Data Science and Retirement Planning

Joy Torres is a 28 year old creative and yoga instructor that loves her work, but is concerned about her financial health. Since her small yoga studio doesn't offer a 401K, she wonders how best she can prepare for retirement. She follows bloggers in the FIRE movement (Financial Independence, Retire Early) and purchases best selling investment books, like Tony Robbins' Money: Mastering the Game. Of all of the information she has read, the "All Weather Portfolio" seems the most in line with her need for growth and tolerance for risk, but she still feels uncertain.

In so many of these publications, which are made for a non-technical audience, there is an oft-quoted refrain of "we have crunched the numbers so you don't have to." And while she understands the promise of financial security and being able to focus solely on the work that she loves, she still hesitates to begin investing.

The purpose of this project, **Data Science and Retirement Planning**, is to build a code base that allows an investor to test hypotheses that they've read about online or in investment media, to demystify the results, and to give them a greater sense of confidence so they can begin planning for their future with open eyes.

In this first version of the project, the objectives are as follows:

1) Develop functions to construct diversified portfolios with multiple categories of funds.

2) Develop functions to evaluate the performance of the portfolios, including regular contributions of a fixed amount and rebalancing the proportions of funds across categories.

3) Create a simulation experiment to evaluate the claim that an All Weather Portfolio will return on average, 9.72% each year, and lose only 3.93% during its worst year.

4) Recommend funds for an All Weather Portfolio based on the results of the simulation experiment, with special care taken to avoid losses during the recession.

5) Evaluate several ML tools can for predicting a loss of 2% or more in a stock index fund in the upcoming week using averages of various time-lengths from the other components of the All Weather Portfolio.

Note that this version exists as a first step towards a product that is useful for an end-user. This version does not account for fees, tax considerations, the re-investing of dividends paid by funds in the portfolio, or what to do in the event of a correctly identified loss of 2% or more. Please also note that past performance is not an indicator of future performance.

EDA + Simulation Data Preprocessing

A Jupyter notebook will be used to assemble pandas dataframes for each of the categories of investment identified in the All Weather Portfolio. Building an All Weather Portfolio involves the following categories and proportions of funds.

30% in stock index funds
40% in intermediate bond funds
15% in long term bond funds
7.5% in gold
7.5% in broad basket commodities

Specific funds within each category have been chosen to ensure that their history goes back until at least mid 2007, so that the financial crash of 2008 is present in the data. The categories and validated funds are as follows.

- Stocks: FUSEX***** (FXAIX), PREIX, SWPPX, VIGRX, VFINX
- Intermediate Term Bonds: BIV, HYG, IEF, IEI, IGIB, ITE, TIP
- Long Term Bonds: PRULX, WHOSX, VUSTX
- Gold: INIVX, OPGSX, SGGDX, USERX, VGPMX
- Broad Basket Commodities: DBC, DJP, GSG, GSP

Relatively clean data sets are found as downloadable csv files at https://finance.yahoo.com/ by searching the ticker codes listed above, clicking on the historical data tab, and downloading. Unfortunately, Yahoo Finance discontinued its public API in 2017 and a free alternative API was not found at the time of this version. Future versions should utilize public API data.

***FUSEX was absorbed by FXAIX in late 2018.**

The CSV_Formatting function:

Each csv will be processed with a custom-written ‘csv_formatter’ function. It will load the csv using the pd.read_csv function. And from the headers ‘Date’, ‘Open’, ‘High’, ‘Low’, ‘Close’, ‘Adj. Close’ and ‘Volume’, it will select the ‘Date’ column as an index for a new dataframe with a single column for the closing price, ‘Close’. The column label for the closing prices will be set as the ticker code from the csv title. To generate percent changes for each day, a new column called ‘Next Day Values’ will be made by taking the closing price values and shifting them one day previously. A final column, ‘Percentages’ will be created from the calculation dividing the next day closing value by the current value. A dataframe consisting of the date index and the percentages is returned by the csv_formatting function.

Now that the data has been treated, the individual fund dataframes within each category can be inner joined by date. This results in usable dataframes for portfolio creation and analysis. These category data frames are then passed to the weekly master function for further treatment.

The Weekly_Mastering function:

Now that the data has been downloaded and represented as incremental changes, it will be helpful to think about the most appropriate representation of the data for future use. For every portfolio simulation, an amount in a particular fund will need to be multiplied down every percent increment to obtain its value throughout the life of the investment. Several problems arise:

1) Weekends, holidays, and the like will not be represented
2) NaN entries may provide difficulties
3) Resampling and simulation may prove time-intensive with this many calculations

To address problems 1 and 2, the date index will need to be transformed into Datetime objects and (for easy reading) time is not included in the format. Then, a second dataframe with a single index column will be created containing all dates within the range of the finance dataframe. The two will be 'outer joined', so that now all days missing from the original are listed with a NaN value. Lastly, .fillna(1) will be applied to turn all NaN values into 1's. Thus, any value previously obtained by multiplying the previous day's percentage change will not be affected by passing through the NaN entry (now a multiplication by 1). Now the dataframe is working for every day of the calendar year, indexed properly by date.

To address problem 3, the updated, working dataframe will be resolved into weekly summary percentages. This will reduce the number of calculations by a factor of 7, and will also provide a platform that lends itself well to future investment experiments (like how much should I contribute every 2 weeks to build wealth?). There are several steps in the algorithm to create this dataframe.

The number of weeks is calculated by dividing the number of rows in the working dataframe (every day in the calendar year) by 7. The number of weeks is used to set the range for a for loop. Weekly portfolio segments are then built by using .iloc[i*7:(i+1)*7] to pull 7 row 'chunks' of the dataframe at a time. Within each 'chunk', .iloc[:,] is used in conjunction with .product() to obtain the product for each column/fund's daily percentages, making a weekly summary of change. The weekly values for each fund in the category dataframe are then made into a column of a dataframe with corresponding dates from each chunk as the column headers.

This dataframe is then Transposed to place the datetimes back along the row labels and the fund names are restored as column labels. The weekly index is also reset as a datetime index. The weekly mastering is complete and the data frame of weekly incremental changes is returned.

Finally, functionality has been added to orient every df to the end at the nearest Saturday. This ensures that the weeks are labeled consistently on the same datetime indices and merges between the data frames will be successful.

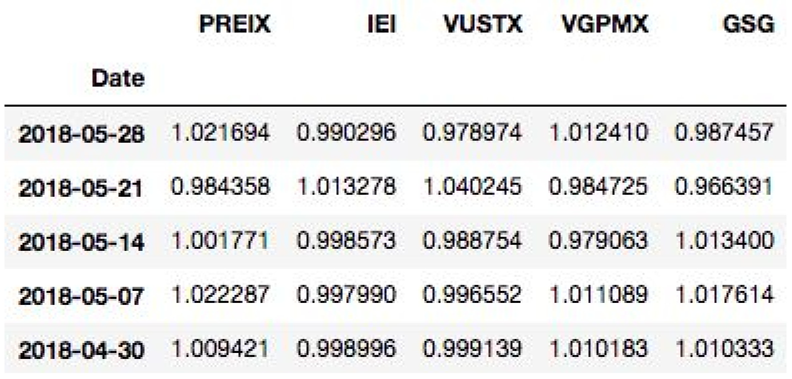
<u>Random Portfolio Generation:</u>
The final step towards preparing the data for evaluation, is the creation of random portfolios. A random portfolio generating function is created by doing the following.

1) For each category, use .sample(axis=1) to select a random column label

2) .loc the column of the weekly master dataframe using that label

3) Merge all sampled weekly data frames by date

4) Return the resulting random portfolio, formatted weekly and ready for analysis!

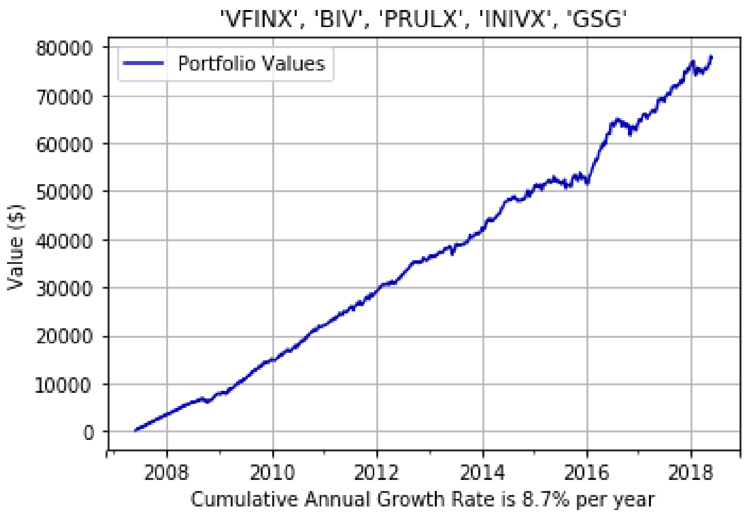


	PREIX	IEI	VUSTX	VGPMX	GSG
Date					
2018-05-28	1.021694	0.990296	0.978974	1.012410	0.987457
2018-05-21	0.984358	1.013278	1.040245	0.984725	0.966391
2018-05-14	1.001771	0.998573	0.988754	0.979063	1.013400
2018-05-07	1.022287	0.997990	0.996552	1.011089	1.017614
2018-04-30	1.009421	0.998996	0.999139	1.010183	1.010333

<u>EDA, Simulation, and Statistics</u>

Functions for the Cumulative Annual Growth Rate (CAGR) and Time Weighted Return were developed for the portfolio dataframes like the one pictured above. When evaluated, individual portfolios can be evaluated as line graphs or with summary statistics.

Because of the nature of the timeline chosen in this project, there are is not much value in the portfolio at the time of the market crash. To compensate for this imbalance, another "alternate timeline" was manufactured by reversing the order of the weekly data, placing the 2008 downturn at the end of the timeline. CAGR and time weighted returns for the purpose of evaluating All Weather Portfolios will be made by averaging the results from both timelines.



With these basic evaluation tools, the simulation experiment can be run.
To simulate a portfolio

1) Select one random fund from each category to form the portfolio.
2) Accept an initial investment amount and a recurring bi-weekly addition.
3) Accept a user specified proportion for the classes of the portfolio.
4) Evaluate the performance of the portfolio across historic data.
5) Rebalance the portfolio at a regular, user specified interval.
5) Output Compound Annual Growth Rate (CAGR)
6) Output the worst loss across 52 consecutive weeks.

To perform the data exploration

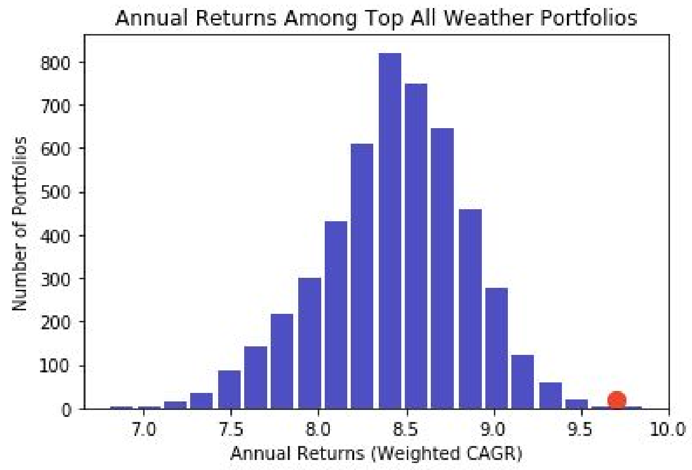
1) Create distributions of portfolios with various input parameters
2) Create distributions of portfolios which change under a variety of rules.
3) Develop hypotheses for improving on the All Weather Portfolio
4) Test the hypotheses against historical data

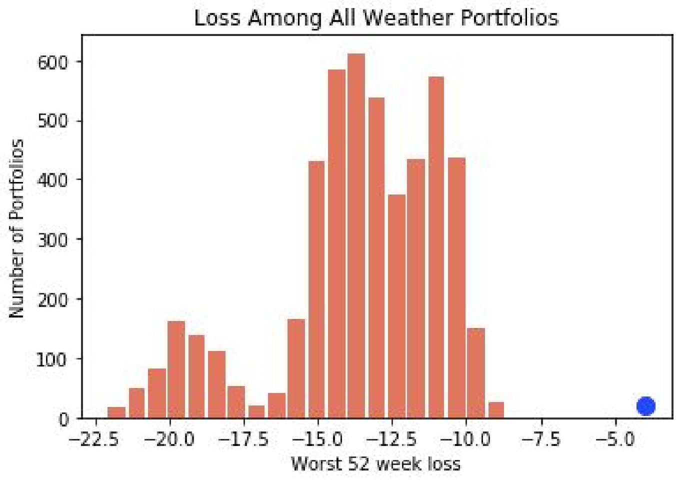
Statistical Methods

1) Simulated distribution will be tested for a mean of 9.7% CAGR
2) Simulated distribution of losses will be tested for a max loss of 4.3%

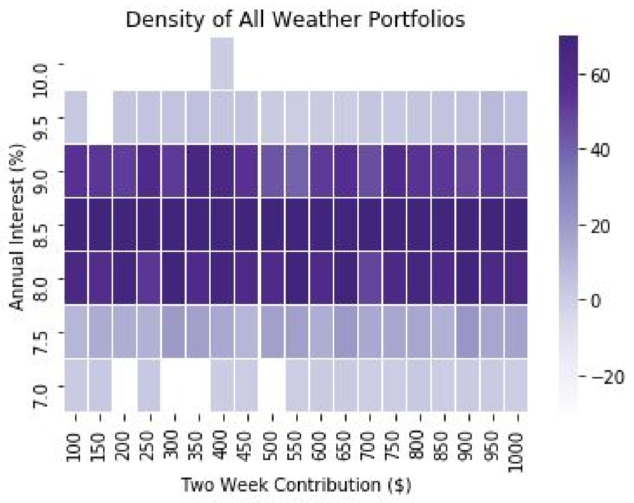
After simulating 5000 All Weather Portfolios only 4% of random portfolios had an expected CAGR over 9.7 percent and 100% of the worst years calculated had losses exceeding 4.3 percent! The analysis implies that it is extremely unlikely the expected (mean) CAGR and loss values are 9.7 and 4.3 percent, respectively. Based on the analysis, it does not seem likely that this kind of portfolio performs as well as indicated in the book <u>Money: Mastering the Game</u>.

However, additional insights can still be gained from studying the resulting dataset.





A bootstrap hypothesis test determined that there is no significant difference (p = .20) between those portfolios where the investor contributed $100 a pay period and those where the contribution was $1000. As seen in the graph, each vertical "slice" of return values is comparable to the next, regardless of how much is paid into the portfolio every two weeks.



Further examination of the results provide an idea of which funds and combinations of funds provide the best returns. Frequencies were examined for which funds appear in those portfolios with CAGR above 9 and which appear in those portfolios which experienced losses in the smallest quartile of all losses. As a result, the following optimally performing portfolio was recommended:

VIGRX, IEF, WHOSX, SGGDX, ANY COMMODITY FUND

The recommended portfolio experienced a CAGR of 9.99%, with a worst year's loss of - 9.33%. This is still impressive, considering the normal losses experienced by the other All Weather portfolios - 10% to - 21%. How well does this recommended portfolio perform? The S&P's 500's 10 year average through 2018 was 8.46%, and it's biggest loss in 2008 was 36.55%.

The portfolio most recommended by the data analysis delivers on the promise of keeping pace with the market and minimizing losses, albeit not as spectacularly as originally promised. It is a well balanced investment plan for an amateur investor.

<u>Machine Learning and the All Weather Data</u>

If the hypothesis of the All Weather Portfolio is correct, there are relationships between these different investment sectors which compensate for dips in the stock market and shield the portfolio from major losses. It is possible that these relationships are reactive to the lowering of index value, but this project will apply machine learning to determine if the relationships between these sectors might be used to predict upcoming losses in the stock market.

Methodology

The data to be trained and predicted will be the anticipated weekly change in VIGRX fund value. Previous work has been published predicting increases or decreases in a single company's stock, but without a significant amount of precision. In keeping with the philosophy of the All Weather Portfolio, this Machine Learning project will only focus on losses which might be seen as significant (2% or more in one week). Such losses only account for about 14% of the data, but will likely be easier to predict the numerous smaller losses which might more often than not just be interpreted as noise for most classifiers. When the models are trained, all weeks with losses equal to or worse than 2% and all weeks which do not experience those incremental changes will be flagged with 0 and 1 respectively. Additionally, Naive Resampling, SMOTE and ADASYN methods will be performed to develop training sets with additional observations appropriate for a 2% loss. These methods for expanding the data to interpolate alternate data points consistent with a 2% loss will be necessary, as ML Classifiers are designed to classify the majority of the data correctly as they are tuned.

The features will include average changes in each of the aforementioned sectors, using funds from the sample space identified in the project which precedes this one: Data Science and Retirement Planning, Resampling the All Weather Portfolio. Additional features included trailing averages/values for both the VIGRX value, and trailing values from each of the identified sectors. Durations of 2, 3, 6, 9, 12, and 15 weeks were arbitrarily chosen to create the trailing values.

The code for this process can be found in the "ML Preprocessing" Jupyter notebook, but an outline of the underlying algorithm is as follows.

1. Accept the weekly summaries as dataframes for each fund from the previous analyses.

2. Create a timed_eval function which accepts a label, dataframe, and integer, then

    a. Selects a number of rows from the domain dataframe, offset by 1 week to avoid using the prediction week's values in the training set.

    b. Computes the product of all values in each row/week (since the data frame it accepted contains values of 1 + % change) to measure growth over that period.

    c. Attaches the resulting products, (the trailing values), to the corresponding date-time index value that the measurement precedes. Return the resulting data frame with the desired label

The dataframes passed to this function include the VIGRX fund itself, as well as dataframes formed by averaging the fund behavior across the other categories of the All Weather Portfolio. For instance, in the long-term bond funds, PRULY, VUSTX, and WHOSX, the incremental changes for all three funds are averaged for each week to create a broad summary of that part of the market to be used as a feature. Then, the trailing measurements for that summary are used as additional features.

The function is then used for the aforementioned numbers of weeks to generate a variety of trailing values for each week in the fund and summary histories (oriented to the same date-time indices), and the results are then merged into a single dataframe to serve as the feature set.

Lastly, the VIGRX index values are converted from percentages to binary, where 0 represents weeks that saw a loss of 2% or more, (.98 or lower in the dataframe values), or 1 otherwise. This resulting array is used for validation.

The following classifiers were run “out of the box” in the “ML Initial” notebook and also had their hyperparameters tuned for optimal F1 scores:

Support Vector Machine (C, Gamma)
Logistic Regression (C)
Gradient Boosting (min_samples_split, min_samples_leaf, max_depth, etc.)
Decision Tree (min_samples_split, min_samples_leaf, max_depth, etc.)
Random Forest (n_estimators, max_depth, etc.)

The F1 score was chosen since it accounts for both precision and recall, enabling it to mitigate both types of classification error. Practically speaking, an investor who may decide to move money from a stock index will be concerned that they do not move the funds out when the value is going up, or leave them in when the value is actually experiencing a significant loss.

To provide an additional stress test of the machine learning algorithms, an additional hold-out set of data (from June 2018 to Feb 2019) was then evaluated using the classifiers. 9 of those 34 weeks of data experienced significant losses.
Note that in the following table, F1 scores are presented in the format a/b, where a is the F1 score for classifying a significant loss and b is the alternative case.

Classifier	**Imbalanced-Learn Adjustment**	**F Stats on Train/Test/Split July 2007 - May 2018**	**F Stats on June 2018 - Feb 2019**
SVC	none	.00/.93	.00/.85
LogReg Classifier	none	.00/.93	.00/.85
Gradient Boost	none	.17/.93	.00/.85
Decision Tree	none	.23/.91	.44/.80 untuned 0.33/0.86 tuned
Random Forest	none	.21/.92	.20/.86 untuned .00/.85 tuned
SVC	SMOTE	.46/.68 untuned .91/.92 tuned	.36/.70 untuned .00/.85 tuned
LogReg Classifier	SMOTE	.59/.64 untuned .64/.63 tuned	.42/.68 untuned .11/.68 tuned
Gradient Boost	SMOTE	.84/.84 untuned .82/.83 tuned	.29/.68 untuned .32/.73 tuned
Decision Tree	SMOTE	.81/.79 untuned .82/.81 tuned	.21/.69 untuned .13/.75 tuned
Random Forest	SMOTE	.84/.83 untuned .65/.64 tuned	.42/.78 untuned .43/.71 tuned
SVC	ADASYN	.33/.67 untuned .93/.94 tuned	.42/.68 untuned .00/.85 tuned
LogReg Classifier	ADASYN	.63/.66 untuned .67/.67 tuned	.40/.65 untuned .18/.65 tuned
Gradient Boost	ADASYN	.86/.83 untuned .84/.83 tuned	.30/.71 untuned .20/.67 tuned
Decision Tree	ADASYN	.83/.79 untuned .75/.75 tuned	.10/.63 untuned .32/.73 tuned
Random Forest	ADASYN	.88/.86 untuned .63/.66 tuned	.35/.78 untuned .48/.70 tuned

Note that all five classifiers were also run on a sample set subject to Naive Resampling, and while the train-test-split results were consistently in the low 80’s to low 90’s, they unanimously failed the recent data exercise with F1 scores of .00/.85

Evaluation of ML Classifier

Given a prediction from one of the better performing classifiers, what can be done to benefit the investor? In the final phase of this version of the project, a preliminary exploration of the answer will yield a set of new questions that must be answered by further exploration, simulation, and machine learning.

The naive approach taken in the notebook “ML Applied Analysis” is as follows:

1) Refresh those algorithms which performed the best in training/testing

2) Apply those algorithms to the data at hand, creating a new column in the evaluation dataframe which contains 0’s where losses are expected in the upcoming week and 1’s otherwise.

3) Implement an intervention during those weeks where stock index fund losses are expected. In the above mentioned file, a proportion of the stock index fund was distributed to the other funds in the portfolio evenly.

4) Create an additional rebalancing metric that will occur on those weeks that are not expecting a loss (assuming stock index funds no longer account for at least 30% of the portfolio)

5) Conduct a grid search of proportion values that specify the dosage of the intervention.

6) Build alternate Classifier models which allow the better performing models to “vote” on the expected outcome, and study the effects of different vote thresholds for classification.

7) Evaluate results.

Unfortunately, in each portion of the exploration above the naive redistribution of funds during flagged weeks resulted in decreased CAGR (and steeper decrease as the intervention dosage increased!). These results suggest that either the naive redistribution method is faulty, or the classifiers are not accurate enough to be effective.

It will be an objective of version 2 of this project to address this challenge and provide both an effective intervention to mitigate losses due to market downturns and an optimal dosage for that intervention. Other economic data may also be imported as features to complement the existing data and improve the ML results. Additional explorations can be made using the existing code to determine which funds are likely to **not** lose money at the same time the stock index is flagged for losses. This would allow for more sophisticated interventions and greater improvement in the bottom line. If the tools developed can bring the expected CAGR to 11 - 13%, the final phase of version 2 will be to make a real-time productionized version which can run analyses and make weekly recommendations for users trying to maximize performance in their All Weather Portfolios.

<u>Conclusion</u>

For independent investors like Joy Torres, taking the first steps towards building wealth and security should be much easier. She can take advantage of the growth of the stock market without exposing herself to huge risks. While the 3.93 maximum loss was not evident in the data we observed, it is still possible to mitigate significant losses following this project's recommendations. So why should she feel more confident in beginning ensuring her financial health for years to come?

- **We chose a strategy (the All Weather Portfolio) which aligned with her values.**

- **We experimented on these portfolios to find the best performing versions, and avoided bad combinations of funds (which could lose over 20% in one year, despite the diversification!).**

- **We set up reasonable expectations for the recommended portfolio.**

- **We built functions and tools which can be run again and again, and can offer more insights and recommendations in the future.**

The Machine Learning tools developed in this iteration of the project provided interesting results, but additional analysis and work must be done before they can effectively improve the performance of the All Weather Portfolio.