Passive Portfolio Management

Predicting excess returns with machine learning

Satya Prakash, Jared Berry, George Krug, Maria Matsira

06.15.2019

ABSTRACT

It is generally accepted that the individual investor can expect more reliable, less volatile returns by allocating resources into funds that cover a wide-swath of the market, rather than chasing down returns on individual stocks. This most commonly manifests as allocating those resources into an index fund such as the S\&P 500, which historically outperforms bear markets and corrections in the long-term. We build on this heuristic, and propose a strategy of passively holding a well-diversified selection of assets (i.e. those listed in the S\&P 500), along with leveraging leading indicators of predicted performance of underlying assets to adjust holdings accordingly. Using financial markets and fundamentals data, we construct a machine learning model for classifying positive returns in a basket of S\&P 500 stocks, relative to performance of the market. We rely on purely financial and time-based features informed by the finance literature, and employ a number of cross-validation frameworks to generate appropriate out-of-sample performance metrics. Leveraging a number of popular machine learning models with our curated data set, we find optimal predictive performance is achieved using finely-tuned ensemble learning frameworks estimated at the ticker-level, restricting the size of out-of-sample validation folds to reflect the presentation of the prediction problem in the real world. Using the results of these models, we provide users an interface to view predicted probabilities of out-performance in a prevailing time horizon, along with quality-of-prediction scores, for all of the tickers in our basket.

BACKGROUND

The mutual fund industry has seen tremendous inflows of capital into passively managed funds that seek to simply match the returns of benchmark indices such as the S&P 500, despite reputed financial research illuminating a number of relatively simple and backtested portfolio construction strategies that outperform the market over long periods. While investing in a passive fund may be an appropriate strategy for a large number of people, it forgoes all attempts of optimization in exchange for low cost efficiency, and the comfort of knowing that returns will match the market as a whole. Based on this consideration, the project will employ the full data science pipeline by attempting to build a tool that can help optimize the traditional passive fund without demanding excess expenses. Furthermore, the goal of the project will be to engineer a feature space containing financial metrics associated with potential alpha—returns in excess of the S&P 500—and to train a machine learning model to predict stock performance relative to its benchmark index.

THE PROBLEM

We recognize that, 1) index funds are incredibly difficult to beat using active investing approaches, and 2) predicting stock prices is an incredibly difficult task in and of itself. The random walk theory prescribes that stock prices can be generally modeled as following a random walk process, which would imply that the only useful information for predicting a stock's value tomorrow is the value today. Despite this, countless quantitative hedge funds and individual investors seek to optimize portfolio holdings by pouring money into stocks predicted to outperform their peers. Further, given that so much of today's market participants are underscoring their decisions with rigorous quantitative methodology/AI, we hope to divine, to the extent possible, some signal through the noise, and arm investors with an additional leading indicator for stock performance in a monthly window.

Moreover, recognizing the dominance of index funds for reliable returns in the long-term, we hope to pair the long-term viability of a passively managed fund for reliable returns with signal derived from machine learning prediction of those returns to better leverage such a passive fund. Examining returns relative to an index fund over a longer time-horizon contextualizes this firmly in the passive stock fund management space. Traditional time-series models, however, tend to perform particularly poorly in long-horizons, and a random-walk model would be completely untenable at a horizon of, say, one month ahead. As such, using more robust machine learning approaches, particularly those that are agnostic about specification and functional form, are likely to overcome these limitations.

HYPOTHESIS

Recognizing that markets today are driven, in large part, by high-frequency and algorithmic training, we hypothesize that, despite the inherent noisiness surrounding, and difficulty predicting, stock price movements, using machine learning should allow us to exploit some signal in the noise. Using a curated selection of features developed from financial markets and fundamentals data informed by the finance literature, along with machine learning algorithms, that are agnostic about the form of the relationship between our features and a future returns target, can we predict returns in excess of a benchmark index?

In order to more carefully frame our exercise, we seek to predict returns in excess of the S&P 500 and intend to frame these returns in the context of a classification problem. Rather than predicting absolute returns, we seek to predict whether a given asset in the S&P 500 basket will outperform the S&P 500 in a prevailing time-period. The resulting predicted probabilities of outperformance, along with metrics about the quality of that prediction, could serve as useful leading-indicators to optimize a passive stock fund.

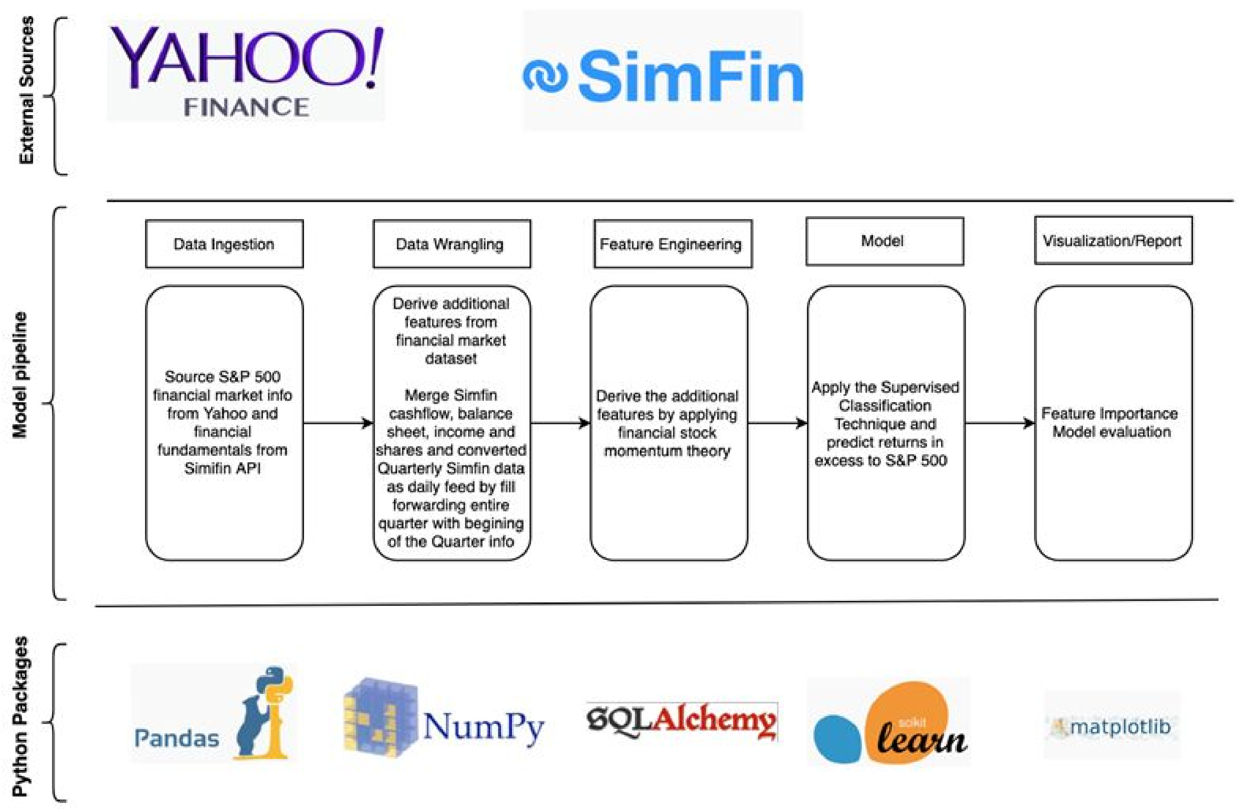
PROJECT OVERVIEW

Before attending to a thorough discussion of our project in the context of the data science pipeline, it is appropriate to discuss the project at a high-level to inform the purpose and underlying research questions we seek to address.

- Purpose: Provide passive stock fund investors additional, leading indicators about relative performance about the stocks contained in their portfolio to adjust holdings accordingly, informed by machine learning algorithms run on well-established financial features derived from markets and fundamentals data.
- Unit of Analysis: Ticker-level data, with financial characteristics available at a business-daily frequency, since 2011. Our instance is the ticker-level data for one of the 500, S\&P 500 listed stocks at a given point in time.
- Research Questions:
    - Are financial features derived from markets and fundamentals data, alone, predictive of relative returns of S&P 500 stocks?
    - Which features are most relevant for informing these predictions?
    - Are the characteristics of other assets in the basket relevant for prediction (i.e. does panel-level prediction outperform ticker-level prediction)?
- Hypothesis: Derived features from financial and market data will be predictive of relative returns for individual stocks in a panel-level data set of S&P 500 stocks.

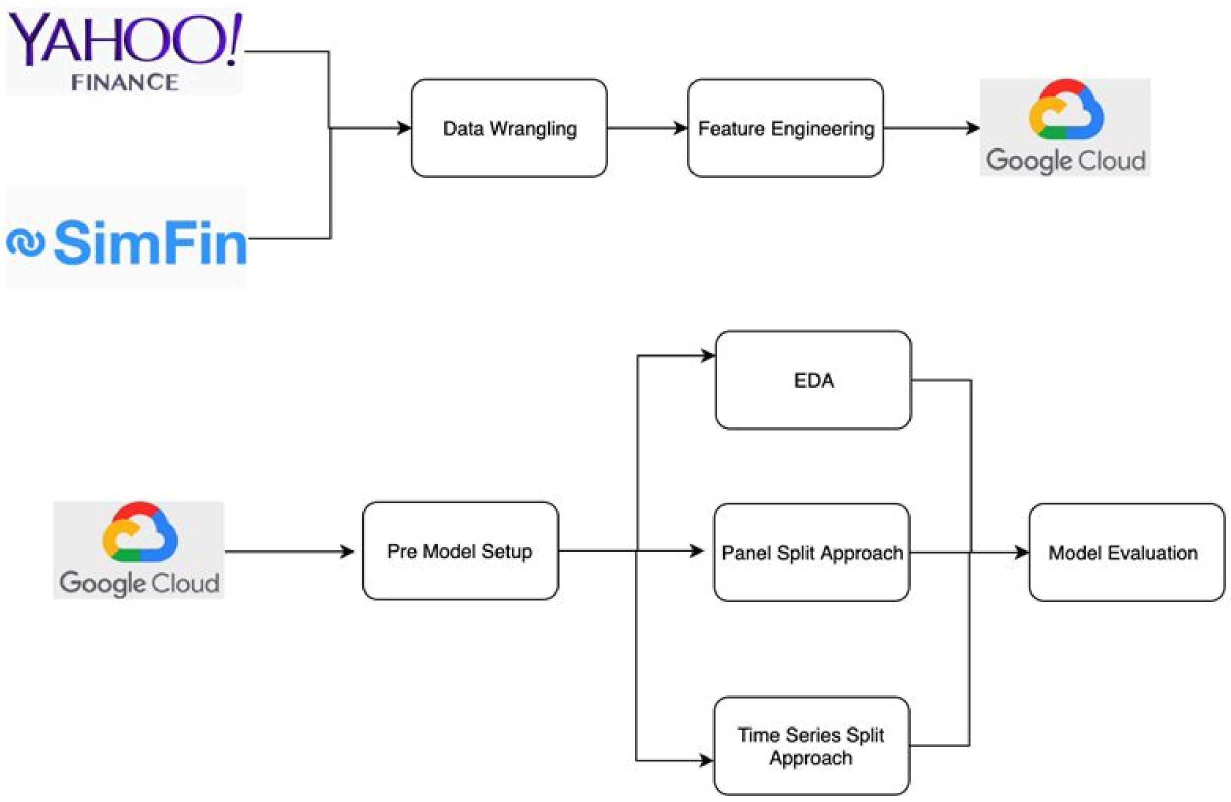
PROJECT ARCHITECTURE

Below, we will attend in detail to our pipeline, and the steps therein to address our research questions and hypothesis.


External Sources
YAHOO! FINANCE
SimFin
Model pipeline
Data Ingestion
Data Wrangling
Feature Engineering
Model
Visualization/Report
Source S&P 500 financial market info from Yahoo and financial fundamentals from Simifin API
Derive additional features from financial market dataset
Merge Simfin cashflow, balance sheet, income and shares and converted Quarterly Simfin data as daily feed by fill forwarding entire quarter with begining of the Quarter info
Derive the additional features by applying financial stock momentum theory
Apply the Supervised Classification Technique and predict returns in excess to S&P 500
Feature Importance Model evaluation
Python Packages
Pandas
NumPy
SQLAlchemy
scikit learn
matplotlib

DATA INGESTION

We initially explored Yahoo! Finance, IEX, Quandl, Wharton Research Data Services (WRDS), Tradier, Alpha Vantage, and SimFin to source the market and fundamentals data necessary to perform our analysis. Based on pricing and ease of use, we chose to use the Yahoo! Finance API for market data, and SimFin or fundamentals data. The ingestion framework for both of these sources are detailed below.



Yahoo Ingestion

We chose Yahoo! Finance as our primary source for financial markets info, largely due to easy integration with Python through pandas DataReader. Each day, we pull all listed S&P 500 stocks from the Yahoo API since the most recently pulled date and ingest directly into a Postgres database hosted on the Google Cloud. We note that the tickers listed on the S&P 500 index do change over time (i.e. DWDP was removed from the S&P 500 list on June 3rd), which necessitates daily scraping of the latest list of S&P 500 tickers to ensure the list is populated correctly, relative to the day we pull. Raw Yahoo! Finance data contains a small selection of standard financial markets features (’High’, ’Low’, ’Open’, ’Close’, ’Volume’, ’AdjClose’ [or adjusted close]), and we derive some additional time-based features such as ’Dayofweek’, ’Dayofyear’, ’Is_month_end’, ’Is_month_start’, ’Is_quarter_end’, ’Is_quarter_start’, ’Is_year_end’ and ’Is_year_start’. These help to group data by day for standardization, and allow us to capture some measure of seasonality in the stock prices. In addition, we pull the SPDR S&P 500 ETF (SPY) index (an ETF which tracks the S&P500 and serves as our proxy for the S&P 500 from here on) to generate targets, using the same techniques as above. The Postgres database these data, along with the SimFin data discussed below, are stored in serves as our WORM store for feature engineering and modeling pipelines further on.

SimFin

Fundamentals data is remarkably difficult to source without steeply-priced subscriptions to providers like Bloomberg and Quandl. SimFin provides free access to an API with data for 2,470 companies, covering over 266,943 financial statements, scraped directly from freely-available 10-K regulatory reports. Using the SimFin API, we pull in quarterly cash flow, balance sheet, income, and shares statements for listed S&P 500 companies from 2011:Q1 through 2019:Q2. Automated pulls are not set up for this source because fundamentals data is only available at a quarterly frequency. We pull as needed, for this analysis, but have functionality set up to hit the API in an automated fashion, should that functionality be necessary further on. This quarterly data is standardized in the code which conducts the pulls, and is presently stored in flat files, which are loaded into the SimFin wrangling code detailed below to transform them into a synchronous, daily-frequency dataframe which is stored in Postgres. Prior to wrangling, the SimFin fundamentals data sets boast a combined 127 features.

DATA WRANGLING

While the Yahoo! Finance data is quite tidy and ready to go directly out of the API, the SimFin data requires significantly more effort to normalize. Our quarterly SimFin data, as it is pulled from the API, poses a problem for merging into the business-daily Yahoo! Finance data: 1) We do not have information about when earnings reports and statements are released; 2) Quarter-end dates do not necessarily coincide with business days; 3) In order to avoid a significant amount of missing data, our quarterly data needs to be responsibly converted to daily. Unfortunately, we do not have a solution for this first issues, so can only merge the data on quarter-end dates.

We can address points 2) and 3) simultaneously. After pulling all quarterly statement data into memory and merging on ticker, we create a separate data set containing a single column with all dates from 2011:Q1 to the maximum date in the quarterly data set. We merge each ticker to this data frame to stably match the quarter-ends that are populated for each ticker to a daily data set, without needless duplication of entries. We then forward fill the quarter-end values through to the next populated quarter, populating a daily-frequency data set of all fundamentals features. While the financial statements are with respect to the prior quarter (i.e. the 2011:Q1 statements capture what occurred from 01/01/2011 to 03/31/2011), we would not have that information available during the quarter, so use it for the next quarter when that information would be available. As such, we prevent the leakage of future data into the past, an issue we have to be immensely cognizant of in this exercise.

Additionally, since the data set is stored at the daily-frequency, we no longer need to worry about quarter-end dates taking place on non-business days, and can perform a stable merge onto the Yahoo! Finance data. We also remove sparse features, especially considering the incidence of missing values is compounded when converting the data to a daily frequency. Features which are missing for more than 15% of total observations are removed, though far fewer are captured in our final model (see Feature Selection).

As all data is stored at the ticker-day frequency, merging is straight-forward across all data sources, so long as keys are aliased correctly for merges. We merge momentum features onto the Yahoo! Finance data, and then merge the daily SimFin data onto this. As our data is real-world data, updating daily in real time, we note that the raw data set contains 1.04 million observations of 87 unique features across both data sets, for 506 unique ticker, as of June 5, 2019

FEATURE ENGINEERING

Based on the basic set of daily stock data ingested from the yahoo api, we were able to engineer additional features that provide insight into stock momentum. In general terms, stock momentum attempts to capture stock strength based on the price movement over a certain time interval. In context of the project, momentum is simply the price return over a certain time period.

The decision to explore stock momentum features is based on research that shows that portfolios constructed with the highest ranking momentum stocks, i.e. the ones with best price returns, outperform weak momentum portfolios and also the S&P 500 (Gray & Vogel, 86). Moreover, portfolios constructed to capture intermediate momentum with a twelve month look-back window outperform both short term and long term momentum stock selection strategies (86). This finding aligns very well with the project goal since the model will be trained with target variables looking one month ahead.

Therefore, to generate features that intend to capture intermediate term momentum, rolling price returns will be calculated on a monthly and yearly basis. The basic yahoo dataset has a time series 'Adjusted Close' feature for each stock. The 'Pct_Change_Monthly' and 'Pct_Change_Yearly' momentum features are calculated by taking the percent difference of the adjusted close at time t and the adjust close at time t

– n, where n is the number of look back days for the period. Since each date in the dataset is a trading day, n will be 252 for a yearly look back and n will be 21 for a monthly period.

Once we have price returns with a look back of one year and a look back of one month, the rankings can then be calculated to capture the research supported benefits of intermediate term momentum and short term momentum (Long term momentum which looks back sixty months as defined by Gray & Vogel (81) was left out since our dataset only spans from 2011-present). The momentum (price return) rank features are labeled 'Return_Rank_Yearly' and 'Return_Rank_Monthly' for the intermediate and short term basis, respectively. For each day in the time-series dataset, the percent change values of each stock listed in the S&P 500 are sorted from greatest to least return. The momentum rank is simply the index position obtained from the sorted list. Thus, the top momentum ranked stocks have distinct integer values closer to zero while the weaker momentum stocks have rank values closer to five-hundred.

In addition to the momentum rank features above which offers insight into relative price performance, a more robust momentum strength metric can be generated by taking into account price path quality. Momentum quality that accounts for price return and path smoothness is based on the Frog-In-The-Pan (FIP) hypothesis, which posits "a series of frequent gradual changes attracts less attention than infrequent dramatic changes. In support of the FIP hypothesis, given a set of strong momentum (high returning) stocks, choosing subsets with smoother price paths outperform subsets composed of more choppy, volatile paths (Da, Gurun and Warachka (2012)). Investors therefore underreact to continuous information" (Da, et al 2012). The purpose of generating a momentum quality score is to capture high returning stocks that receive less market attention, which can be a source of alpha.

To quantify path quality in a given time period, the percentage positive returning days relative to the percentage of negative returning days will be used to signal path smoothness. With this in mind, combining path quality with absolute price return, we arrive at the following equation adapted from Da, et Al that can be used to represent momentum quality:

MomentumQuality = +-(stock_return)(pct_up_days – pct_down_days)

where stock_return is the price return based on a predetermined number of look back days, pct_up_days is the percentage of positive returning days, and pct_down_days is the number of negatively returning days. The above is used to generate the

‘Momentum_Quality_Yearly’ and ‘Momentum_Quality_Monthly’ features.

In addition the research supported momentum features, the group decided to also include stock volatility, relative strength index (RSI), and also the trailing one month return of the SPY exchange traded fund that tracks the performance of the S&P 500. Volatility is simply the standard deviation of the stock price over a given time period divided by the stock price. The RSI is an oscillating value between zero and one that accounts for price change and speed. Last, the trailing S&P return feature is included to help signal systematic trends in the market.

We also construct a market beta feature to more empirically capture the relationship between our ticker-level returns and the S\&P 500 index. The beta feature is well-established in the finance literature (specifically as it is used in the CAPM pricing model), and is computed by scaling the covariance between the market and an individual asset by the variance of the market, and so behaves similarly to a regression coefficient. We compute this feature on a rolling-basis, with the previous 21 trading days as the horizon to calculate the beta for a given day.

In addition to features generated using the financial markets data, we leveraged the fundamentals data sourced through SimFin to construct common financial ratios that are typically used to value companies or provide indicators of financial health and performance. We construct: earnings per share (EPS), price-earnings ratio, debt ratio, debt-to-equity ratio, and return on assets (ROA) features. All of these features (with the exception of debt-to-equity) are constructed using both financial market and fundamentals data, so still vary at the business-daily frequency. Lastly, we construct a number of returns relative to a number of different horizons and add smoothed (using exponential moving average smoothing) variants of all of the above features as potential alternatives to the noisier originals.

We perform standard-scaling to normalize units across features, at the ticker-level for the ticker-level modeling framework and at the day-level for the panel-level framework. Missing values are replaced with zeros, which reflect the mean in scaled features.

- Pct_Change_Class
- Rolling_Yearly_Mean_Positive_Days,
- Rolling_Monthly_Mean_Positive_Days
- Rolling_Monthly_Mean_Price
- Rolling_Yearly_Mean_Price
- Momentum_Quality_Monthly

- Momentum_Quality_Yearly
- SPY_Trailing_Month_Return
- Pct_Change_Daily
- Pct_Change_Monthly
- Pct_Change_Yearly
- RSI
- Volatility
- Yearly_Return_Rank
- Monhtly_Returns_Rank
- Pct_Change_Class
- Rolling_Yearly_Mean_Positive_Days
- Rolling_Yearly_Mean_Positive_Days
- Rolling_Monthly_Mean_Positive_Days
- Rolling_Monthly_Mean_Price
- Rolling_Yearly_Mean_Price
- Momentum_Quality_Monthly
- Momentum_Quality_Yearly
- SPY_Trailing_Month_Return

COMPUTATION AND ANALYSIS - TARGET LABEL CALCULATION

Of paramount importance to our modeling pipeline and predictive analysis is proper framing of our target. As stock return prediction is inherently a time-series problem, framing our target in a way that prevents leakage and accurately reflects the state of the world we seek to predict is crucial for useful results. Returns are, by definition, calculated as the percentage change in the price of a stock from one point in time to another. In our case, we calculate returns on the adjusted close (*Ad jClose*) feature from the Yahoo! Finance data set. Thus, the raw, n-day ahead return for stock *i* at time t is calculated as:

$$return_{t+n,i} = \frac{AdjClose_{t+n,i} - AdjClose_{t,i}}{AdjClose_{t,i}}$$

Returns on the S&P 500 are calculated using the same formula, at the same point in time, t, over the same horizon t+n. As we seek to frame this as a problem of relative performance of the ith stock with respect to the index, our target is calculated as the difference of the return of the ith stock and index return, calculated as:

$$relative\_return_{t+n,i} = return_{t+n,i} - return_{(t+n,S\&P\ 500)}$$

Moreover, in order to abstract away from a standard continuous regression framework, we simplify our target as a binary variable to leverage classification models and associated performance metrics. As such, our final target is calculated as follows:

$$target_{t,i} = \begin{cases} 1\ if\ relative\_return_{t+n,i} > 0 \\ 0\ if\ relative\_return_{t+n,i} \leq 0 \end{cases}$$

We explored a number of possible targets (rank, moving-average returns, absolute returns as opposed to relative) and horizons (1-, 5-, 10-, and 21-days ahead). We settled on relative returns to more closely contextualize our exercise in the passive fund space. For the purpose of this analysis, we use relative returns over a 21-day horizon (i.e. month-ahead in terms of trading days). The reason for this horizon is threefold: 1) Month-ahead return is a fairly standard horizon, (as opposed to an arbitrary selection of trading days ahead); 2) horizons greater than one day ahead are significantly harder for workhorse time-series models to perform on (particularly random-walk and auto regressive models), justifying the use of a more complex approach; and 3) given our emphasis on passive investment, 21-days ahead is a reasonable horizon for rebalancing, rather than predicting and rebalancing each day-ahead. This also provides us with a natural hold-out set upon which we can develop our product. At any given point in time, we cannot know what will prevail over the next month, so can use models fitted on data for which we do have the target to predict what will happen now.

Attending briefly to the time-series properties of our target, we conduct Augmented Dickey-Fuller tests at the ticker-level to determine whether our series exhibit unit root processes. We find that, for all tickers in our sample, we can soundly reject the null hypothesis that our target contains a unit unit, implying a degree of stationarity and predictive value contained in lagged variants that is essential for us to expect any kind of result moving forward.

EXPLORATORY DATA ANALYSIS & FEATURE SELECTION

Exploratory Data Analysis (EDA) is the phase where a preliminary, but also crucial first "dive" into the data takes place. In his *Experimental Design and Analysis* (2018), Seltman calls it “a critical first step in analyzing the data” and explains that “the main reasons to perform it, is to detect mistakes, check assumptions, make a preliminary selection of appropriate models to determine relationships among the explanatory variables, and, finally, assess the direction and size of the relationships between exploratory and outcome variables”.

As explained in the chapters above, for our project, EDA needed to take place in two stages: on an aggregate level, after the wrangling of the data collected from different sources, so as to investigate the relationships between different features; and on a approach-specific level to account for both the panel and the ticker-level analysis of our data.

In the initial exploration of our features’ relationships, 45 features (a combination of categorical and numeric variables) were included. For these, summary statistics were extracted, and several correlation matrices were plotted to determine feature relationships.

During EDA, we detected multicollinearity among certain features, mostly the ones constructed from raw ones, like in the case of the five Yahoo!Finance features, ‘High’, ‘Open’, ‘Low’, ‘Close’ and ‘Adjusted Close, and the momentum indicators. Based on the detection of one-to-one relationship between features, as illustrated in the Yellowbrick Pearson correlation matrix below (Figure 2), we were able to soundly exclude a preliminary set of features from our analysis.

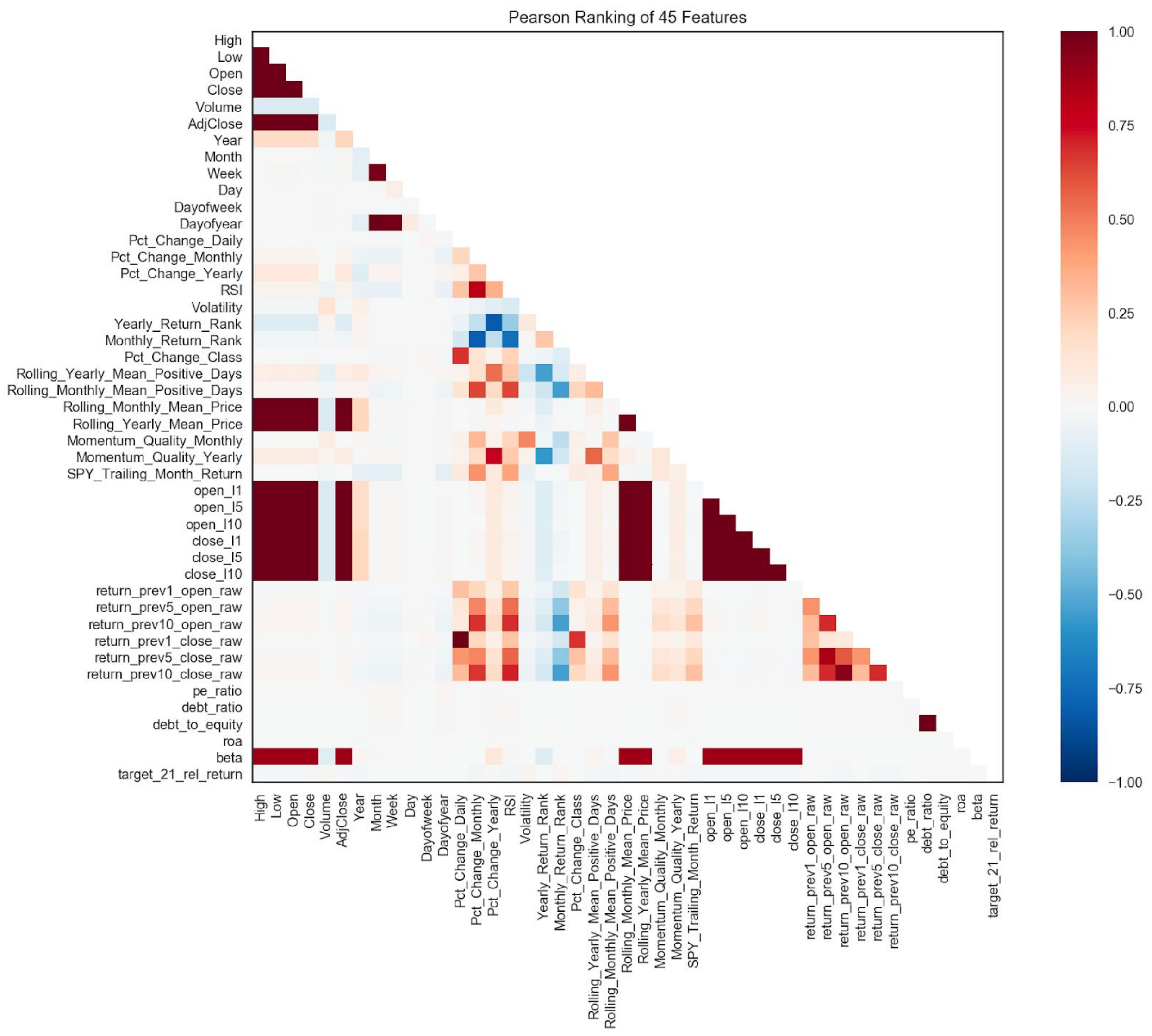


Figure 2. Pearson feature correlation matrix

Since we explore this hypothesis at both the ticker- and panel-levels, we inform our feature selection through both our exploratory data analysis and first-pass modeling, with plots of both regression models. The panel-level feature importance plots reflect feature importance in the aggregate, whereas the ticker-level feature importance plots reflect average importance across all tickers, individually. As such, we expect overlap, but a different ordering of importance since these are two different framings of the question at hand.

Fortunately, engineered features do float to the top of our feature importance plots for both specifications of the model (see Figures 3 and 4 below). The market beta feature, beta, tops the ticker-level chart, and sits fourth in our panel-level regressions. It's worth noting that very few non-engineered features appear important at all in the panel-level models, and the combined fundamentals and markets data tops the list of importances. The ticker-level importances are more of a mixed bag, and the decrease in importance from feature to feature occurs more slowly, likely because the importance of features does differ reliably across individual entities. This might also shed some light on the difficulty of modeling the panel - features that might be significantly important for specific entities may be pushed out in the panel-level, handicapping overall performance.

Based on these importance plots, and the determination of significant multicollinearity in some engineered features, we arrive at a final selection of 27 features, of which, five are raw financial markets features, two are time based, 17 are engineered from financial markets data, and three are constructed using a combination of fundamentals and financial markets data.

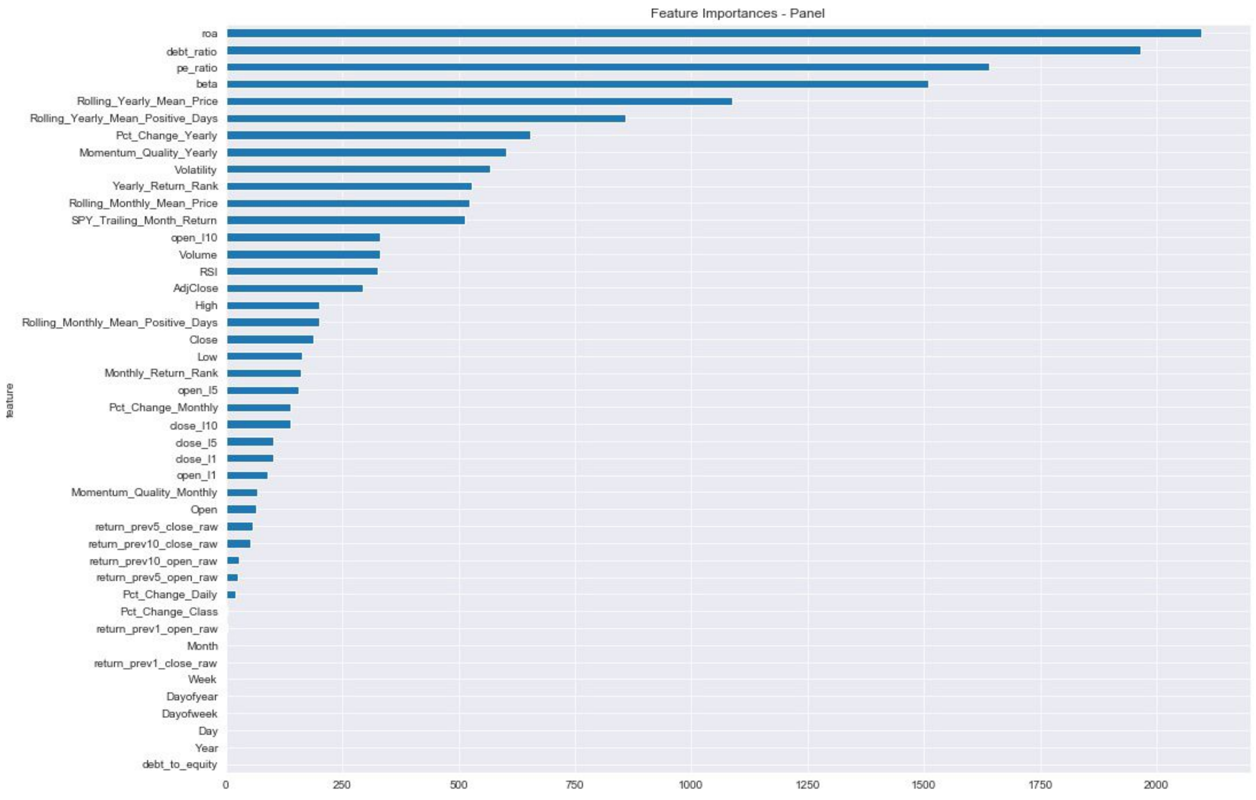


Figure 3. Feature ranking at panel level

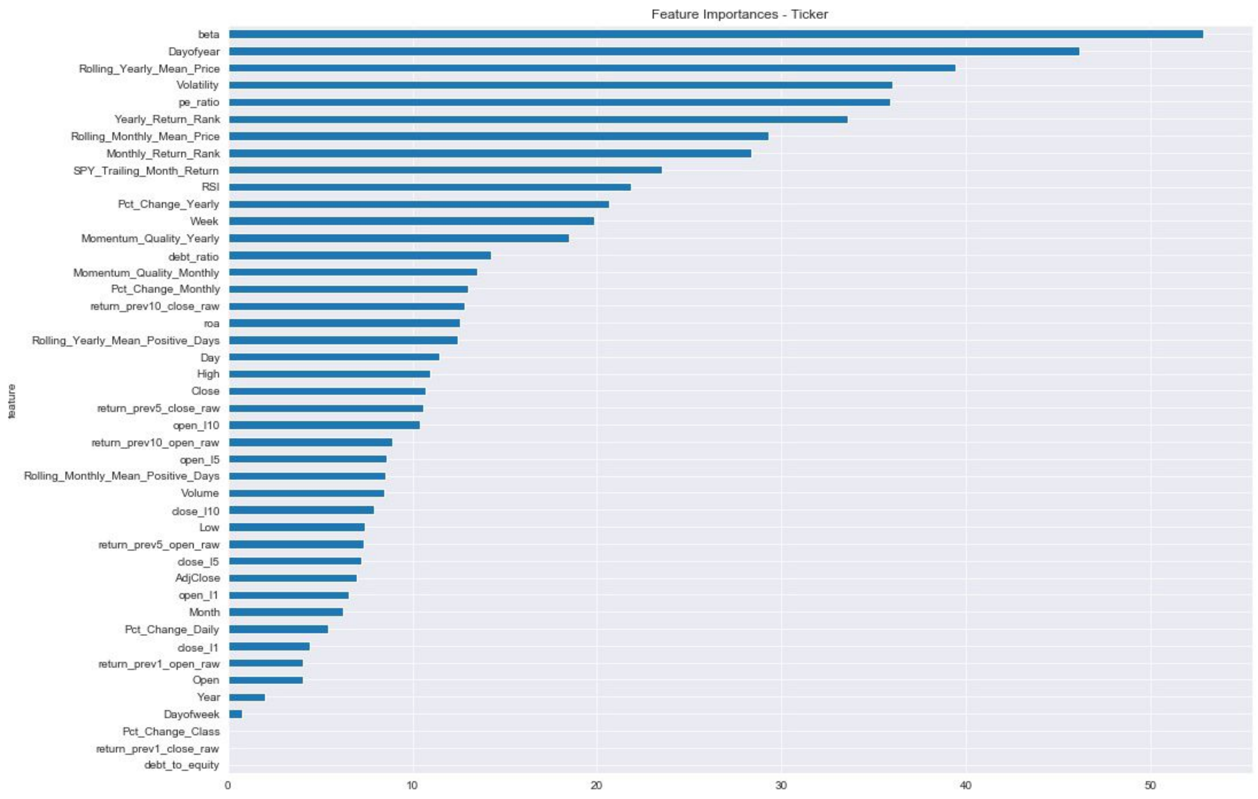
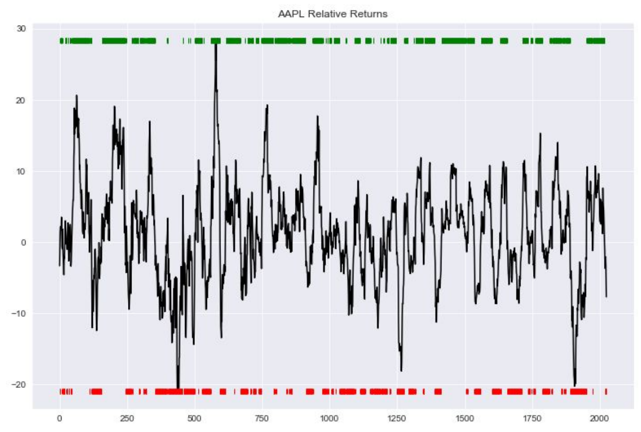
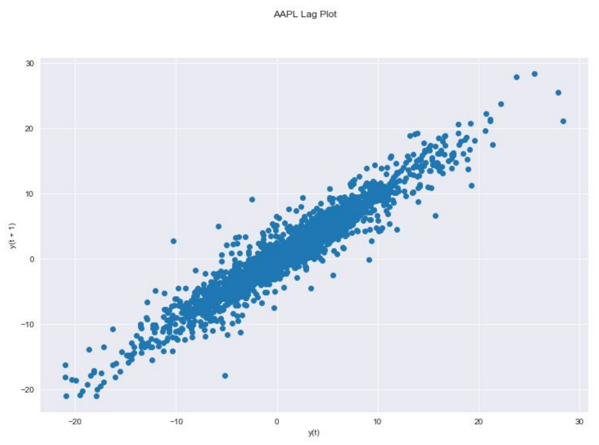


Figure 4. Feature ranking at ticker level

Since understanding our target is so critically important to the viability of our project, it is worth taking a moment to visualize it directly. As we have over 500 individual tickers in our data set following different paths, it would be burdensome to chart the path of each. For the sake of illustration, we examine the returns for Apple (ticker symbol AAPL). Below, we display the 21-day ahead relative returns for AAPL, along with green and red rugs to signify up and down days from which our binary target is derived. Further, we illustrate the lag structure of our target. Since we are not doing pure time-series analysis, this is not of paramount importance, but does help us to understand that there is serial autocorrelation present in our target (i.e. data from the past is predictive of the future). This is preferable to a series that evolves truly at random, which would be nearly impossible to model, despite our best efforts.





CROSS-VALIDATION FRAMEWORK

As alluded to in our discussion of the target we seek to predict, the time-series nature of our data necessitates very careful application of a cross-validation framework to accurately reflect out-of-sample performance as it would present in the real-world. We explore modeling frameworks in both a panel-level context, and ticker-level context. Regardless of the level of analysis, time-series splits are critical to prevent leakage of future stock-price information into the training set. Standard time-series splits prevent the use of future data in training data, so gradually expand the training set to predict a validation set containing only future data, consist with the real world prediction problem we face.

In the panel-level context, our data set consists of approximately one-million observations, (roughly 2,000 trading days worth of data for 500 stocks). Time-series splits, out-of-the-box, offer one framework for predicting the returns of the stocks on the panel, but are inherently agnostic of the ticker-level characteristics of our data set. Thus, in order to ensure each fold contains information about *all* entities, we implement a user-defined panel-splitting function that iteratively constructs time-series splits for each entity in the panel, and forces each training and validation set to contain information about all entities. This allows us to more reliably explore our research question regarding the value of interrelationships between different tickers characteristics.

In the ticker-level regression framework, standard time-series splits are more appropriate, but we explore two additional, bespoke implementations of the out-of-the-box Scikit-learn function. In both cross-validation frameworks, we restrict the size of the validation set to 21 observations, with a minimum training-set size of 252 observations (one year of trading days). Our rationale for doing so evolves from the structure of our target-since, at any given point in time, we are faced with predicting a month's worth of data out-of-sample, we seek to replicate that prediction problem within our cross-validation framework. We do not believe it is entirely realistic or informative to use validation sets of 160+ observations (what we face using twelve-fold time-series splits) since this reflects predictions that are up to 181 days out-of-sample. We implement both a windowed approach (iteratively moving a training-set of 252 observations and validation set of 21 observations through time) and recursive approach consistent with the Scikit-learn implementation that gradually grows the training set. We do not apply this framework at the panel-level because it is computationally infeasible, though the code is written such that it *can* be used, given more computing resources.

We acknowledge, moreover, that this represents something of an optimization problem in and of itself, per the bias-variance trade-off. A cross-validation framework with too many splits would be subject to a problem of high variance, while a framework with too few is subject to high bias. While we select a train-test split size that accurately reflects the prediction problem at hand, we recognize the potential this poses for overfitting. We did not have a significant amount of time to explore this optimization problem, and leave it to further research.

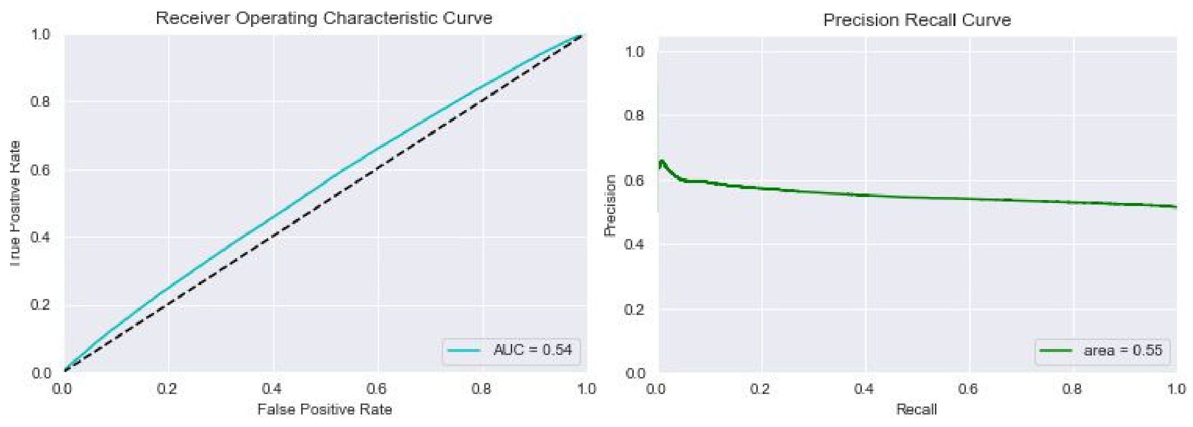
MODELING

Given the four cross-validation frameworks discussed above, along with the information we established during our feature selection/first-pass modeling process, we explored a number of workhorse classification modeling approaches. These include: logistic regression, kNN (k-nearest neighbors), SVC (support vector classifier), RandomForest, GBM (gradient boosting machines), and, finally, the LightGBM implementation of gradient-boosting decision trees. We observed reliably higher performance using ensemble modeling frameworks, and focused hyper-parameter tuning predominantly on the LightGBM implementation due to the high-degree of tuning available and speed of model fitting. Additionally, we remove 20 stocks from the initial 506 in our panel due to insufficient sample size (less than 5 years of data) and to avoid duplicitous entries.

MODEL SELECTION

On the outset, we found that all models performed very poorly on the panel-level, panel cross-validation sets. F1 and AUC scores for the logistic regression, kNN, RandomForest, GBM and LightGBM models were 0.51 and 0.47, 0.51 and 0.45, 0.52 and 0.51, 0.51 and 0.49, and 0.51 and 0.50, respectively. This would suggest that, at the panel-level using the panel cross-validation sets, our models are, colloquially, no better than a coin flip and are essentially assigning classes at random. Using the out-of-the-box time-series cross-validation splits, the same models (save for kNN due to time/compute associated with fitting) boasted F1 and AUC scores of 0.52 and 0.46, 0.51 and 0.51, 0.52 and 0.5, 0.54 and 0.49. With such poor performance across the board, the only marginally better-performing model that presents on the panel-level data set is the LightGBM. Fortunately, the LightGBM implementation of gradient-boosted decision trees provides for exceptionally fine hyper-parameter tuning, and fits models remarkably quickly, even with a data set of approximately one million observations. Moreover, given the computation constraints of fitting kNN models on our data, we will not consider them for models fit on the ticker-level data, and will focus exclusively on the ensemble modeling frameworks, with particular emphasis on the LightGBM. We did attempt to fit SVC models, but found the runtime was prohibitive (6 hours for the first 3 folds of the panel data set).

We rely on ROC/AUC, Precision-Recall curves, and standard classification report output (particularly weighted-average F1-scores) for evaluation. Note that AUC and Precision-Recall scores are calculated with respect to raw, predicted probabilities, while the "predicted" classes necessary to generate classification report have been generated using a dynamic, discrimination-threshold search which seeks to optimize the decision threshold for F1-score. Given that a passive investment strategy would inherently encourage as little adjustment as possible, we note positive-class precision scores as well, which indicate the ability of the models to avoid false-positives. Given the nature of our problem, we are less bothered by missing possible excess returns, and more concerned with avoiding mistakenly shifting concentration into an asset that will underperform based on our predictions.

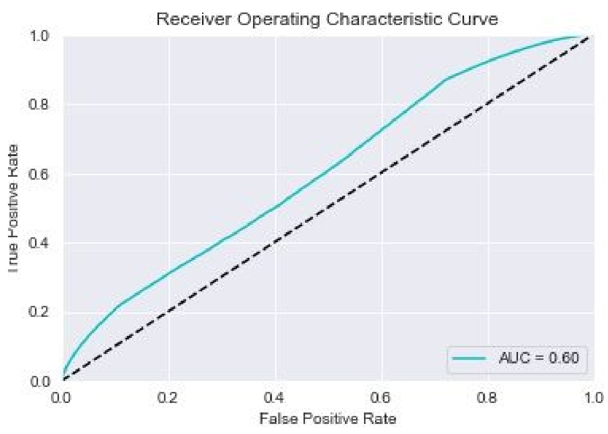


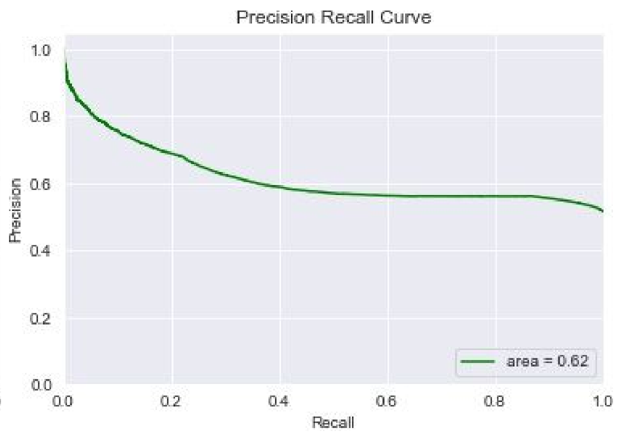
MODEL EVALUATION

We find that our best performing panel-level model is the LightGBM implementation of gradient-boosted decision trees, with max-depth of three and 10,000 estimators. The figure on the left above displays the modest amount of lift our LGBM model gets in the AUC score, calculated on 12 out-of-sample validation folds, and the figure on the right above displays the very small improvement our model provides in terms of the precision-recall trade-off. While these results are certainly nonstellar, they do represent a marginal improvement on random chance, and do so reliably, out-of-sample. Classification reports reveal this model yields a weighted-average F1-score of 0.53, and a positive-class precision score of 0.54.

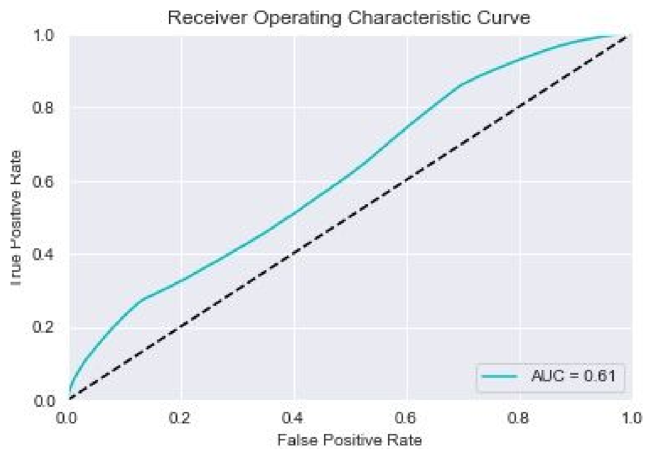
Despite these results, it is of note that the panel-level cross-validation framework yielded, for all intents and purposes, non-results, but the time-series cross-validation framework did. This suggests that the models fit on ticker-agnostic cross-validation splits performed better, so requiring every entity to be reflected in the fold did not improve performance. This somewhat counter-intuitive finding may shed some light on the generally poor performance overall-with so many individual assets evolving on their own, unique path over time, there is likely far too much noise in the data set to reliably discern the performance of individual tickers relative to the index.

This leads us naturally to the second framing of our research question - ticker-level modeling. By modeling each ticker's out-/under-performance we hope to discern more signal than in the large, noisy panel. Using the same 12-fold time-series cross-validation framework, we find similarly disappointing F1 and AUC scores for the Scikit-Learn implementations of RandomForest and Gradient-Boosting Classifier at 0.51 and 0.52, and 0.52 and 0.54, respectively. However, we note a reliable improvement in the out-of-sample AUC (and Precision-Recall scores for the LightGBM implementation, as illustrated in the figures below, lending further credibility to the use of this model, and our choice to model at the ticker-level. Moreover, these models yield weighted-average F1-scores of 0.57 (with positive-class precision and F1-scores of 0.58 and 0.60, respectively).



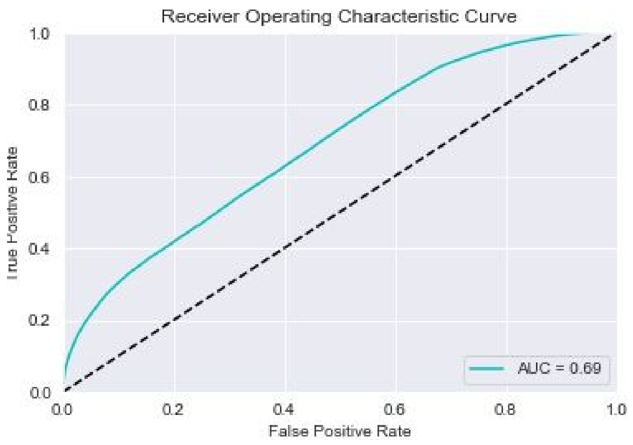


As indicated in the cross-validation section above, we also explore a more bespoke cross-validation framework that addresses the possible handicap that including very stale data in our training set may present. Using largely the same structure as the out-of-the-box TimeSeries splits, we implement a rolling window cross-validation framework (as described above) with a 252-day training window and a 126-day (or 6-month) test window, which amounts to approximately 14 splits. In doing so, we find modest *improvements* in our scores, which implies that data too far from the past is not helping, and could in fact be hurting, our performance.



By that logic, if data too far from the past is unhelpful for predicting the future, it might be realistic to believe that data too far into the *future* may be an unrealistic target. As such, we use our cross-validation framework as an additional dimension for tuning our models, in an effort to frame this in a way that more accurately represents the prediction problem we face in the real-world. At any given point in time, we are behind our target, mechanically, by 21 days. As such, there is a natural 21 day holdout set upon which we hope to predict performance relative to the index (this is the information that should inform rebalancing in an application of this model). In this vein, we considered a cross-validation framework that 1) restricts the size of the validation set to 21 trading days; and 2) restricts the size of the training set to the 252 prior trading days. This addresses two possible concerns that may be affecting the performance of the bulkier models detailed above. First, we'd expect that the most salient information for predicting returns in the near-term, would be the most recent information that transpired. Several year-old data would likely be stale, and no longer accurately reflect the state of things at the present time, and, as such, we would not expect it to be significantly informative for future decisions regarding stock price. Second, predicting a validation set of approximately 160 observations (as is the case with 12-fold time-series splits as of time of writing) represents predictions of relative performance 181 days in the future. Recognizing the inherently mercurial nature of even day-ahead returns, it is no wonder our model has difficult predicting returns more than 6-months into the future.

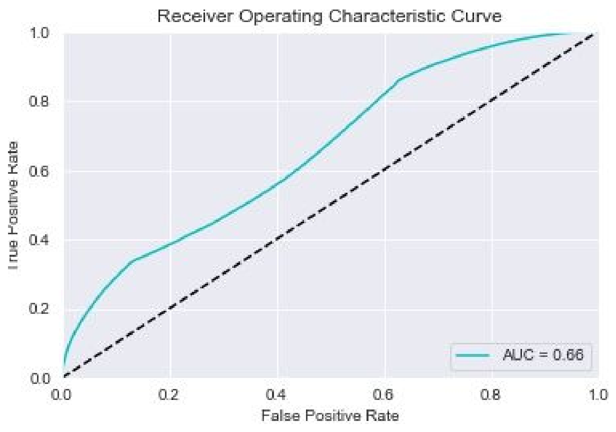
Using this cross-validation framework, along with the same, tuned LightGBM model, we find reliably better results out-of-fold. The figures below demonstrate the marked improvement in both AUC, which moves us well past the territory of random chance, and into the realm of actually detecting some signal through the noise.

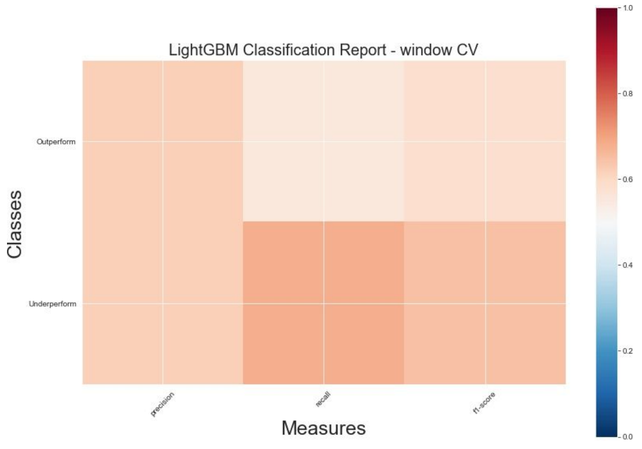


We do, however, recognize that this invites the serious possibility of overfitting - while this framing of the out-of-fold samples does more accurately represent the problem we face, it also requires the model to be fit on over 80 different splits. Averaging over so many splits does imply we may be overfitting our data purely as a result of random chance in predicting a few of those folds very well.

In order to arrive at a final model that reconciles this to some extent, we examined validation set sizes of 21, 42, 63, 84, and (as displayed above) 126 observations, each linking to real-world trading-day horizons of 1-, 2-, 3-, 4-, and 6-months, respectively. We are willing to sacrifice some performance to address the concerns associated with overfitting. Additionally, we also tuned our models further to address concerns of overfitting the *training* data, specifically. As the LightGBM model grows trees leaf-wise, rather than level-wise, it is far more accurate, but far more complex and therefore more likely to overfit the data. Noticing in-sample AUC scores of 0.95 and higher, we tuned our final model to handicap performance in-sample, in hopes of making the model more generalizable with data out-of-fold.

Using a final model with: 1) 10,000 estimators; 2) a learning rate of 0.1; 3) max depth of 2; and 4) maximum bins and leaves of 25 each, trained on a rolling window cross-validation framework with 252 days of training data and 63 days of test data, we find the following, reasonable results. We find an AUC score of 0.66, and area under the Precision-Recall curve of 0.67. As the classification report heat map suggests, weighted-average F1-scores have improved somewhat. We find weighted-average F1-scores of 0.62, along with 0.62 precision and 0.67 F1-score for the positive class. Moreover, we note that there is significant disparity in model-performance both between different entities (suggesting that some tickers are more predictable than others in the context of our data set) and between validation folds (suggesting there are certain months that are more predictable than others). These novelties are a natural starting point for further research.





In addition, we applied a custom built discrimination threshold search function to optimize the discrimination threshold for a grid of possible values between 0.25 and 0.75, and relative to a specified metric. We found this modestly improved our results, at least at the ticker level, and report the classification report using that search above.

APPLICATION

Given the non-trivial performance of our ticker-level models, we identify a unique opportunity to deliver a data product that allows users to select individual tickers within our basket to examine overall predictability, the path of relative returns historically, and predicted probabilities of performance in the 21-day holdout period that naturally exists using our target. As such, with data that is ingested on a daily frequency, on any given day, one would be able to assess the predicted performance of tickers in their portfolio, and adjust accordingly.

As a part of our next-steps, we hope to render a proof-of-concept of this application in a Jupyter notebook using the Plotly and ipywidgets libraries, to provide user-interaction for selection specific stocks to assess the trajectory of their performance historically, the classes associated with the raw returns that we seek to predict, indicators of overall predictability (based on evaluation criterion established above), and predicted probabilities of out performance in the next month, relative to the day the application is visited. Rather than store the results of every model in memory to index, we defer to a framework that models the ticker performance in real-time and provides immediate results. In a more robust implementation, one would likely store ticker-level data in a database to be accessed by the user directly, and in doing so could provide to users rankings of stocks by both predictability and predicted outperformance in the coming month.

Naturally, we would hope to map the predicted probabilities of returns to their actual returns in the real-world, and backtest to determine if knowing this leading information is helpful for rebalancing a passive portfolio.

ASSESSMENT

The problem we sought out to address is, admittedly, an extremely difficult one. Predicting stock price movements is regularly disregarded as a fool's errand, particularly without untold amounts of domain expertise and resources. Still, the models we developed, particularly at the ticker-level, indicate \textit{some} predictive power beyond random chance that could be exploited as a leading indicator for optimal rebalancing of a passive portfolio. Gratuitous domain expertise and resources notwithstanding, we address a few key limitations of our approach, and provide tangible next-steps to improve upon the analysis herein.

LIMITATIONS

If the past few months are any indication, it is clear that the fluctuations in the stock market are driven just as much by emotion and sentiment as any underlying fundamental financial principles or models. Despite this, given that so much of today's market participants are underscoring their decisions with rigorous quantitative methodology/AI, we sought to divine, to the extent possible, some signal through the noise and arm investors with an additional leading indicator for stock performance in a monthly window. We believe our models *have* latched onto some signal, but cannot possibly capture the emotions/irrational exuberance narrative.

So much of what has driven movements in stock prices within the past six months has been emotion, fundamental shifts in foreign policy and developments in other advanced economies (i.e. Brexit, trade war, etc.), Federal Reserve communications, and anything that comes out of the president's twitter. Incorporating broad-based measures of geopolitical, fiscal, and monetary developments would likely improve our models, but would significantly expand the scope of our research question, and move this exercise more within the realm of a Natural-Language Processing (NLP) exercise to gauge sentiment in the economy broadly. Building a model purely based on characteristics of the underlying financial data, abstracts away from these drivers that are far more difficult to capture, and therefore handicaps the ability of our models to accurately predict some of these fluctuations.

Beyond the limitations of our data set, we did not explore the advances in machine- and deep-learning that have led to more powerful models purpose-built for time-series prediction. Recurrent neural networks (RNNs), particularly long short-term memory models, have manifested as the gold-standard for time-series prediction in the field today. In order to keep our pipeline consistent with the Scikit-learn API, and to leverage the skills we developed through the certificate program, we chose not to leverage these kinds of models, but could reasonably expect model performance to improve using these more advanced approaches.

NEXT STEPS

The limitations described above lead naturally to some tangible next steps to explore with our analysis, which we attend to here in addition to some less abstract possibilities related to our product. Incorporating sentiment surrounding the market itself, and capturing the more intangible geopolitical, monetary, and fiscal considerations that inevitably drive movements in stock prices, would likely go a long way toward improving the results of the model. The largest shifts in the market in the past few months have largely been developments related to trade tensions between the U.S. and China, along with signalling from the Federal Reserve Board of Governors regarding the stance of monetary policy. The trade considerations, in particular, would be extremely difficult to capture with employing sentiment analysis and natural-language processing, though including them simply as an "event" dummy-variable could make a difference-doing so would take a considerable amount of time as an individual would likely have to hand-pick what "events" are significant enough to reflect in a feature.

Incorporating more robust, purpose-built models for time-series analysis would not go amiss, either. We abstracted away from these types of models (to the extent possible), largely since they were a departure from the curriculum of the program. Given more time, we would hope to put our data to the test using an LSTM model, or using more advanced time-series regression techniques such as symbolic regression. Even with our current suite of models, we were somewhat limited in our ability to tune hyper-parameters given how computationally intensive some of our models were - panel-level data sets were particularly difficult to tune since models could take hours to run, and doing a great deal of tuning at the ticker-level felt like it would lead to overfitting on single tickers. In most cases, defaults, or collections of hyper-parameters that reduced overfitting performed best. However, given more time and, possibly, a GPU, we would spend more time tuning our models

We did not have time to develop a working demo of a product/application, and can envision some ways to do so. The next step would be to have this pipeline running automatically, each day, to populated data in an app that a user could access to get information about best- and worst-performing stocks per our prediction framework. Additionally, using the more involved windowed cross-validation approach offers an opportunity to compare the characteristics of different time periods with those moving forward. Since the models perform quite well in some folds, and poorly in others, an additional layer of complexity that weights the quality of predictions by the similarity of a given time-period to the present could prevent rebalancing in error.

Lastly, we did not have enough time to robustly map our predictions to actual returns, and compare these returns relative to the market in absolute terms. Given more time, we would do so, and could enter into an additional optimization problem wherein machine-learning is used to determine how much concentration in stocks predicted to underperform should be moved into those predicted to outperform.

CONCLUSION

Despite the significant difficulty our research questions presented, we are pleased to have divined some signal in the overwhelming noisiness of stock returns. Revisiting our high-level hypothesis, (derived features from financial and market data will be predictive of relative returns for individual stocks in a panel-level data set of S&P 500 stocks), we have found that the panel-level approach was not particularly helpful for predicting returns and, in fact, entity-agnostic, pure time-series cross-validation frameworks yielded better out-of-sample performance in the panel.

We were happy to find that the significant time invested in feature-engineering, and informing the cultivation of a set of features with the existing literature, was worthwhile as engineered features topped our feature importance charts. Additionally, the careful thought put into developing cross-validation frameworks that more accurately reflected the prediction problem as it presented outside of this simulation bore the most reliable, interesting results. The clearest next steps for the project are further development of an application (outside the scope of a Jupyter notebook) that would be consumer-facing to provide real-time indications of signals of predicted under- and over-performance, and mapping of these signals to real returns, and providing automated rebalancing of a passive portfolio.

REFERENCES

1. Documentation of scikit-learn 0.21.2 (n.d.). Scikit-Learn. Retrieved June 12, 2019, from http://scikitlearn.org/stable/documentation.html
2. Fuller, W. A. (1976). Introduction to Statistical Time Series. New York: John Wiley and Sons. ISBN 0-471-28715-6.
3. Gray, Wesley R., and Jack R. Vogel. *Quantitative Momentum: A Practitioners Guide to Building a Momentum-Based Stock Selection System*. Wiley, 2016.
4. Gray, Wes, and Jack Vogel. “Frog in the Pan: Identifying the Highest Quality

Momentum Stocks." *Alpha Architect*, 18 Aug. 2017, alphaarchitect.com/2015/ 11/23/frog-in-the-pan-identifying-the-highest-quality-momentum-stocks/

5. Guolin Ke, Qi Meng, Thomas Finley, TaifengWang,Wei Chen,Weidong Ma, Qiwei Ye, Tie-Yan Liu. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 3149-3157.
6. LightGBM docs (n.d.). Microsoft. Retrieved June 12, 2019, from https://lightgbm.readthedocs.io/en/latest/
7. Various discussion/forum posts from "Two Sigma: Using News to Predict Stock Movements." Kaggle. Retrieved June 12, 2019, from https://www.kaggle.com/c/two-sigma-financial-news/overview
8. Seltman, H.J., *Experimental Design and Analysis*, Carnegie Mellon University, 11 Jul. 2018, http://www.stat.cmu.edu/~hseltman/309/Book/Book.pdf
9. SimFin API Tutorial, SimFin, 6 Jan. 2019, Retrieved March 2019, from https://medium.com/@SimFin_official/simfin-api-tutorial-6626c6c1dbeb
10. Zhi Da, Umit Gurun, and Mith Awarachka, "Frog in the Pan: Continuous Information and Momentum", 2014

ACKNOWLEDGEMENTS

We thank Karen Belita for her helpful comments and guidance throughout this process, and for helping us frame our final product in light of initially non-stellar results; we also thank Franco Bueno Mattera for his help in developing our code base.

AUTHOR CONTRIBUTIONS

J.B. developed the ingestion and wrangling pipeline for fundamentals data from SimFin, engineered SimFin related features, developed all modeling, target generation, cross-validation, and discrimination threshold search code (and accompanying presentation materials), performed model selection and tuning, and contributed significantly to the development of the final product and model evaluations portion of the project; G.K. developed all feature engineering code, developed the pipeline to create momentum-based features, contributed to early-stage modeling, and contributed to the development of the final product and model evaluation portions of the project; M.M. contributed significantly to exploratory data analysis and feature selection; S.Y. developed pipeline for ingestion of markets data from Yahoo! Finance, hosted and maintained the central Postgres database, and contributed to exploratory data analysis, feature selection, and modeling. All authors contributed to written progress reports, final presentation slides, and final paper.

GITHUB

Our GitHub repository is located at:
https://github.com/georgetown-analytics/Passive-Stock-Fund-Optimization