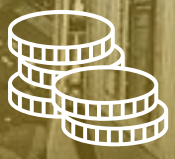
Not Over Thinking

Optimize Asset Selection with Multi-factor Models Backed by Machine Learning Techniques



38.10% Annualised Return; Information Ratio = 1.339

19 factors at play including Fama-French and much more

Machine Learning models to optimize stock selection – Ridge Regression, SVM, Random Forest

Haixiang
2022.04 | Vol 2.
hxyan.2015@gmail.com | github.com/hxyan2020

Strategy & Economic Rationale

Factor models have long been criticised for its instability and decreasing prediction capability, especially in the new century when large volume of transactions have been handed over to automation. However, upon closer investigation, it is not the factor theory that has become inefficient, and if we optimise factor selection algorithm to identify the factors truly at play and remove the ones that don't in a machine-calculated manner, we are able to improve strategy return considerably.

In this strategy, I employed several linear machine learning models such as SVM and RF to fine tune the factor identification process with grid search. Factors are obtained by placing model results in ascending orders select the top ones as factor in play. Results are also attached in the analysis.

Parameter Setup

- Portfolio Rebalancing Frequency: 30-day (flexible)
- Long/Short (function 'buy_signal'): Long and Short
- Factor List (list 'g.__factorList'):
    - Valuation factors
        - Earning to Price EP
        - Book Value to Price BP
        - Price per Share PS
        - Dividend per Price DP
        - Rate of Debt RD
        - Cashflow Projection CFP
    - Capital Structure
        - Log_NC
        - Leverage LEV
        - Current Market Value CMV
        - Fixed Asset Coverage Ratio FACR
    - Profitability
        - NI_p
        - NI_n
        - Gross Profit Margin GPM
        - Return on Equity ROE
        - Return on Asset ROA
        - OPTP
    - Growth
        - Price/Earning-to-Growth PEG
        - g
        - G_p
- Training Models (list 'g.method', k-fold = 4, *grid search setup see following screenshot)
    - SVM (C = 100, gamma = 1)
    - Linear Regression
    - Ridge Regression (Random State = 42, Alpha = 100)
    - Random Forest (Random State = 452, Estimator = 500)
- Benchmark Index (function 'set_benchmark'): 000985.XSHG (Comprehensive index covering both Shanghai and Shenzhen Exchanges)
- Stock Holdings (variable 'g.__feasible_stocks'): eliminate suspended stocks and use the remaining as "feasible stocks" – out of feasible stocks, 5 stocks are eventually singled out and held due to optimal factor performance.
- Back-testing period: 1 January 2014 – 31 July 2018
- Slippage and Transaction Fees (function 'set_slip_fee'): 0.0003 to 0.002 depending on exchange rates

Determine Signal

1. Before execution, tidy up data by winsorize, standardize, and neutralize (across industry sectors) the factor list.

```
# winsorization
for fac in g.__winsorizeList:
    df_train[fac] = winsorize_med(df_train[fac], scale=5, inclusive=True, inf2nan=True, axis=0)
    df[fac] = winsorize_med(df[fac], scale=5, inclusive=True, inf2nan=True, axis=0)

# standardization
for fac in g.__standardizeList:
    df_train[fac] = standardlize(df_train[fac], inf2nan=True, axis=0)
    df[fac] = standardlize(df[fac], inf2nan=True, axis=0)

# neutralization
df_train = neutralize(df_train,g.__industry_set)
df = neutralize(df,g.__industry_set)
```

2. Set cross validation-fold = 4 and define 4 methods deployed (SVR, Linear Regression, Ridge Regression, Random Forest; with and without grid search)

```
kfold = KFold(n_splits=4)
if g.__gridserach == False:
    #machine learning w/o grid search
    if g.method == 'svr': #SVR
        from sklearn.svm import SVR
        model = SVR(C=100, gamma=1)
    elif g.method == 'lr':
        from sklearn.linear_model import LinearRegression
        model = LinearRegression()
    elif g.method == 'ridge': #ridge regression
        from sklearn.linear_model import Ridge
        model = Ridge(random_state=42,alpha=100)
    elif g.method == 'rf': #ransom forest
        from sklearn.ensemble import RandomForestRegressor
        model = RandomForestRegressor(random_state=42,n_estimators=500,n_jobs=-1)
    else:
        g.__scoreWrite = False
```

```
else:
    # machine learning with grid search
    para_grid = {}
    if g.method == 'svr':
        from sklearn.svm import SVR
        para_grid = {'C':[10,100],'gamma':[0.1,1,10]}
        grid_search_model = SVR()
    elif g.method == 'lr':
        from sklearn.linear_model import LinearRegression
        grid_search_model = LinearRegression()
    elif g.method == 'ridge':
        from sklearn.linear_model import Ridge
        para_grid = {'alpha':[1,10,100]}
        grid_search_model = Ridge()
    elif g.method == 'rf':
        from sklearn.ensemble import RandomForestRegressor
        para_grid = {'n_estimators':[100,500,1000]}
        grid_search_model = RandomForestRegressor()
    else:
        g.__scoreWrite = False
```

3. Factors obtained by ranking the difference between actual values (training set) and predicted values

```
y_pred = model.predict(X)

# factors obtained by the difference bt. actual values (training set) and predicted values
factor = y - pd.DataFrame(y_pred, index = y.index, columns = ['log_mcap'])

#sort
factor = factor.sort_index(by = 'log_mcap')
```

4. Decide buy and sell list

```
#buy in
for stock in stockset[:g.stocknum]:
    if stock in sell_list:
        pass
    else:
        if current_data[stock].last_price == current_data[stock].low_limit:
            pass
        else:
            stock_buy = stock
            order_target_value(stock_buy, cash)
            num = num - 1
            if num == 0:
                break
```

Back-testing Performance

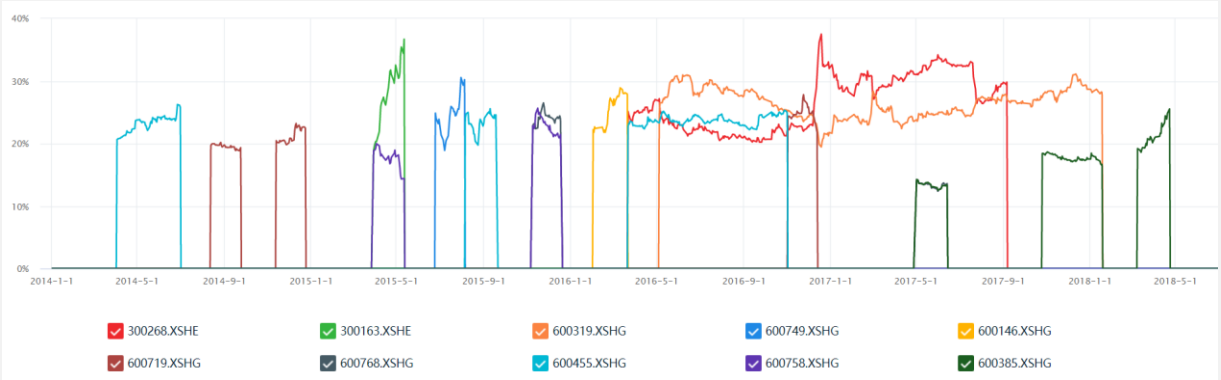
- Cumulative return = 781.04%, with annualised return = 24.00%

- Other Indicators

Alpha	Beta	Sharpe Ratio	Sortino Ratio	Win Ratio	Maximum Drawdown	Information Ratio	Volatility	Calmar Ratio	R^2
0.30	0.97	1.02	1.26	0.548	59%	1.34	0.36	0.65	0.36

- Factor Analysis

- Top 10 Stock Holdings



Potential Improvements

- Portfolio Rebalancing rate set at 30-day for the above testing, which could compromise the agility of algorithm.
- Only top 5 stocks with optimal factor performance are held at execution. Further test could be run to decide the optimal number of holdings per trading day.
- Rigidity of machine learning models could be further examined to arrive at optimal parameter setups.
- Lack of factor building data for some of the stocks in the feasible stock list. Given the regulatory environment in the Chinese stock market, accuracy of some corporate reporting should be re-examined.

For source codes, please visit my GitHub: github.com/hxyan2020. I will upload them once China wins the AFC Asia Cup, or earlier…