Cohort
Jiaqi Xu
09/23/2020

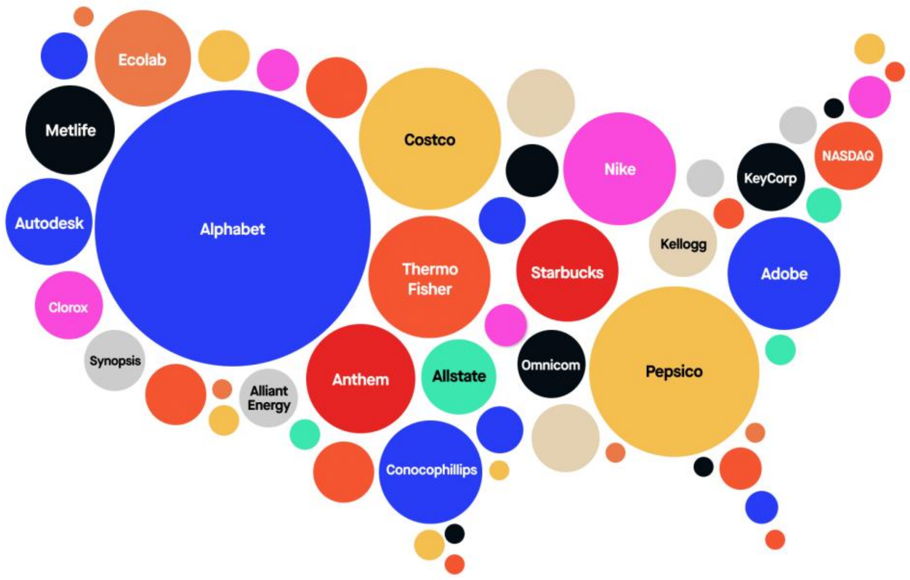
Thanks for Springboard mentor Kevin Glynn

The Next 20 minutes is dictated to

- Project Introduction
- Dataset Extract, Transform and Load (ETL)
- Exploration Data Analysis (EDA)
- Machine Learning Models, Parameters Tuning and Prediction Results
- Trading Results and Analysis
- Future Work

Why S&P 500 Components

- Attractive return - 9.8%/year, including dividends, since inception in 1926 (Wikipedia).
- Mr. Warren Buffet recommended to invest S&P 500 index (Berkshire Hathaway's annual meeting 2020).
- A hot topic to do statistical arbitrage trading on S&P 500 components.



Source: Robinhood Learn

Previous Related Work

- Jegadeesh and Titman (1993) - Early paper for momentum trading method. Select stocks based on their past 6-month returns and holds them for 6 months. Realize a compounded excess return of 12.01%/year from 1965-1989.

- Krauss et al. (2017) - A statistical arbitrage strategy based on deep neural networks (DNN), gradient-boosted trees (GBT), random forests (RAF) on S&P 500 constitutes from 1992 to 2015. Realized return 0.25%/day, Sharpe ratio 1.81/year without transaction cost.

- Fischer and Krauss (2018) - Long short-term memory (LSTM) networks on S&P 500 constitutes. Realized return 0.46%/day, Sharpe ratio 5.8/year without transaction cost.

- Fischer et al. (2019) - Logistic regression (LG) and RAF with 40 cryptocurrency coins on minute-binned data from 18 June 2018 to 17 September 2018. Realized excess return 0.038%/round-trip trade after transaction cost with 0.15%/half-turn.

Goal of This Project

- Find the optimal machine learning methods in one-step stock movement prediction with updated dataset (2005-01-01 to 2019-12-31) .

- Compare the results of my machine learning methods to the previous results, Krauss et al. (2017) and Fischer and Krauss (2018).

- Check the hypothesis that “profits are declining in recent years and there is a severe challenge to the semi-strong form of market efficiency”.

Dataset Extract, Transform and Load (ETL)

- List of historical S&P 500 components - Wharton Research Data Services (WRDS) and Wikipedia.
- Daily data of historical S&P 500 components - Python Yahoo finance module yahoo-finance-1.0.4.
- Missing Components – Delisted from Yahoo finance.

Exploration Data Analysis (EDA)

- **Dataset**: S&P 500 composites from 2005/01/01 to 2020/01/01 (15 years data).

- **Subsets**: Split dataset into 12 subsets - each subset, 750 days/250 days as formation and trading window. All trading windows are non-overlapped. Each formation window, 80% training (5 folder cross validation), 20% testing.

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

Exploration Data Analysis (EDA)

Feature generation:

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

- *Output:* A binary response variable $Y_{t+j,j}^s \in \{0,1\}$ for each stock s. The response $Y_{t+j,j}^s$ equals to one (class 1), if the j-period return $R_{t+j,j}^s$ of stock s is larger than the corresponding cross-sectional median return computed over all stocks and zero otherwise (class 0). Here, set j=1.

Feature Description:

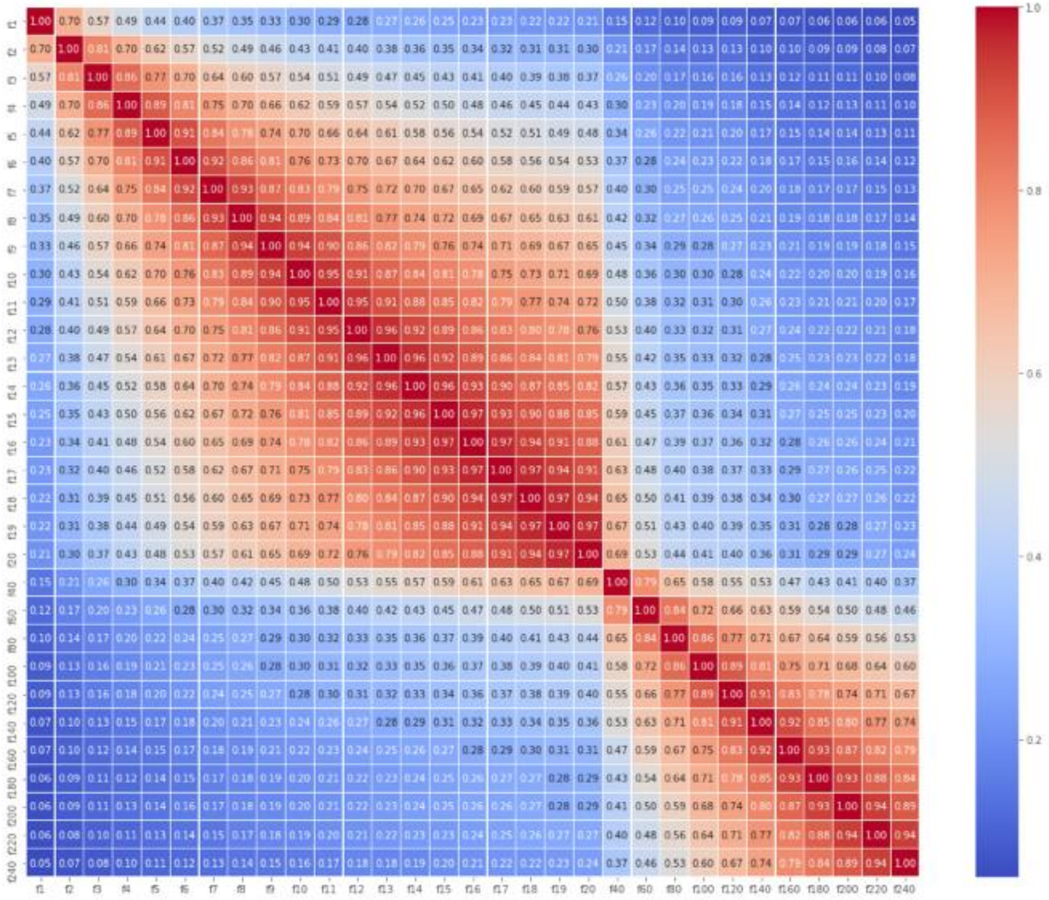


Figure 1. Feature Correlation Heatmap from Formation Subset 10

Machine Learning Methods, Grid-search parameters, Results

<table>
<tr><th rowspan="2">Methods</th><th rowspan="2">Paramters to be tuned</th><th rowspan="2">Be selected values</th><th colspan="12">Optimal values for Dataset</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th></tr>
<tr><td rowspan="3">knn</td><td>Number of neighbors</td><td>{3,5,7,11}</td><td colspan="7">11</td><td>5</td><td colspan="2">3</td><td>5</td><td>11</td></tr>
<tr><td>Weight function used in prediction</td><td>{'uniform','distance'}</td><td>'uniform'</td><td colspan="2">'distance'</td><td colspan="2">'uniform'</td><td colspan="7">'distance'</td></tr>
<tr><td>Metric</td><td>{'euclidean', 'manhattan'}</td><td>'manhattan'</td><td colspan="2">'euclidean'</td><td colspan="2">'manhattan'</td><td colspan="2">'euclidean'</td><td colspan="5">'euclidean'</td></tr>
<tr><td rowspan="4">raf</td><td>The number of trees</td><td>{1500,2000,2500}</td><td colspan="3">2500</td><td>2000</td><td colspan="8">2500</td></tr>
<tr><td>The maximum depth of the tree</td><td>{20,25,30}</td><td colspan="2">20</td><td colspan="2">30</td><td colspan="2">25</td><td colspan="2">30</td><td>25</td><td colspan="3">20</td></tr>
<tr><td>The number of features for the best split</td><td>{log2(n_features), sqrt(n_features)}</td><td>"sqrt"</td><td colspan="2">"log2"</td><td colspan="2">"sqrt"</td><td colspan="7">"log2"</td></tr>
<tr><td>The number of samples to draw in boostrap</td><td>{500, 700, 1000}</td><td>700</td><td colspan="2">1000</td><td>700</td><td colspan="8">1000</td></tr>
<tr><td rowspan="4">lg</td><td>Inverse of regularization strength</td><td>{10^-3, 10^-2,…,10^2,10^3}</td><td>0.01</td><td>0.1</td><td colspan="2">0.001</td><td>1000</td><td>0.01</td><td>0.1</td><td colspan="3">0.001</td><td>10</td><td>0.001</td></tr>
<tr><td>Penalization (learning rate)</td><td>{'l2'}</td><td colspan="12">l2'</td></tr>
<tr><td>Optimization solver</td><td>{'newton-cg','lbfgs','sag'}</td><td>"sag"</td><td>"newton-cg"</td><td>"lbfgs"</td><td>"newton-cg"</td><td>"lgfgs"</td><td>"newton-cg"</td><td>"sag"</td><td colspan="2">"newton-cg"</td><td>"sag"</td><td>"lbfgs"</td><td>"newton-cg"</td></tr>
<tr><td>Maximum iterations</td><td>{10000,20000}</td><td colspan="12">1000</td></tr>
<tr><td rowspan="4">xgboost</td><td>Maximum depth of a tree</td><td>{3,5,7}</td><td colspan="12">7</td></tr>
<tr><td>Minimum sum of instance weight needed in a child</td><td>{5,10}</td><td colspan="8">5</td><td colspan="2">10</td><td colspan="2">5</td></tr>
<tr><td>Number of trees</td><td>{500, 1000}</td><td colspan="12">1000</td></tr>
<tr><td>Penalization (learning rate)</td><td>{0.01, 0.02, 0.05, 0.1}</td><td>0.01</td><td colspan="2">0.02</td><td colspan="3">0.05</td><td>0.02</td><td colspan="2">0.01</td><td colspan="3">0.02</td></tr>
<tr><td>ensemble 1</td><td>Average of knn-xgboost</td><td>Average</td><td colspan="12" rowspan="2"></td></tr>
<tr><td>ensemble 2</td><td>Stacking of knn-xgboost</td><td>Meta-function</td></tr>
<tr><td>mlp</td><td>structures, early stoping, activate function</td><td>31-10-5-1, 256-128-64-32-1</td><td colspan="12">256-128-64-32-1</td></tr>
</table>

Table 2: Grid-search parameters of machine learning methods with the optimal results

Result from Formation (Train/Test) Window

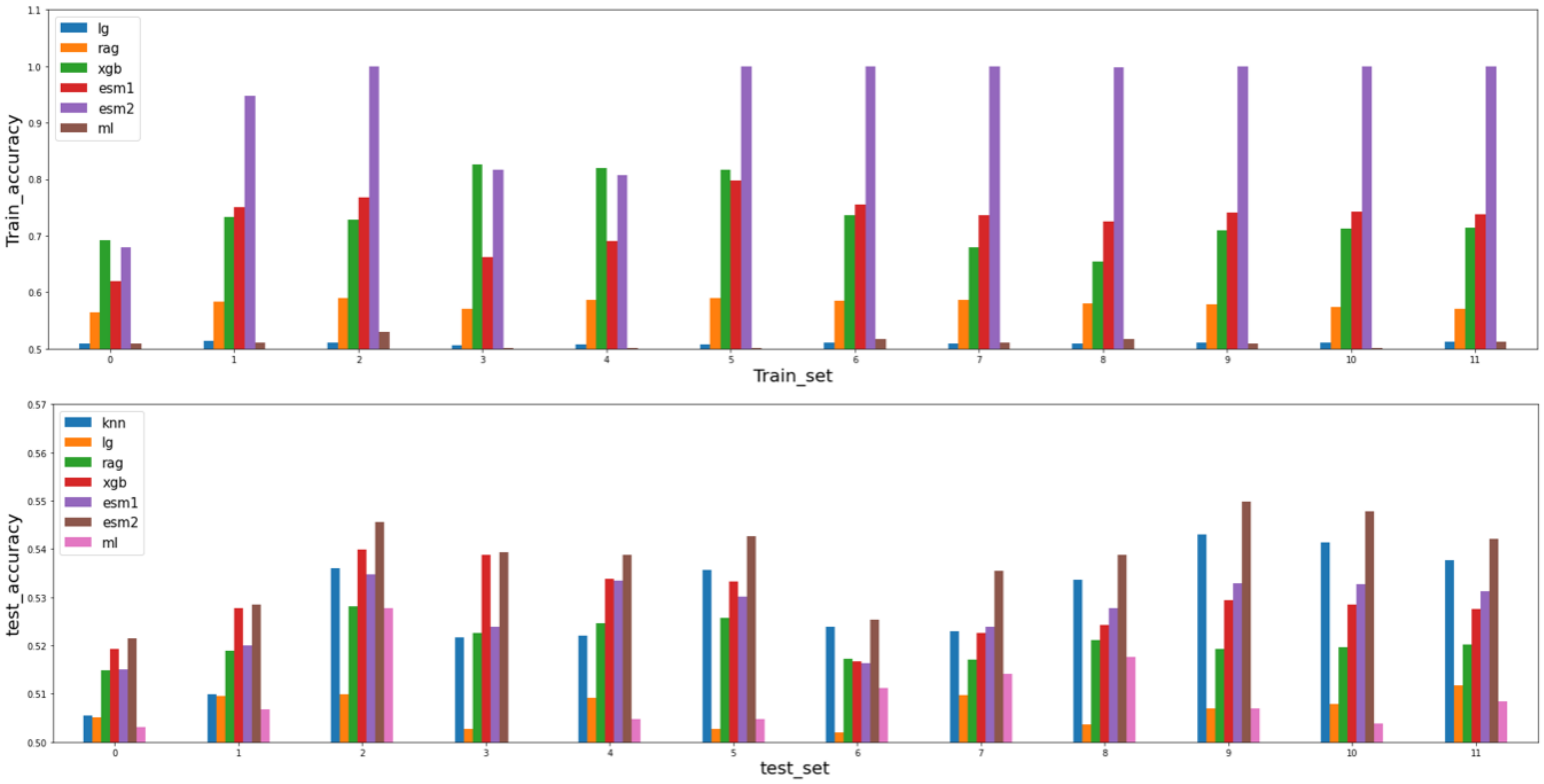


Figure 2. Train and Test Accuracy for machine learning methods with optimal parameters in all subsets

- For train subsets

- 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.

- For test subsets

- Stacking ensemble 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.

Trading window with Momentum

For each of trading dataset

- Sorting 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 table2.

- Long the highest ("expected winner") k probabilities and short the lowest ("expected loser") k probabilities simultaneously.

Trading Results of Momentum (3 stocks/pair)

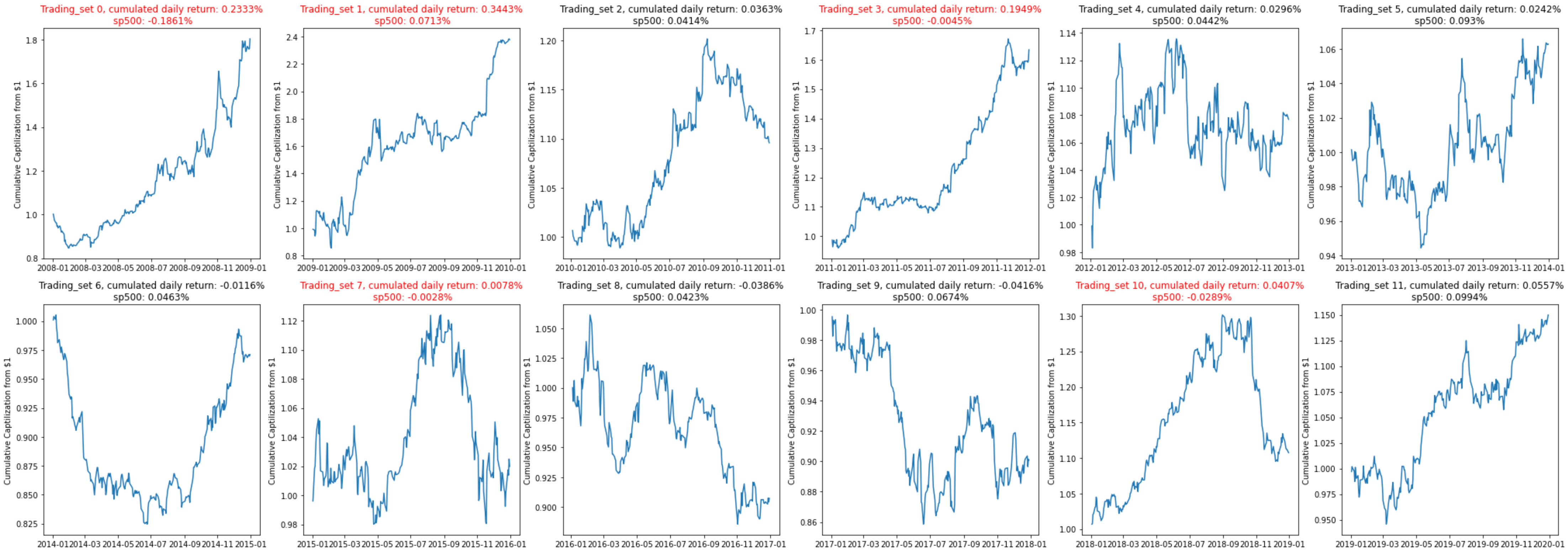


Figure 3: Results of Momentum with 3 stock/pairs

Stocks K=3, 5, 10, 15, 20/pair vs S&P 500

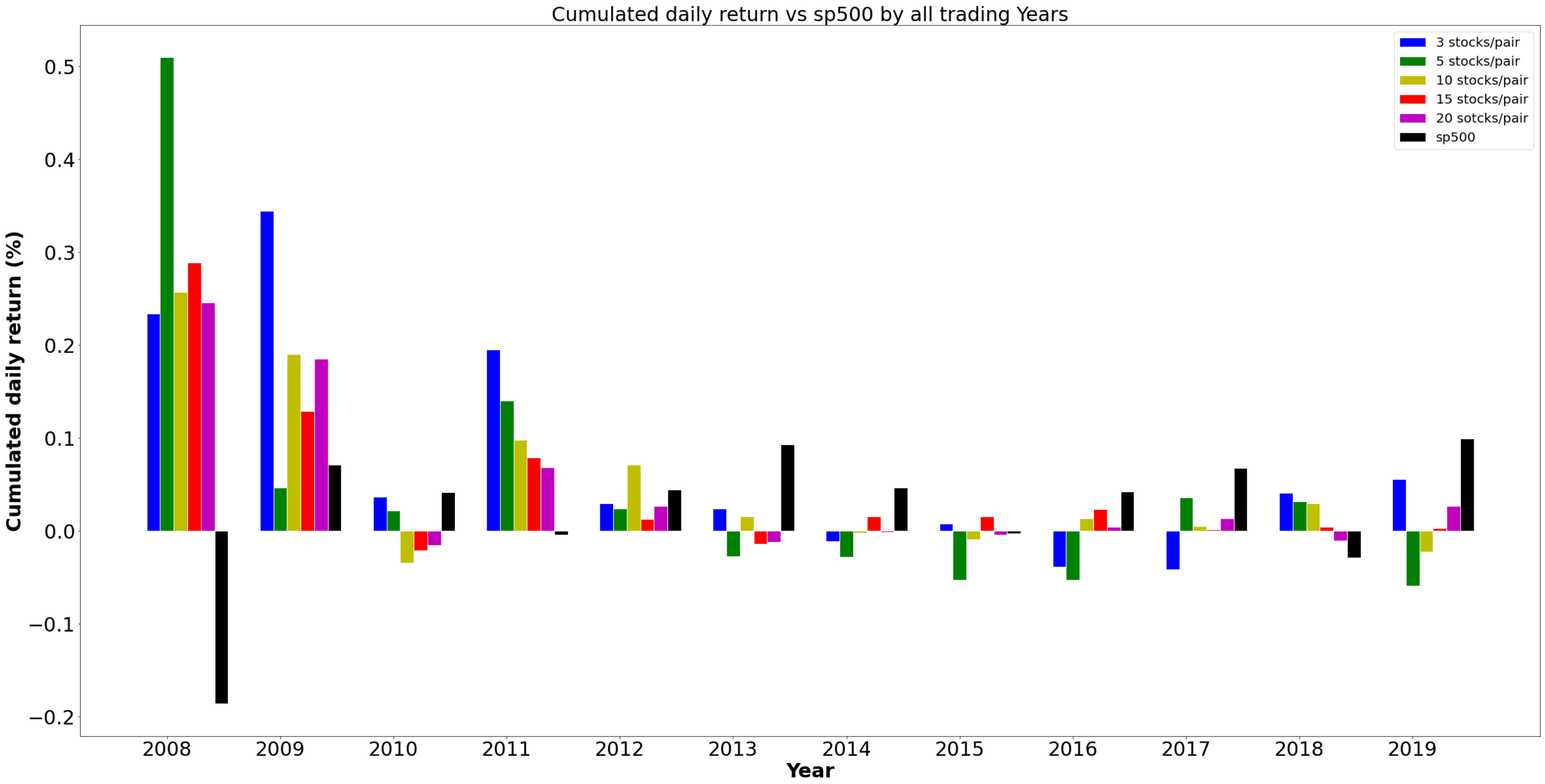


Figure 4: Results of Momentum with 3,5,10,15,20 stock/pair

Conclusion

- 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% after taking the transaction cost while positive daily returns still can 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."

Future Work

- 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).
- Try multi-classification to predict stock's movement.
- 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 doesn'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.

Thank You!

Jiaqi Xu

Email: jiaqiperson@gmail.com

LinkedIn: https://www.linkedin.com/in/jiaqixu1

ResearchGate: https://www.researchgate.net/profile/Jiaqi_Xu10

Project report: https://github.com/jiaqixu/Springboard/blob/master/Capstone/Capstone1/Capstone1_Final_report.pdf