Introduction

Most factor models used in trading strategies are cross-sectional, meaning they rely on the relative ranking of stocks at a specific time. It is assumed that this ranking will remain stable over time and that the factors used for ranking will predict future returns. However, this assumption may not hold true under all market conditions, and the predictability of these factors can change over time, as can market anomalies. This can result in a model that overfits historical data and fails to generalize to new data.

In this report, we will explore a rolling window approach to address this issue, we will selected significant factors in the rolling window and combine them with time-series components to create a more robust model. Then we will build a trading strategy accordingly. The goal is to create a trading strategy that is more robust to changes in market conditions and can adapt to new data.

To achieve this, we selected Fama-French 5 factors plus momentum as the base factors. Then we constructed anomalies based on 101 Formulaic Alphas [Zura Kakushadze, 2015]. Finally, we used a rolling window approach to select the most significant factors, anomalies (alphas) and combined them with time-series components to create a more robust model.

Data Overview

We selected the following data for our analysis:

- **Daily prices and volumes**: Daily prices and volumes for all stocks in the S&P 500 index.
- **Fama-French 5 factors plus momentum**: mktrf, smb, hml, rmw, cma, umd.
- **101 Formulaic Alphas**: We constructed these using daily prices and volumes.
- **Daily returns for 101 Formulaic Alphas**: We generated daily returns for the 101 Formulaic Alphas using high-low arbitrage portfolios.

The data spans from January 1, 2004, to December 31, 2024, covering 20 years. We believe this is a sufficient amount of data to capture market dynamics across various conditions.

Daily Prices and Volumes

The daily prices and volumes for all stocks in the S&P 500 index include:

- **Close, Open, High, Low**: Adjusted close, open, high, and low prices.
- **Vol**: Daily volume.
- **Pct_change**: Daily percentage change. For example, -0.83 indicates a -0.83% daily return.
- **VWAP**: Volume-weighted average price.

We will use this price data to construct the 101 Formulaic Alphas and conduct backtesting. Since we have a sufficient amount of data, we will not fill in any missing data. This approach utilizes a rolling window, which helps us avoid overfitting. For alpha construction, if a stock has missing data during the construction period, we will skip that stock.

Fama-French 5 Factors plus Momentum

The Fama-French 5 factors plus momentum daily factors returns were obtained from WRDS(Wharton Research Data Services). The factors include:

- **mktrf**: Market return minus risk-free rate.
- **smb**: Small minus big (size factor).
- **hml**: High minus low (value factor).
- **rmw**: Robust minus weak (profitability factor).
- **cma**: Conservative minus aggressive (investment factor).
- **umd**: Up minus down (momentum factor).

Alphas Construction

The 101 Formulaic Alphas paper [Zura Kakushadze, 2015] provides a comprehensive list of 101 real-life quantitative trading alphas. The original paper have a average holding period approximately ranges from 0.6 to 6.4 days. For our analysis, since we aim to construct a daily frequency trading strategy, we will hold the alphas for 1 day. However, the look back period for the alphas construction still varies based on the alphas construction method, details can be found in the paper.

We calculated alphas for all stocks in the S&P 500. A sample of the alpha values is shown below, including the alpha value for **AAPL** from January 3, 2005, to January 6, 2005:

trade_date	alpha001	alpha002	alpha003
2005-01-03	0.359342330355	-0.00060693612848	-0.22883143600105
2005-01-04	0.204808971678	-0.06933950066987	0.39975096612885
2005-01-05	0.065386808591	0.24332731404990	0.23698080798364
2005-01-06	0.817049990496	0.13108774439945	0.24530729110312

Then we sorted the alphas by their values on daily basis. Dividing the stocks into 5 groups, We used the arbitrage portfolio (long the top 20% and short the bottom 20%) to calculate the daily returns of each alpha. Sample of the daily returns for the alphas are calculated as follows:

trade_date	alpha001	alpha002	alpha003
2005-01-03	-0.001604	0.004097	0.003343
2005-01-04	-0.002053	-0.001746	0.003168
2005-01-05	0.000321	-0.001533	-0.001220
2005-01-06	-0.002266	0.003295	-0.001950

Significant Alphas Selection

In our backtesting period, 2004-2024, we first selected the alphas daily return is siginificantly different from 0. We used a t-test to determine the significance of the alphas. At the 5% significance level, 29 alphas were found to be significant. The t-test results are shown below:

Alpha	t-stat	Alpha	t-stat	Alpha	t-stat
alpha003	-3.989	alpha060	-4.593	alpha006	-4.558
alpha051	-9.596	alpha025	-11.888	alpha020	-5.886
alpha049	-10.508	alpha028	-4.890	alpha101	9.541
alpha047	-3.564	alpha018	-7.907	alpha040	-2.907
alpha046	-8.055	alpha017	-11.038	alpha005	-2.740
alpha042	-5.334	alpha014	-13.343	alpha016	-2.534
alpha038	-10.072	alpha013	-4.074	alpha053	2.530
alpha035	4.996	alpha012	-4.152	alpha008	-2.374
alpha034	-12.283	alpha010	-3.947	alpha045	1.983
alpha033	-14.349	alpha009	-5.562		

The absolute correlation between the significant alphas daily returns is relatively low, the average absolute correlation is 0.15. The maximum absolute correlation is 0.71, which is between alpha009 and alpha010 (Due to their similar construction method).

Appendix

Appendix A: Formulaic Expressions for Alphas

From the 101 Formulaic Alphas paper [Zura Kakushadze, 2015], we have the following formulaic expressions for the alphas:

Alpha#1: (rank(Ts_ArgMax(SignedPower(((returns < 0) ? stddev(returns, 20) : close), 2.), 5)) -0.5)

Alpha#2: (-1 * correlation(rank(delta(log(volume), 2)), rank(((close - open) / open)), 6))

Alpha#3: (-1 * correlation(rank(open), rank(volume), 10))

Alpha#4: (-1 * Ts_Rank(rank(low), 9))

Alpha#5: (rank((open - (sum(vwap, 10) / 10))) * (-1 * abs(rank((close - vwap)))))

Alpha#6: (-1 * correlation(open, volume, 10))

Alpha#7: ((adv20 < volume) ? ((-1 * ts_rank(abs(delta(close, 7)), 60)) * sign(delta(close, 7))) : (-1 * 1))

Alpha#8: (-1 * rank(((sum(open, 5) * sum(returns, 5)) - delay((sum(open, 5) * sum(returns, 5)), 10))))

Alpha#9: ((0 < ts_min(delta(close, 1), 5)) ? delta(close, 1) : ((ts_max(delta(close, 1), 5) < 0) ? delta(close, 1) : (-1 * delta(close, 1))))

Alpha#10: rank(((0 < ts_min(delta(close, 1), 4)) ? delta(close, 1) : ((ts_max(delta(close, 1), 4) < 0) ? delta(close, 1) : (-1 * delta(close, 1)))))

Alpha#11: ((rank(ts_max((vwap - close), 3)) + rank(ts_min((vwap - close), 3))) * rank(delta(volume, 3)))

Alpha#12: (sign(delta(volume, 1)) * (-1 * delta(close, 1)))

Alpha#13: (-1 * rank(covariance(rank(close), rank(volume), 5)))

Alpha#14: ((-1 * rank(delta(returns, 3))) * correlation(open, volume, 10))

Alpha#15: (-1 * sum(rank(correlation(rank(high), rank(volume), 3)), 3))

Alpha#16: (-1 * rank(covariance(rank(high), rank(volume), 5)))

Alpha#17: (((-1 * rank(ts_rank(close, 10))) * rank(delta(delta(close, 1), 1))) * rank(ts_rank((volume / adv20), 5)))

Alpha#18: (-1 * rank(((stddev(abs((close - open)), 5) + (close - open)) + correlation(close, open, 10))))

Alpha#19: ((-1 * sign(((close - delay(close, 7)) + delta(close, 7)))) * (1 + rank((1 + sum(returns, 250)))))

Alpha#20: (((-1 * rank((open - delay(high, 1)))) * rank((open - delay(close, 1)))) * rank((open delay(low, 1))))

Alpha#21: ((((sum(close, 8) / 8) + stddev(close, 8)) < (sum(close, 2) / 2)) ? (-1 * 1) : (((sum(close, 2) / 2) < ((sum(close, 8) / 8) - stddev(close, 8))) ? 1 : (((1 < (volume / adv20)) || ((volume / adv20) == 1)) ? 1 : (-1 * 1))))

Alpha#22: (-1 * (delta(correlation(high, volume, 5), 5) * rank(stddev(close, 20))))

Alpha#23: (((sum(high, 20) / 20) < high) ? (-1 * delta(high, 2)) : 0)

Alpha#24: ((((delta((sum(close, 100) / 100), 100) / delay(close, 100)) < 0.05) || ((delta((sum(close, 100) / 100), 100) / delay(close, 100)) == 0.05)) ? (-1 * (close - ts_min(close, 100))) : (-1 * delta(close, 3)))

Alpha#25: rank(((((-1 * returns) * adv20) * vwap) * (high - close)))

Alpha#26: (-1 * ts_max(correlation(ts_rank(volume, 5), ts_rank(high, 5), 5), 3))

Alpha#27: ((0.5 < rank((sum(correlation(rank(volume), rank(vwap), 6), 2) / 2.0))) ? (-1 * 1) : 1)

Alpha#28: scale(((correlation(adv20, low, 5) + ((high + low) / 2)) - close))

Alpha#29: (min(product(rank(rank(scale(log(sum(ts_min(rank(rank((-1 * rank(delta((close - 1), 5))))), 2), 1))))), 1), 5) + ts_rank(delay((-1 * returns), 6), 5))

Alpha#30: (((1.0 - rank(((sign((close - delay(close, 1))) + sign((delay(close, 1) - delay(close, 2)))) + sign((delay(close, 2) - delay(close, 3)))))) * sum(volume, 5)) / sum(volume, 20))

Alpha#31: ((rank(rank(rank(decay_linear((-1 * rank(rank(delta(close, 10)))), 10)))) + rank((-1 * delta(close, 3)))) + sign(scale(correlation(adv20, low, 12))))

Alpha#32: (scale(((sum(close, 7) / 7) - close)) + (20 * scale(correlation(vwap, delay(close, 5),

230))))

Alpha#33: rank((-1 * ((1 - (open / close))^1)))

Alpha#34: rank(((1 - rank((stddev(returns, 2) / stddev(returns, 5)))) + (1 - rank(delta(close, 1)))))

Alpha#35: ((Ts_Rank(volume, 32) * (1 - Ts_Rank(((close + high) - low), 16))) * (1 Ts_Rank(returns, 32)))

Alpha#36: (((((2.21 * rank(correlation((close - open), delay(volume, 1), 15))) + (0.7 * rank((open - close)))) + (0.73 * rank(Ts_Rank(delay((-1 * returns), 6), 5)))) + rank(abs(correlation(vwap, adv20, 6)))) + (0.6 * rank((((sum(close, 200) / 200) - open) * (close - open)))))

Alpha#37: (rank(correlation(delay((open - close), 1), close, 200)) + rank((open - close)))

Alpha#38: ((-1 * rank(Ts_Rank(close, 10))) * rank((close / open)))

Alpha#39: ((-1 * rank((delta(close, 7) * (1 - rank(decay_linear((volume / adv20), 9)))))) * (1 + rank(sum(returns, 250))))

Alpha#40: ((-1 * rank(stddev(high, 10))) * correlation(high, volume, 10))

Alpha#41: (((high * low)^0.5) - vwap)

Alpha#42: (rank((vwap - close)) / rank((vwap + close)))

Alpha#43: (ts_rank((volume / adv20), 20) * ts_rank((-1 * delta(close, 7)), 8))

Alpha#44: (-1 * correlation(high, rank(volume), 5))

Alpha#45: (-1 * ((rank((sum(delay(close, 5), 20) / 20)) * correlation(close, volume, 2)) * rank(correlation(sum(close, 5), sum(close, 20), 2))))

Alpha#46: ((0.25 < (((delay(close, 20) - delay(close, 10)) / 10) - ((delay(close, 10) - close) / 10))) ? (-1 * 1) : (((((delay(close, 20) - delay(close, 10)) / 10) - ((delay(close, 10) - close) / 10)) < 0) ? 1 : ((-1 * 1) * (close - delay(close, 1)))))

Alpha#47: ((((rank((1 / close)) * volume) / adv20) * ((high * rank((high - close))) / (sum(high, 5) / 5))) - rank((vwap - delay(vwap, 5))))

Alpha#48: (indneutralize(((correlation(delta(close, 1), delta(delay(close, 1), 1), 250) * delta(close, 1)) / close), IndClass.subindustry) / sum(((delta(close, 1) / delay(close, 1))^2), 250))

Alpha#49: (((((delay(close, 20) - delay(close, 10)) / 10) - ((delay(close, 10) - close) / 10)) < (-1 * 0.1)) ? 1 : ((-1 * 1) * (close - delay(close, 1))))

Alpha#50: (-1 * ts_max(rank(correlation(rank(volume), rank(vwap), 5)), 5))

Alpha#51: (((((delay(close, 20) - delay(close, 10)) / 10) - ((delay(close, 10) - close) / 10)) < (-1 * 0.05)) ? 1 : ((-1 * 1) * (close - delay(close, 1))))

Alpha#52: ((((-1 * ts_min(low, 5)) + delay(ts_min(low, 5), 5)) * rank(((sum(returns, 240) sum(returns, 20)) / 220))) * ts_rank(volume, 5))

Alpha#53: (-1 * delta((((close - low) - (high - close)) / (close - low)), 9))

Alpha#54: ((-1 * ((low - close) * (open^5))) / ((low - high) * (close^5)))

Alpha#55: (-1 * correlation(rank(((close - ts_min(low, 12)) / (ts_max(high, 12) - ts_min(low, 12)))), rank(volume), 6))

Alpha#56: (0 - (1 * (rank((sum(returns, 10) / sum(sum(returns, 2), 3))) * rank((returns * cap)))))

Alpha#57: (0 - (1 * ((close - vwap) / decay_linear(rank(ts_argmax(close, 30)), 2))))

Alpha#58: (-1 * Ts_Rank(decay_linear(correlation(IndNeutralize(vwap, IndClass.sector), volume, 3.92795), 7.89291), 5.50322))

Alpha#59: (-1 * Ts_Rank(decay_linear(correlation(IndNeutralize(((vwap * 0.728317) + (vwap * (1 - 0.728317))), IndClass.industry), volume, 4.25197), 16.2289), 8.19648))

Alpha#60: (0 - (1 * ((2 * scale(rank(((((close - low) - (high - close)) / (high - low)) * volume)))) scale(rank(ts_argmax(close, 10))))))

Alpha#61: (rank((vwap - ts_min(vwap, 16.1219))) < rank(correlation(vwap, adv180, 17.9282)))

Alpha#62: ((rank(correlation(vwap, sum(adv20, 22.4101), 9.91009)) < rank(((rank(open) + rank(open)) < (rank(((high + low) / 2)) + rank(high))))) * -1)

Alpha#63: ((rank(decay_linear(delta(IndNeutralize(close, IndClass.industry), 2.25164), 8.22237)) - rank(decay_linear(correlation(((vwap * 0.318108) + (open * (1 - 0.318108))), sum(adv180, 37.2467), 13.557), 12.2883))) * -1)

Alpha#64: ((rank(correlation(sum(((open * 0.178404) + (low * (1 - 0.178404))), 12.7054), sum(adv120, 12.7054), 16.6208)) < rank(delta(((((high + low) / 2) * 0.178404) + (vwap * (1 - 0.178404))), 3.69741))) * -1)

Alpha#65: ((rank(correlation(((open * 0.00817205) + (vwap * (1 - 0.00817205))), sum(adv60, 8.6911), 6.40374)) < rank((open - ts_min(open, 13.635)))) * -1)

Alpha#66: ((rank(decay_linear(delta(vwap, 3.51013), 7.23052)) + Ts_Rank(decay_linear(((((low * 0.96633) + (low * (1 - 0.96633))) - vwap) / (open - ((high + low) / 2))), 11.4157), 6.72611)) * -1)

Alpha#67: ((rank((high - ts_min(high, 2.14593)))^rank(correlation(IndNeutralize(vwap, IndClass.sector), IndNeutralize(adv20, IndClass.subindustry), 6.02936))) * -1)

Alpha#68: ((Ts_Rank(correlation(rank(high), rank(adv15), 8.91644), 13.9333) < rank(delta(((close * 0.518371) + (low * (1 - 0.518371))), 1.06157))) * -1)

Alpha#69: ((rank(ts_max(delta(IndNeutralize(vwap, IndClass.industry), 2.72412), 4.79344))^Ts_Rank(correlation(((close * 0.490655) + (vwap * (1 - 0.490655))), adv20, 4.92416), 9.0615)) * -1)

Alpha#70: ((rank(delta(vwap, 1.29456))^Ts_Rank(correlation(IndNeutralize(close, IndClass.industry), adv50, 17.8256), 17.9171)) * -1)

Alpha#71: max(Ts_Rank(decay_linear(correlation(Ts_Rank(close, 3.43976), Ts_Rank(adv180, 12.0647), 18.0175), 4.20501), 15.6948), Ts_Rank(decay_linear((rank(((low + open) - (vwap + vwap)))^2), 16.4662), 4.4388))

Alpha#72: (rank(decay_linear(correlation(((high + low) / 2), adv40, 8.93345), 10.1519)) / rank(decay_linear(correlation(Ts_Rank(vwap, 3.72469), Ts_Rank(volume, 18.5188), 6.86671), 2.95011)))

Alpha#73: (max(rank(decay_linear(delta(vwap, 4.72775), 2.91864)), Ts_Rank(decay_linear(((delta(((open * 0.147155) + (low * (1 - 0.147155))), 2.03608) / ((open * 0.147155) + (low * (1 - 0.147155)))) * -1), 3.33829), 16.7411)) * -1)

Alpha#74: ((rank(correlation(close, sum(adv30, 37.4843), 15.1365)) < rank(correlation(rank(((high * 0.0261661) + (vwap * (1 - 0.0261661)))), rank(volume), 11.4791))) * -1)

Alpha#75: (rank(correlation(vwap, volume, 4.24304)) < rank(correlation(rank(low), rank(adv50), 12.4413)))

Alpha#76: (max(rank(decay_linear(delta(vwap, 1.24383), 11.8259)), Ts_Rank(decay_linear(Ts_Rank(correlation(IndNeutralize(low, IndClass.sector), adv81, 8.14941), 19.569), 17.1543), 19.383)) * -1)

Alpha#77: min(rank(decay_linear(((((high + low) / 2) + high) - (vwap + high)), 20.0451)), rank(decay_linear(correlation(((high + low) / 2), adv40, 3.1614), 5.64125)))

Alpha#78: (rank(correlation(sum(((low * 0.352233) + (vwap * (1 - 0.352233))), 19.7428), sum(adv40, 19.7428), 6.83313))ˆrank(correlation(rank(vwap), rank(volume), 5.77492)))

Alpha#79: (rank(delta(IndNeutralize(((close * 0.60733) + (open * (1 - 0.60733))), IndClass.sector), 1.23438)) < rank(correlation(Ts_Rank(vwap, 3.60973), Ts_Rank(adv150, 9.18637), 14.6644)))

Alpha#80: ((rank(Sign(delta(IndNeutralize(((open * 0.868128) + (high * (1 - 0.868128))), IndClass.industry), 4.04545)))ˆTs_Rank(correlation(high, adv10, 5.11456), 5.53756)) * -1)

Alpha#81: ((rank(Log(product(rank((rank(correlation(vwap, sum(adv10, 49.6054), 8.47743))ˆ4)), 14.9655))) < rank(correlation(rank(vwap), rank(volume), 5.07914))) * -1)

Alpha#82: (min(rank(decay_linear(delta(open, 1.46063), 14.8717)), Ts_Rank(decay_linear(correlation(IndNeutralize(volume, IndClass.sector), ((open * 0.634196) + (open * (1 - 0.634196))), 17.4842), 6.92131), 13.4283)) * -1)

Alpha#83: ((rank(delay(((high - low) / (sum(close, 5) / 5)), 2)) * rank(rank(volume))) / (((high low) / (sum(close, 5) / 5)) / (vwap - close)))

Alpha#84: SignedPower(Ts_Rank((vwap - ts_max(vwap, 15.3217)), 20.7127), delta(close, 4.96796))

Alpha#85: (rank(correlation(((high * 0.876703) + (close * (1 - 0.876703))), adv30, 9.61331))ˆrank(correlation(Ts_Rank(((high + low) / 2), 3.70596), Ts_Rank(volume, 10.1595), 7.11408)))

Alpha#86: ((Ts_Rank(correlation(close, sum(adv20, 14.7444), 6.00049), 20.4195) < rank(((open + close) - (vwap + open)))) * -1)

Alpha#87: (max(rank(decay_linear(delta(((close * 0.369701) + (vwap * (1 - 0.369701))), 1.91233), 2.65461)), Ts_Rank(decay_linear(abs(correlation(IndNeutralize(adv81, IndClass.industry), close, 13.4132)), 4.89768), 14.4535)) * -1)

Alpha#88: min(rank(decay_linear(((rank(open) + rank(low)) - (rank(high) + rank(close))), 8.06882)), Ts_Rank(decay_linear(correlation(Ts_Rank(close, 8.44728), Ts_Rank(adv60, 20.6966), 8.01266), 6.65053), 2.61957))

Alpha#89: (Ts_Rank(decay_linear(correlation(((low * 0.967285) + (low * (1 - 0.967285))), adv10, 6.94279), 5.51607), 3.79744) - Ts_Rank(decay_linear(delta(IndNeutralize(vwap,

IndClass.industry), 3.48158), 10.1466), 15.3012))

Alpha#90: ((rank((close - ts_max(close, 4.66719)))^Ts_Rank(correlation(IndNeutralize(adv40, IndClass.subindustry), low, 5.38375), 3.21856)) * -1)

Alpha#91: ((Ts_Rank(decay_linear(decay_linear(correlation(IndNeutralize(close, IndClass.industry), volume, 9.74928), 16.398), 3.83219), 4.8667) rank(decay_linear(correlation(vwap, adv30, 4.01303), 2.6809))) * -1)

Alpha#92: min(Ts_Rank(decay_linear(((((high + low) / 2) + close) < (low + open)), 14.7221), 18.8683), Ts_Rank(decay_linear(correlation(rank(low), rank(adv30), 7.58555), 6.94024), 6.80584))

Alpha#93: (Ts_Rank(decay_linear(correlation(IndNeutralize(vwap, IndClass.industry), adv81, 17.4193), 19.848), 7.54455) / rank(decay_linear(delta(((close * 0.524434) + (vwap * (1 - 0.524434))), 2.77377), 16.2664)))

Alpha#94: ((rank((vwap - ts_min(vwap, 11.5783)))^Ts_Rank(correlation(Ts_Rank(vwap, 19.6462), Ts_Rank(adv60, 4.02992), 18.0926), 2.70756)) * -1)

Alpha#95: (rank((open - ts_min(open, 12.4105))) < Ts_Rank((rank(correlation(sum(((high + low) / 2), 19.1351), sum(adv40, 19.1351), 12.8742))^5), 11.7584))

Alpha#96: (max(Ts_Rank(decay_linear(correlation(rank(vwap), rank(volume), 3.83878), 4.16783), 8.38151), Ts_Rank(decay_linear(Ts_ArgMax(correlation(Ts_Rank(close, 7.45404), Ts_Rank(adv60, 4.13242), 3.65459), 12.6556), 14.0365), 13.4143)) * -1)

Alpha#97: ((rank(decay_linear(delta(IndNeutralize(((low * 0.721001) + (vwap * (1 - 0.721001))), IndClass.industry), 3.3705), 20.4523)) - Ts_Rank(decay_linear(Ts_Rank(correlation(Ts_Rank(low, 7.87871), Ts_Rank(adv60, 17.255), 4.97547), 18.5925), 15.7152), 6.71659)) * -1)

Alpha#98: (rank(decay_linear(correlation(vwap, sum(adv5, 26.4719), 4.58418), 7.18088)) rank(decay_linear(Ts_Rank(Ts_ArgMin(correlation(rank(open), rank(adv15), 20.8187), 8.62571), 6.95668), 8.07206)))

Alpha#99: ((rank(correlation(sum(((high + low) / 2), 19.8975), sum(adv60, 19.8975), 8.8136)) < rank(correlation(low, volume, 6.28259))) * -1)

Alpha#100: (0 - (1 * (((1.5 * scale(indneutralize(indneutralize(rank(((((close - low) - (high close)) / (high - low)) * volume)), IndClass.subindustry), IndClass.subindustry))) scale(indneutralize((correlation(close, rank(adv20), 5) - rank(ts_argmin(close, 30))), IndClass.subindustry))) * (volume / adv20))))

Alpha#101: ((close - open) / ((high - low) + .001))