Applying Machine Learning to Futures Open Interest Data and Other Indicators

By Paul Fomenko, advised by Dan Trepanier

SCU MSBA Capstone Project

2019

Table of Contents

Introduction:

Outperforming market indexes such as the S&P500, the Russell 2000, etc., is the top goal for almost every money manager, asset allocator, and regular investor. To do this, many financial gurus try to create a replicable and repeatable strategy (based on fundamental analysis of companies, technical analysis of stock movement, high frequency trading, algorithmic / programmatic trading, and more) to outperform the market consistently, often with limited success. While chasing this superior return, you may see, in the news, in discussions online, investors frequently opining and guessing on what the "Smart Money" is doing, Smart Money meaning the top performing traders and experts on Wall Street. This Smart Money is often implied to be better at generating alpha (excess return a.k.a. out performance of the market) than the average trader, and therefore traders desire to emulate the patterns of the Smart Money.

In the paper below, I will discuss the sources of data on Smart Money movements, previous attempts at identifying Smart Money movements and other market timing strategies, our recreation of that previous methodology, and finally an application of machine learning techniques to those previous attempts at outperformance to create a more robust indicator from those data sources.

Data Source 1: Commitment of Traders Report:

The financial industry values quantitative analysis. A potentially valuable source of data to determine Smart Money movements is the Commitment of Traders report.[1] This is a report prepared by the Commodity Futures Trading Commission (CFTC). Released every Friday, the report provides a breakdown of the open interest for futures (and options on futures) on the previous Tuesday for all contracts, and is available in text, csv, and excel formats. The data breaks down open interest between "reportables" (funds and traders that are large enough to be reported on by the clearing members and exchanges) and non-reportables, which are essentially retail traders.

This breakdown of the open interest between commercial, professional, and retail traders is potentially helpful in understanding where the "Smart Money" is placing their bets. This assumes the commercial traders are better informed than of retail traders. In addition, the Commitment of Traders data provides insights into a large variety of contract underlyings such as equity indices, short term bonds, and long term bonds. The open interest data can add on another layer of market mechanics (equity vs bond portfolio weighting of Smart Money) to assist in equity market timing. I will recreate a previous attempt

[1] The Commitment of Traders report has existed since 1987, but has only been weekly since March 21, 1992. Before then, it was bi-monthly. In addition, options data was not available until October 6, 1995. For the purposes of this paper and the paper cited, only the weekly data will be used, and also will include options data once available.

to capitalize on this report further into the paper to show exactly how advantageous this Commitment of Traders information can be (See Smart Money Indicator Section).

Figure 1: Sample of Commitment of Traders Data, text format (also available in .csv and .xlsx)

```
--------------------------------------------------------------------------------------------------------------------------------------------------------------
Traders in Financial Futures - Options and Futures Combined Positions as of September 3, 2019
--------------------------------------------------------------------------------------------------------------------------------------------------------------
             Dealer            :          Asset Manager/       :           Leveraged          :             Other            :     Nonreportable    :
          Intermediary         :          Institutional        :             Funds            :          Reportables         :       Positions      :
    Long  :   Short  : Spreading:    Long  :   Short  : Spreading:    Long  :   Short  : Spreading:    Long  :   Short  : Spreading:    Long  :   Short   :
--------------------------------------------------------------------------------------------------------------------------------------------------------------
E-MINI S&P 500 STOCK INDEX - CHICAGO MERCANTILE EXCHANGE   ($50 X S&P 500 INDEX)
CFTC Code #13874A                                                  Open Interest is 3,651,107
Positions
   243,280    421,266    202,730  1,392,661    702,890    329,370    159,915    444,111    738,545     92,270    343,410    133,209    359,128    335,576

Changes from:       August 27, 2019                                Total Change is:   -58,330
   -13,148    -32,062      6,278    -13,448    -15,024    -13,582     12,152     46,241    -91,483     -3,242     17,395     18,958     39,186      4,949

Percent of Open Interest Represented by Each Category of Trader
       6.7       11.5        5.6       38.1       19.3        9.0        4.4       12.2       20.2        2.5        9.4        3.6        9.8        9.2

Number of Traders in Each Category                                 Total Traders:       616
        35         36         46        153        101        120         75        110        108         48         58         54
--------------------------------------------------------------------------------------------------------------------------------------------------------------
```

The Smart Money Indicator (Commitment of Traders Data):

In 2018, Raymond Micaletti, a quantitative portfolio manager in the financial industry, released a paper on SSRN titled *"Want Smart Beta? Follow the Smart Money: Market and Factor Timing Using Relative Sentiment".* In this paper, Micaletti uses the Commitment of Traders data, which he normalizes into z-scores to then set up an equation of relative sentiment calculations to create a SMI, or "Smart Money Indicator". The futures contracts Micaletti uses are: the S&P500 Index, 10-Year Treasury Bond, 30-Year Treasury Bond. This SMI functions as positive or negative (+ / -) signal to either go long in the market strategy, or go all cash.

To calculate this SMI, he goes through a series of steps:

1. Calculate the ratios of the commercial and non-reportables net open interest to the total open interest of a certain contract.
2. Calculate the z score for both ratios for all N sample sizes (N = 39 weeks, 52 weeks, 65 weeks, 78 weeks, 91 weeks, 104 weeks)
3. $(Z\text{-score}_{(commercial,\ N)} - Z\text{-score}_{(non\text{-}reportable,\ N)}) / 2$
4. Calculate the Maximum (and Minimum, for 30-year Treasury Bonds) for M periods (M = 1 through 13 weeks)
5. $SMI = (SP_{MAX} - T30_{MIN}) + (T10_{MAX} - T30_{MAX})$
6. $SMI - SMI_{Cumulative\ Median}$

Within this SMI, the first two steps help transform the CoT data into a standardized number that shows how much the reportables and non-reportables are long each contract. Step Three builds a

reportable versus non-reportable sentiment component. Step Four transforms this component with a maximum or minimum lookback. Step Five creates two relative sentiment indicator functions and subtracts one from that other. These two relative sentiments, the first being an equity bullishness indicator, and the second being a yield curve indicator, are derived using common financial concepts of equity vs fixed income mechanics, and yield curve mechanics. The first is the S&P500 minus 30 Year. This portion of the equation helps calculate whether the market is more bullish equities than bonds. If it is positive, then that is bullish on equities; if negative, it is bearish on equities and bonds are being preferred. The second portion of the equation provides additional perspective on the interest in Treasury Bond futures, by subtracting the 30 year from the 10 year. This creates a longer duration or shorter duration preference indicator. If this portion of the equation is positive, it indicates that the open interest is more heavily on short duration, indicating general bullishness in the financial markets; if negative, this indicates that the interest is on long duration more than short duration, indicating bearishness in the short term in comparison to the long term.

Another key point to understand regarding the indicator is the outcome of using the M periods on the final result and what that means in terms of when and how often the indicator flips positive and negative. For the lower M period SMIs, because the M lookback is so low, the indicator can move more often and therefore the SMI flip signs more frequently. For longer period Ms, the numbers don't change as often since the look back is longer, and so the SMI changes more slowly. Because of this, you can safely assume that the longer period SMIs tend to react too slowly to potential drawdowns and to the market rebounds, and the shorter periods flip signs too quickly and can potentially be too volatile. Generally, this can be seen across the N periods as well. Anecdotally, the best performing SMIs are somewhere in the middle. See Figure 2 below.

Micaletti's results show that holding an asset while the SMI is positive, and going cash otherwise (SMI+), significantly outperforms holding while SMI is negative, and going cash otherwise (SMI-). In addition, his results show significant outperformance of the simple, but tried and true strategy of buying and holding a market index such as SPY. Finally, Micaletti provides extensive analysis within his paper to showcase the statistical significance of the SMI's outperformance.

Figure 2: SMI+ Table (Left) and SMI- Table (Right), 0 – 12 = 1 – 13 M, up to 8/23/2019

	39_weeks	52_weeks	65_weeks	78_weeks	91_weeks	104_weeks
0	4.539849	3.644969	3.774530	3.898936	4.379420	4.198782
1	5.676613	4.901377	7.589593	6.883613	6.521106	7.236572
2	5.301557	7.803039	7.641589	7.838004	8.931294	9.957934
3	9.182826	9.141907	7.676590	5.421681	6.218918	6.671936
4	10.425227	10.969056	10.727243	9.919397	8.261316	8.477363
5	7.974660	9.695875	10.081669	6.941032	6.519101	8.317432
6	8.465433	10.991625	7.951080	9.887490	8.746091	8.841065
7	7.985886	8.765701	6.890064	6.666472	7.146277	8.145221
8	7.929971	8.907985	7.397589	5.176839	5.666931	6.902610
9	6.533755	6.881690	6.034415	3.991342	4.877144	4.568215
10	7.315440	6.541802	6.833500	6.166271	5.896851	5.577137
11	6.092581	6.052383	6.385624	5.314189	4.481930	4.648292
12	6.691764	5.716331	4.995956	6.132343	5.593748	5.949963

	39_weeks	52_weeks	65_weeks	78_weeks	91_weeks	104_weeks
0	1.309763	1.631324	1.575329	1.525064	1.357743	1.416155
1	1.047478	1.213154	0.783458	0.863809	0.911828	0.821677
2	1.121581	0.762027	0.778127	0.758628	0.665763	0.597124
3	0.647527	0.650425	0.774579	1.096731	0.956135	0.891215
4	0.570359	0.542082	0.554301	0.599444	0.719755	0.701412
5	0.745628	0.613264	0.589796	0.856663	0.912108	0.714899
6	0.702401	0.540969	0.747839	0.601379	0.679861	0.672558
7	0.744579	0.678340	0.863000	0.891945	0.832059	0.730014
8	0.749830	0.667505	0.803793	1.148602	1.049268	0.861432
9	0.910063	0.864050	0.985369	1.489756	1.219182	1.301630
10	0.812819	0.908943	0.870144	0.964299	1.008356	1.066161
11	0.975962	0.982444	0.931174	1.118915	1.326689	1.279207
12	0.888574	1.040200	1.190188	0.969634	1.062995	0.999355

Although Micaletti analyzes various SMI calculations, not just for every N and M combination (78 combinations), but also various samples and methods (no options data, 40% percentile median, 3 year median, etc.), his primary results that he shows graphically and textually are the initial and main methodology (full dataset, regular cumulative median), with the parameters set to N = 78 weeks and M = 5 weeks. He notes that for the period of usable data, March 1995 - December 2017, this specific SMI+ would have cumulatively returned you 8.23 cents for every dollar invested, and SMI- would have returned .52 cents for every dollar invested. For our machine learning models, we will use the parameters he shows in his primary results.

Recreating the Smart Money Indicator:

In order to verify his results, we calculated our own SMI from the same dataset using Python's Pandas and Numpy packages. To ensure our own SMI recreation was accurate and similar, after preparing the SMI time series data, we calculated market cumulative returns (S&P500 index, in this case) for the same SMI selection that Micaletti uses in the paper to highlight the strength of the indicator (N = 78, M = 5). For our SMI, we return 8.25 cents for every dollar for SMI+, and SMI- returns .46 cents (Micaletti's Results are 8.23, and .52 Respectively). Given the similarities, we believe the minor differences are most likely due to slightly different methods to calculate the return. As a result, the replication of the SMI seems to have worked, and we use this N = 78 parameter and SMI commercial minus non-reportable

sentiment calculation (Step 3 listed above) in our normalizations and feature engineering of our CFTC Commitment of Trading features (S&P500, 10-year Treasury, and 30-year Treasury contracts) later on.

Figure 3: SMI calculated using Python

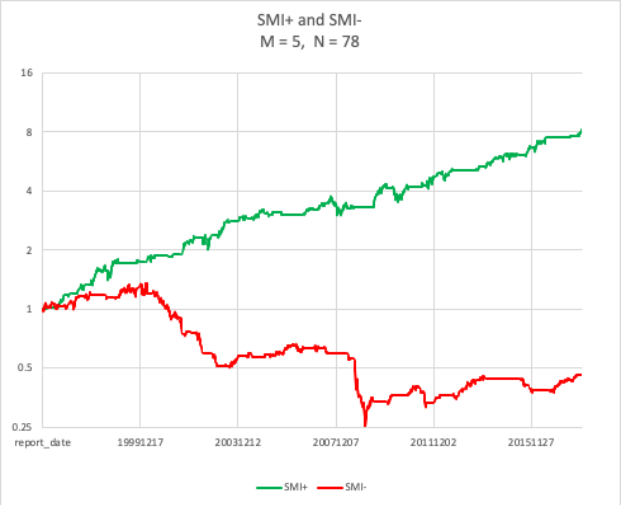
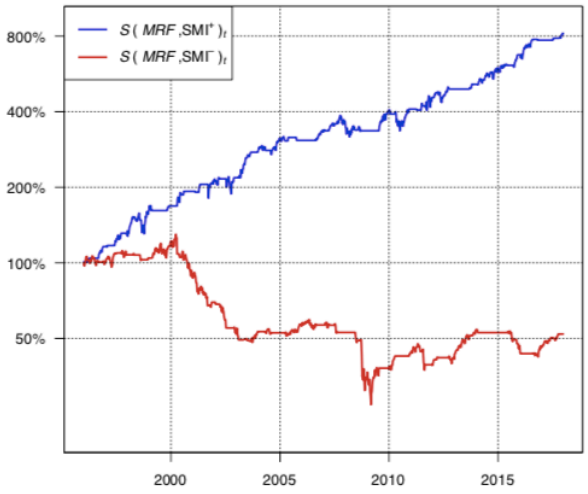


Figure 4: Raymond Micaletti's SMI



Data Source 2: MSCI EAFE Index, and Bloomberg-Barclays U.S. Aggregate Bond Universe Index:

For the second data source, we used historical returns compounded weekly for the MSCI EAFE Index, which is an emerging market / international market index, and the Bloomberg-Barclays U.S. Aggregate Bond Universe Index. These index price data were sourced from a Bloomberg Terminal. Both indices were then transformed into another positive or negative (+ / -) momentum indicator (nicknamed "Canary Indicators") based on the indices' returns. The indicator's nomenclature is in reference to its sourcing from a portion of the investment strategy highlighted in the paper *"Breadth Momentum and the Canary Universe: Defensive Asset Allocation (DAA)"*.

Canary Features and 13612W Momentum:

In this paper, authors Wouter J. Keller and Jan Willem Keuning expanded on their previously created Vigilant Asset Allocation Strategy through the incorporation of a "Canary" signal that signals the need to buy crash protection using a breadth momentum indicator, also referred to as 13612W momentum. The 13612W momentum name comes from how the momentum is calculated: 1 (month return), 3 (month return), 6 (month return), 12 (month return), W(eighted). Rather than using a more common momentum indicator such as a 12-month return minus the 1-month return momentum, this momentum uses a "breadth" of returns and weights them to create a positive or negative indicator of momentum. The formula is as follows:

((1-month return × 12) + (3-month return × 4) + (6-month return × 2) + 12-month return)) / 4

This type of calculation provides a more robust interpretation of momentum as it evaluates multiple timeframes and weighs the more recent returns more strongly to affect whether the momentum is positive or negative. Their preferred choices for their Canary indicator are VWO ***and*** BND (versus VWO or BND indicating the need for crash protection), which are the emerging market index fund and aggregate US bond index fund provided by Vanguard. To emulate these tickers, since our CFTC exists before the inception of these index funds, we use the indices that they track, the MSCI EAFE and Bloomberg Barclay Aggregate Bond Universe Index, respectively. In the paper, they use this crash protection canary signal on various mixes of portfolios, across various timelines, and they also implement selling to cash as you draw down, while buying back into the portfolio as your gains come back. For the purposes of our Machine Learning Market Timing model however, we only poach their concept of VWO

and BND momentum indicators as features, and we do not use their overall portfolio strategies to estimate potential returns, instead opting for a simpler strategy of just using S&P500 as our portfolio.

Data Source 3: Philadelphia Federal Reserve Anxiety Index

Finally, as our last data set for our machine learning model creation, we pull in the one quarter forward look ahead Anxiety Index forecast. This forecast is part of the Survey of Professional Forecasters, which is a survey conducted by the Federal Reserve of Philadelphia that surveys professional forecasters and academics on a variety of macro-economic variables and metrics. For this particular index, they ask the forecasters the probability of a recession in the current quarter and each quarter for 4 quarters ahead. The results are generally reported on in the middle of the quarter (For example, for Q3 2019, the results were posted in August 15th, 2019). As a result, using the current quarter (Q0) forecast probably isn't feasible or useful as it gets released halfway through the quarter, so instead we use the Q1 look-ahead forecast. We use this forecast in the following way. The look ahead forecast for Q4 that was released on August 15th 2019, will be used for all Q4 weekly data points within the model's data set.

I opted to include this feature in the data set as fear of impending recession can bleed into expectations of future stock prices and lead to a stock decline, and so could provide an additional type of crash protection that may not get captured in the Canary Indicators, as the Anxiety Index is a) a recession forecast vs just a momentum indicator, b) is an indicator of the economy's health vs an indicator of US fixed income or Emerging Market equities, and c) is a professional forward look on the economy (and indirectly the stock market) vs a current tracker of momentum. In addition, forecasters generally can see a recession coming probably slightly, and when in current recession, forecasters will predict the recession to continue for the next quarter, until conditions improve. For our feature, we will use the one quarter look ahead as a constant value for that full quarter that it is forecasting, as a probability / percentage value from 0 to 1.

Applying Machine Learning to Our Data Sources

Although the strength of the SMI's market timing looking back historically is undeniable, there are some doubts that the indicator is an effective predictor of returns in actual practice in the future. For instance, notice that in the table in Figure 2, for M = 5, N = 78, the outperformance diminishes to 694% cumulative return, as opposed to the 825% when excluding 2018 and 2019 data.

Although the sample is small for this 2018 and 2019 area, and therefore total cumulative return possible is lower, there does seem to be more variation in return across SMIs when including them. In addition, it is hard to pick or create a strategy on the SMI: which SMI choice do we choose? Do we just

build an average of them all? Weighted average? Choose a sample of the configurations that historically perform the best?

Because of this problem / issue, we began testing to see if any machine learning classifications and techniques could provide some complementary or even better predictive power to the SMI, and help us create a trading strategy that is viable.

To begin, we first excluded the data during 2018 and 2019 years as last and final testing sample. We then split the remaining data in half for out of sample testing. The three main model types we looked at were: Gradient Boosting Classifier, Random Forest Classifier, and Logistic Regression, all from Python's SKLearn packages. For the settings, we used the following settings to customize our model fitting:

> ***Machine Learning Classifiers****: max_features = 2, max_depth = 2, n_estimators = 4, max_leaf_nodes = 4, min_samples_split = 6*
>
> ***Logistic Regression****: intercept = False and class_weight = 'balanced'*

First, we re-ran some machine learning classifications on the full SMI replicated data frame available, to see if a model could create a stronger SMI from a full set of the SMI's themselves. This did not create any viable and successful models, obviously due to the overkill on the amount of features. Subsamples of various SMI's with different Ns and Ms did not create any models with predictive power either. Mixing the SMI with Machine Learning Models / Regressions did not create a model that had higher than 50% accuracy or outperformance.

So, we started from scratch and only took a more raw form of the CFTC CoT Data: the z-scores (after Step 3) for each contract, with a sample size of N = 78, plus a running cumulative median for those z-scores, akin to the last step in the SMI calculation. This led to 6 features (Z-Score 78 S&P, T10, T30, and their medians). We added in the other two data sources, the Canary Momentum Indicators (titled "momo_agg" and "momo_eem"), and the Recession Fear Index (titled "rf. See below for a summary description of the full dataframe used for our Machine Learning runs. For our first run, these are the results:

Figure 5: Dataframe "df_ml" Used For Machine Learning

	z_78_sp	z_78_sp_median	z_78_t10	z_78_t10_median	z_78_t30	z_78_t30_median	momo_eem	momo_agg	rf	stock_return	ml_ret
count	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1326.000000	1325.000000	1326.000000
mean	-0.044905	-0.106655	-0.044728	0.008558	-0.016118	-0.056323	0.009393	0.049228	0.149381	1.001794	0.586727
std	1.183642	0.472221	1.075054	0.155290	1.070416	0.116173	0.265450	0.059493	0.120418	0.023003	0.492607
min	-3.620454	-0.624772	-2.751720	-0.473390	-2.746649	-0.333149	-1.785071	-0.172875	0.040400	0.854136	0.000000
25%	-0.895113	-0.364014	-0.876624	-0.048241	-0.799491	-0.122359	-0.117817	0.008730	0.084700	0.990222	0.000000
50%	-0.096375	-0.114588	-0.073611	-0.002543	-0.046882	-0.034064	0.050458	0.050783	0.112100	1.003919	1.000000
75%	0.767240	-0.082766	0.714253	0.049011	0.808874	0.007386	0.182080	0.088612	0.158800	1.014388	1.000000
max	3.506666	2.534761	3.245378	1.182067	3.154247	0.984728	0.693460	0.259517	0.747800	1.131729	1.000000

Figure 6: Gradient Boosting Classifier Model Results (Sample)

```
Training Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                              50    211
1                              20    272
Accuracy Score of Training = 0.5822784810126582
Area under curve (AUC):  0.6509998425444812


Stock Return of Training Set = 2.1097088438953895
Stock Return of Training Set Preds Positive = 4.693819432519178
Stock Return of Training Set Preds Neg = 0.4494652753957996


Test Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                              50    201
1                              50    316
Accuracy Score of Test = 0.593192868719611
Area under curve (AUC):  0.5390242309450721


Stock Return of Test Set = 2.467675739217292
Stock Return of Test Set Preds Positive = 2.8158691627123433
Stock Return of Test Set Preds Neg = 0.8763460220006598


2018-2019 Confusion Matrix
Predicted pos or neg return  True
Actual pos or neg return
0                              35
1                              51
Accuracy Score of Most Recent 2 Years = 0.5930232558139535
Area under curve (AUC):  0.5047619047619047


Stock Return of Most Recent 2 Years  = 1.2262003830094492
Stock Return of Most Recent 2 Years Preds Positive = 1.2262003830094492
Stock Return of Most Recent 2 Years Preds Neg = 1.0
```

S&P return orange = 2.1097088438953895
ML return blue = 4.693819432519178

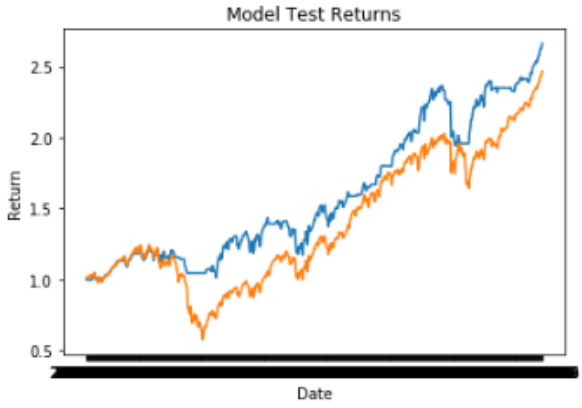
S&P return orange = 2.467675739217292
ML return blue = 2.8158691627123433

S&P return orange = 1.2262003830094492
ML return blue = 1.2262003830094492

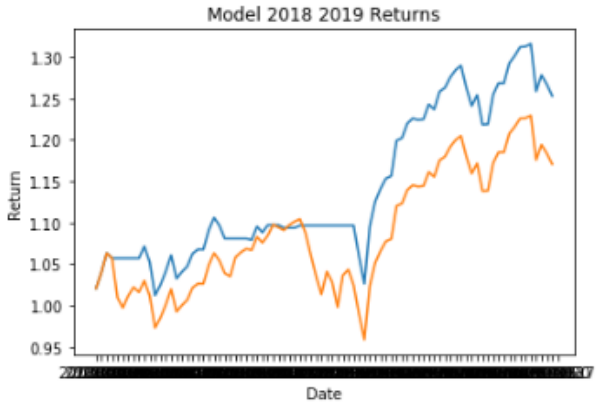
Figure 7: Random Forest Classifier Model Results (Sample)

```
Training Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                              138    123
1                               91    201
Accuracy Score of Training = 0.6130198915009042
Area under curve (AUC):  0.6147916338634335


Stock Return of Training Set = 2.1097088438953895
Stock Return of Training Set Preds Positive = 5.4701949831154675
Stock Return of Training Set Preds Neg = 0.3856734267073294


Test Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                               83    168
1                              105    261
Accuracy Score of Test = 0.5575364667747164
Area under curve (AUC):  0.5334454531600373


Stock Return of Test Set = 2.467675739217292
Stock Return of Test Set Preds Positive = 2.661900633052313
Stock Return of Test Set Preds Neg = 0.9270352576563641


2018-2019 Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                               12     22
1                               11     42
Accuracy Score of Most Recent 2 Years = 0.6206896551724138
Area under curve (AUC):  0.583795782463929


Stock Return of Most Recent 2 Years  = 1.170587301472578
Stock Return of Most Recent 2 Years Preds Positive = 1.2531095659170772
Stock Return of Most Recent 2 Years Preds Neg = 0.9341460103018958
```

S&P return orange = 2.1097088438953895
ML return blue = 5.4701949831154675



S&P return orange = 2.467675739217292
ML return blue = 2.661900633052313



S&P return orange = 1.2293459666361066
ML return blue = 1.3160105091481453

Figure 8: Logistic Regression Model Results

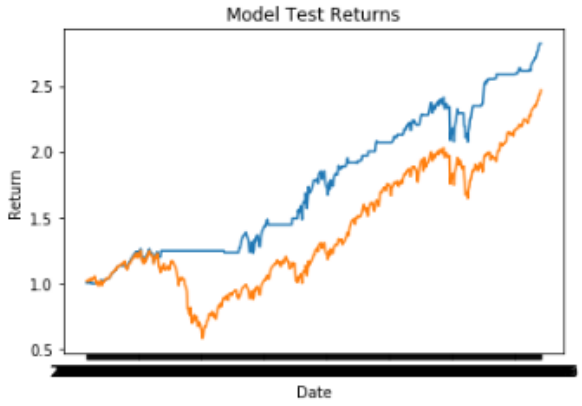
```
Training Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                              184     77
1                              162    130
Accuracy Score of Training = 0.5678119349005425
Area under curve (AUC):  0.583228362987456


Stock Return of Training Set = 2.1097088438953895
Stock Return of Training Set Preds Positive = 2.996558891885509
Stock Return of Training Set Preds Neg = 0.7040438449610799


Test Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                              147    104
1                              173    193
Accuracy Score of Test = 0.5510534846029174
Area under curve (AUC):  0.5562776217534234


Stock Return of Test Set = 2.467675739217292
Stock Return of Test Set Preds Positive = 2.8223825161516536
Stock Return of Test Set Preds Neg = 0.87432363441012


2018-2019 Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                               15     19
1                               24     29
Accuracy Score of Most Recent 2 Years = 0.5057471264367817
Area under curve (AUC):  0.586015538290788


Stock Return of Most Recent 2 Years  = 1.170587301472578
Stock Return of Most Recent 2 Years Preds Positive = 1.1662065291620922
Stock Return of Most Recent 2 Years Preds Neg = 1.0037564292438264
```

Logit Regression Results

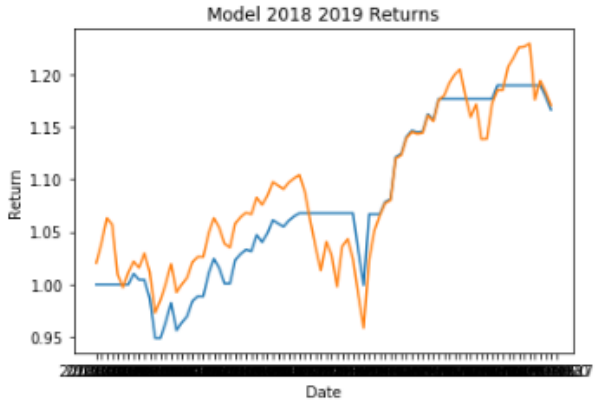
Dep. Variable:	ml_ret	**No. Observations:**	1326
Model:	Logit	**Df Residuals:**	1317
Method:	MLE	**Df Model:**	8
Date:	Fri, 08 Nov 2019	**Pseudo R-squ.:**	0.005988
Time:	11:35:01	**Log-Likelihood:**	-893.68
converged:	True	**LL-Null:**	-899.06
Covariance Type:	nonrobust	**LLR p-value:**	0.2153

	coef	std err	z	P>|z|	[0.025	0.975]
z_78_sp	0.1380	0.050	2.766	0.006	0.040	0.236
z_78_sp_median	0.3001	0.176	1.707	0.088	-0.044	0.645
z_78_t10	-0.1603	0.069	-2.332	0.020	-0.295	-0.026
z_78_t10_median	-0.6158	0.559	-1.101	0.271	-1.712	0.480
z_78_t30	0.1229	0.070	1.745	0.081	-0.015	0.261
z_78_t30_median	-0.2305	0.661	-0.348	0.727	-1.527	1.066
momo_eem	0.5036	0.220	2.293	0.022	0.073	0.934
momo_agg	3.5272	1.182	2.984	0.003	1.210	5.844
rf	0.7341	0.397	1.847	0.065	-0.045	1.513

S&P return orange = 2.1097088438953895
ML return blue = 2.996558891885509



S&P return orange = 2.467675739217292
ML return blue = 2.8223825161516536



S&P return orange = 1.2293459666361066
ML return blue = 1.1894400013907827

Preliminary Issues:

Returns vary from model to model and from testing sample to testing sample, but all outperform or are at least in-line with market performance (based on Sharpe Ratio), with some performing better than others. See Figure 9 below for Preliminary Results and Ensemble Methods (each on 1000 times run, except Logistic Regression) compared to the S&P500.

Figure 9: Average Returns and Standard Deviation Table

	In Sample (1992-2005)			Out Of Sample (2005 – 2018)			Held Back (2018-2019)		
Strategy	Avg Weekly Return Annualized	St. Dev	Sharpe Ratio	Avg Weekly Return Annualized	St. Dev	Sharpe Ratio	Avg Weekly Return Annualized	St. Dev	Sharpe Ratio
S&P500	1.447%	2.553%	.5668	1.716%	2.236%	.7674	1.754%	1.899%	.9236
GBC	4.241%	2.462%	1.723	1.672%	2.151%	.7773	1.902%	1.892%	1.005
Random Forest	4.545%	2.402%	1.892	1.647%	2.167%	.7600	1.894%	1.848%	1.025
Logistic Regression	2.603%	2.498%	1.042	2.130%	1.994%	1.068	1.697%	1.764%	.9620
Ensemble	5.921%	2.402%	2.465	1.972%	2.038%	.9676	2.434%	1.779%	1.368
Ensemble #2	4.293%	2.489%	1.725	2.048%	1.982%	1.033	3.810%	1.666%	2.287

When we re-ran the Gradient Boosting Classifier and Random Forest Classifier model fits on the same training data set multiple times, we had differing outputs, to a significant degree, especially in terms of return. On occasion, certain model fits (again, on the same training set) would occasionally produce returns on the test samples that did not outperform. So, what was going on? Well, whenever a model fit occurs for the two classifiers used, the model creates a small test sample (10% of the training sample that is given, unless otherwise configured) to identify the optimal levels of gradient boosting / decision trees[2]. In most modeling cases (or at least the ideal cases), this would not create a major issue in terms of accuracy of the model.

However, in this particular case, there are two reasons why this small out of sample testing going on during the model fitting of the training set can create significantly varying results: Firstly, our small data sample: Because the CFTC CoT data is only provided weekly, and for relatively small time period

[2] The Logistic Regression Model does not do the out of sample testing within the training dataset as the other two do, and as a result, the issue described above does not occur for it.

(roughly 25 years), we only end up with 1326 data points, and that is before any data set dividing. After we create two different test samples, our training data set is only 617 data points. This means that the ~62 data points it takes for out of the training set to adjust and fit the model on can be quite different and not completely represent the characteristics of the full training dataset, or even the full dataset as a whole. Secondly, and maybe more importantly: the model is not predicting the size of the return, only that it is positive or negative. With these two factors combined, whenever the model fitting occurs, the errors the model makes due to the idiosyncratic sample taken from the training set can have varying effects on the size of the return it will produce on the testing sample. Hypothetically, you can imagine that the errors will only lead to small drawdown occurring, or a small return being missed, depending on whether the error is a false positive or a false negative. Or, on the other hand, it could lead to big drawdowns or large returns being missed. How bad the errors truly are would depend on the fit, even if generally the accuracy is the same across each fit.

Approach for a Solution:

To help combat this deficiency in the preliminary approach, we developed some makeshift solutions in lieu of the fact that we can't use a longer range of data to test the true robustness and validity of the model and its features (which would be the ideal solution), due to the constricting factor of the CFTC CoT Data. Our solution was simple, yet effective in showing that the final model is significant in its outperformance and the underlying features used could be good information on market timing, with the caveat that more data to fit and test on would be more telling, and that there may be better models / configurations / feature engineering that we have not found yet.

First, we modified the training sample that the models are fitted on by feeding the classifiers an equal amount of Ones and Zeroes, a.k.a. positive returns and negative returns, to achieve the goal of providing a more valuable GBC model. Since the number of positive returns outnumber the negatives, we accomplished this by excluding all positive returns less than 50 basis points. The logic here is that if we need to provide a more equal weighting of Ones and Zeroes, the positive returns we should exclude should be the ones that are pretty much flat, as they should generally provide us less value in determining the coefficients / decision trees in the model. After this, our training sample became 292 positive returns, and 261 negative returns. However, this did not improve the performance significantly for the GBC model, so we turned to other possible solutions.

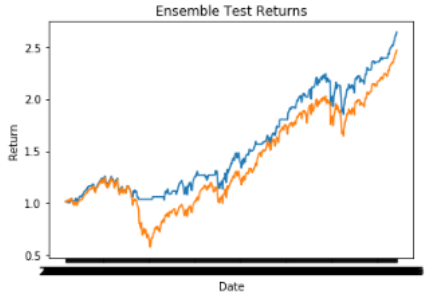
Second, rather than choosing only one of the models we tested, we chose an average of all three model's predictions, also known as the ensemble method. The equation is as follows:

IF ([Probability of Positive Return for GBC] + [Probability of Positive Return for Random Forest] + [Probability of Positive Return for LogReg]) / 3 > 0.5, THEN Positive Prediction, ELSE Negative Prediction.

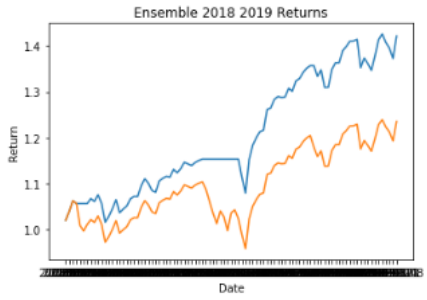
The reasoning behind using this as a stopgap measure against the issues we were getting described above is intuitive. If one fit ends up sub-par, the other two should theoretically compensate and the average should be a more robust model that outperforms. In addition, if the false positives and false negatives for one of the subpar fits are relatively close to the 0.5 threshold, then the other fits should help push the probability over the edge into the correct designation. See results for our first run below.

Figure 10: Ensemble Model Results (Sample)

```
Ensemble Test Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                              79    172
1                              94    272
Accuracy Score of Test = 0.5688816855753647
S&P return orange = 2.467675739217292
ML return blue = 2.6410179349024996
```




```
Ensemble Test 2018 2019 Confusion Matrix
Predicted pos or neg return  False  True
Actual pos or neg return
0                              10     26
1                               4     53
Accuracy Score of Test = 0.6774193548387096
S&P return orange = 1.2287391806629422
ML return blue = 1.4135690148023026
```



As you can see, the ensemble method produces superior returns in both samples, and better than any fit on its own. If we use an accuracy benchmark of percentage of "ups" divided by the total, the first out of sample benchmark accuracy would be 59%, and the second out of sample benchmark accuracy would be 61%. As you can see, the first ensemble method we developed did not fully outperform in all measures we could employ (the first out of sample accuracy benchmark), but does succeed in some

measures, such as the second out of sample accuracy benchmark, and out-performance of our returns benchmark, a buy-and-hold strategy of the S&P500, on both out of sample datasets tested.

To address the variation in fits, we re-run the ensemble model fit 1000 times to see how many times the model outperforms a standard buy and hold strategy, and what the average outperformance is. Results are below, and you can see the Appendix for full details. The first table in each column is the description of the accuracy of the 1000 models, the second is the buy and hold S&P Strategy (does not change other than infinitely small differences in the calculations due to Numpy rounding), and the third is the S&P return using the model's timing. The tables with suffix 2 (2nd column) are the 2018-2019 test sample. As you can see, the model has an average accuracy of 57%, and 62% respectively, outperforms on average by 27% and 11% respectively, and has an outperforming fit at least 75% of the time (776 times out of 1000 and 939 times out of 1000 respectively). When re-running this 1000-times loop for fitting, the outputs remain relatively the same.

Figure 11: Ensemble Model Results, 1000 Times

```
ensemble_accuracy_test
count    1000.000000
mean        0.574849
std         0.011357
min         0.513776
25%         0.567261
50%         0.575365
75%         0.583468
max         0.606159
dtype: float64
```

```
ensemble_accuracy_test_2
count    1000.000000
mean        0.625581
std         0.030216
min         0.526882
25%         0.612903
50%         0.623656
75%         0.645161
max         0.698925
dtype: float64
```

```
ensemble_test_pl_sp
count    1.000000e+03
mean     2.467676e+00
std      8.886228e-15
min      2.467676e+00
25%      2.467676e+00
50%      2.467676e+00
75%      2.467676e+00
max      2.467676e+00
dtype: float64
```

```
ensemble_test_pl_sp_2
count    1.000000e+03
mean     1.228739e+00
std      6.886827e-15
min      1.228739e+00
25%      1.228739e+00
50%      1.228739e+00
75%      1.228739e+00
max      1.228739e+00
dtype: float64
```

```
ensemble_test_pl_ml
count    1000.000000
mean        2.735051
std         0.365926
min         1.466132
25%         2.497260
50%         2.757407
75%         2.982116
max         4.014735
dtype: float64
```

```
ensemble_test_pl_ml_2
count    1000.000000
mean        1.330631
std         0.083468
min         1.166757
25%         1.243757
50%         1.325203
75%         1.394727
max         1.533225
dtype: float64
```

Ensemble Model Version 2:

Although we had created some minor outperformance in our ensemble model and ran the fits 1000 times to show that it outperformed 75% or greater on our two out of sample datasets, that still left significant doubt on whether any random fit was going to be strong or not. Running this 1000 times is not viable, due to the excessive computational time (although not prohibitively long yet, with more data could it could be), and a 75% chance of outperformance on any one fit is not a strong enough probability that anyone would definitely agree to trade on. In addition, the standard deviation of the cumulative returns based on fit was decently large, at 36% for the out-of-sample returns. We searched for improvements, and came up with a simple solution.

If we ran each of the two classifiers that were producing varying results 10 times and appended them into a prediction data frame, and then took the median of that data frame, and averaged that median with the logistic regression results, then we could theoretically "eliminate" the bad fits that were corrupting our models. If we get outperformance 75% of the time on the fits, then the median prediction of 10 times would most likely be the prediction with a superior fit. Upon running, we found that it was, and it slightly improved our average return on the 1000-run test, with less variation in returns from the fits. For our first out of sample data, we returned an average 29% above buying and holding the S&P500, with outperformance occurring 980 times out of 1000. For the second sample, we returned an average of 26% above buying and holding the S&P500, with outperformance occurring 1000 times out of 1000. See figure 12 below.

Figure 12: Ensemble Version 2 Results, 1000 Times

```
ensemble_accuracy_test
count    1000.000000
mean        0.569908
std         0.004394
min         0.555916
25%         0.567261
50%         0.570502
75%         0.572123
max         0.583468
dtype: float64
```

```
ensemble_accuracy_test_2
count    1000.000000
mean        0.660215
std         0.017873
min         0.602151
25%         0.645161
50%         0.666667
75%         0.677419
max         0.709677
dtype: float64
```

```
ensemble_test_pl_sp
count    1.000000e+03
mean     2.467676e+00
std      8.886228e-15
min      2.467676e+00
25%      2.467676e+00
50%      2.467676e+00
75%      2.467676e+00
max      2.467676e+00
dtype: float64
```

```
ensemble_test_pl_sp_2
count    1.000000e+03
mean     1.228739e+00
std      6.886827e-15
min      1.228739e+00
25%      1.228739e+00
50%      1.228739e+00
75%      1.228739e+00
max      1.228739e+00
dtype: float64
```

```
ensemble_test_pl_ml
count    1000.000000
mean        2.752434
std         0.136295
min         2.337150
25%         2.658454
50%         2.748445
75%         2.849418
max         3.137180
dtype: float64
```

```
ensemble_test_pl_ml_2
count    1000.000000
mean        1.491336
std         0.018083
min         1.353590
25%         1.484953
50%         1.497831
75%         1.498301
max         1.547869
dtype: float64
```

Final Takeaways and Room for Improvement:

Our initial searches for trading strategy that can outperform buying and holding the S&P500 and where we ended up are very different. Initially starting as a project to determine if we can improve upon the Smart Money Indicator, we ended up taking features from a variety of sources, all that provide different, but useful information on the market. And yet even still, with a wide range of sources, feature engineering, and model types deployed, building a model that can consistently outperform the market by all measures we can use is still elusive. Overall, I believe that the model in its current state has a legitimate chance for success, but still most definitely has room for improvement and fine-tuning. As more data becomes available, I believe the model can become stronger.

This capstone project has been extremely useful in expanding my personal knowledge in Python, in machine learning, in critical thinking, and more. Although it was very challenging and time-consuming tackling bugs in code and other set-backs, I found the project extremely rewarding. Thank you for reading through my paper!

Resources / Links / Citations:

Smart Money Indicator / CFTC:

Raymond Micaletti, Want Smart Beta? Follow the Smart Money: Market and Factor Timing Using Relative Sentiment, SSRN, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3164081

Raymond Micaletti, https://alphaarchitect.com/2019/02/08/relative-sentiment-a-unique-market-timing-tool-that-isnt-trend-following/

U.S. Commodity Futures Trading Commission, https://www.cftc.gov/sites/default/files/idc/groups/public/@commitmentsoftraders/documents/file/disaggregatedcotexplanatorynot.pdf

U.S. Commodity Futures Trading Commission, https://www.cftc.gov/MarketReports/CommitmentsofTraders/HistoricalCompressed/index.htm

Canary Indicator / Defensive Asset Allocation Strategy:

Wouter J. Keller, Breadth Momentum and the Canary Universe: Defensive Asset Allocation, SSRN, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3212862

TrendXplorer, Breadth Momentum and Vigilant Asset Allocation (VAA), https://indexswingtrader.blogspot.com/2017/07/breadth-momentum-and-vigilant-asset.html

The Anxious Index (Recession Fear Index):

Federal Reserve Bank of Philadelphia, The Anxious Index, https://www.philadelphiafed.org/research-and-data/real-time-center/survey-of-professional-forecasters/anxious-index

Miscellaneous Readings Used:

Sanjiv R. Das, Karthik Mokashi, and Robbie Culkin, Are Markets Truly Efficient? Experiments Using Deep Learning Algorithms for Market Movement Prediction