SYS 6014 PROJECT - RYAN KRECHEL - FEB 18, 2020

INTRODUCTION

For this project, I intend to create a model for seasonal investing to find a way to beat a buy-and-hold investment strategy. In theory, there are certain market trends pertaining to certain days of the year. For instance, the beginning of December tends to be a time when investors sell off stocks to cover the cost of taxes. This mass tendency to sell generally causes a decrease in the stock market. If an investor is aware of this trend, he or she should choose to invest in less aggressive accounts (perhaps bonds). My goal is to find other trends based on historical data and find a strategy for trading on a day to day basis within a calendar year to maximize returns.

Success is defined as beating buy-and-hold. If the strategy output by the decision making tool has a return greater than any of the individual funds alone, then a better investing strategy has been found.

I have chosen seven Vanguard funds to analyze. Each represents a different level of risk and different sector of the market.

1. VTI - Large Cap Blend, Stock Market ETF
2. VXF - Mid Cap Blend, Extended Market ETF
3. VBK - Small Cap, Small-Cap Growth ETF
4. VEA - International, FTSE Developed Markets ETF
5. VWO - Emerging Markets, FTSE Emerging Markets ETF
6. VPU - Utilities, Utilities ETF
7. BLV - Bond, Long-Term Investment Bond

Data will be collected from 2008 to 2019. This is a common timeframe that all of these ETFs share. This will allow the performance of each ETF to be compared to each other fairly.

ACTION SET

The decision-maker's action set A : A complete itemization of all the options available to the decision-maker, from which the decision-maker selects a unique choice.

There are 8 decisions in this model:

1. Invest in VTI
2. Invest in VXF
3. Invest in VBK
4. Invest in VEA
5. Invest in VWO
6. Invest in VPU
7. Invest in BLV
8. Do Not Invest

STATE SPACE

The state space or sample space X : The space from which observation data are drawn.

Data will be collected using Yahoo's share price history tool from 2008-2019. Preprocessing will be done on the "open" and "close" prices to convert the share prices into daily losses or gains. This will be produced and uploaded into Matlab code with the corresponding dates to use in the analysis.

DATA GENERATING PROCESS

Some description of the data generating process. This might take the form of a statistical model describing the statistical properties of random variables X1,X2,…,XN∈X. For example, you might posit that each observed data points x1,…,xN∈X are the realization of a independent, identically distributed (i.i.d.) normal random variables X1,…,XN, each with mean θ and variance σ2 : Xn N(θ,σ2). Here, θ and σ2 are unobserved parameters: we don't observe them directly, but the sampling process provides us with data that is useful to refine our beliefs about their values.

Once the raw data is uploaded into Matlab, it will need to be sorted by date and by gain/loss. The code will be written to assess January 2nd as a specific decision node with historical data from all other January 2nd data points. Within this day's worth of data, there will be a set of positive gains and a set of negative losses. In this way, I will be able to track success of each historical success of each ETF for any given day, both in likelihood of positive return and average value of positive/negative return. The code will repeat this process for each calendar day to develop a "strategy" for the year.

PARAMETER SPACE

The parameter space Θ : the set of possible values for the unobserved parameters. For the example above, in which the data are generated as a set of i.i.d. normal random variables, the mean θ might be any real number, while the variance σ2 might be any non-negative real number: θ∈R and σ2≥0. The parameter space is then the set of all possible values of θ and σ2 : Θ=R×[0,∞).

I began to describe this above. By breaking each day into positive and negative returns, I will be able to track the percentage of positive days for each ETF on any given calendar day.

Ideally, I will be able to identify an ETF for each day that has more than a 50% chance of providing a positive return.

PRIOR BELIEFS

A description of the decision-maker's prior beliefs. These beliefs describe the information or beliefs the decision-maker has about the values of the unobserved parameters before collecting data. These beliefs can generally be represented as a probability distribution over the parameter space Θ. In the case in which data are normally distributed, it might be that, based on previous experience, the decision-maker has a strong belief that the mean parameter θ is greater than 4, and less than 6. She feels, though, that she has no basis for seeing any of the values between 4 and 6 as being more likely than any other. In this situation, she might represent her prior beliefs about θ by treating θ as a random variable drawn from a uniform distribution over the intervel [4,6] : θ U[4,6].

It is a commonly known fact that the stock market tends to average an 8% return. However, this is an average for the year, and I as the decision maker do not know much about any particular day. For the sake of simplicity, I will assume no prior beliefs and take the percentage of a particular investment returning a positive gain as 50%.

However, I have many years of data available to me which I will apply to my ignorant prior beliefs to build a more reliable model.

IMPROVING THE MODEL

A description of the process for getting better information, to sharpen the decision-maker's beliefs about uncertain parameter values. Bayesian analysis provides a rigorous procedure for combining prior information with sample data to generate updated beliefs. These beliefs are represented by the decision-maker's posterior distribution over the unobserved parameter values.

Every year adds new data to the model, albeit very slowly. However, I will assess the performance of my model's ability to accurately improve investments by taking smaller samples of data. For instance, I will write the code to have the ability to only use data from 2008-2017. In this way, I can allow the model to make a decision about 2018, and use the extra data I already have to prove the validity of the model. I also will have the ability to use January to April of 2020 as a secondary test.

PAYOFFS

The relevant payoffs that could result from the action. Payoffs will be a function of the selected action a, and the realized value of an uncertain random variable. In many applications, it makes sense to model payoffs as a function of the action a and the value of unobserved parameter(s) θ. In other applications, it may make sense to model payoffs as a function of the chosen action and a future value of a random state variable X. In a typical application, one action might best under some values of the uncertain parameter value, while others are better for other values. In such applications, the uncertainty genuinely matters. This situation creates incentives for getting better information, to reduce the decision-maker's uncertainty, in order to guide choices reliably towards actions that generate higher payoffs. When payoffs are uncertain as a function of the selected action a, it can be helpful to think of each action as associated with a lottery over a set of possible payoffs.

The payoffs will change based on the data set being observed and will be calculated directly using the historical data. For instance, in a situation where data from 2008 to 2019 is used, up to 12 different data points will exist for each calendar day (depending on where weekends fall because weekends are non-trading days with no data). For each calendar day, the data points will be separated by positive, negative, and net-zero return. The positive days will be averaged to find the positive payoff, and the negative days will be averaged to find the negative payoff.

UTILITY FUNCTION

The decision-maker's utility function or loss function.

Here is a sample utility function for choosing to invest in VTI on Feb 2:

$$EU(VTI_Feb2) = (AvgPosReturn_Feb2) \cdot (PercPosReturn_Feb2) + (AvgNegReturn_Feb2) \cdot (Perc$$

Here is an example from data collected from 2008 to 2019 for January 2, the first trading day of the year.

Decision	Avg Pos Return	Percent Pos Return	Avg Neg Return	Percent Neg Return	Estimated Utility
Invest in VTI	1.32	67	-0.62	33	0.6752
Invest in VXF	0.95	67	-0.63	33	0.4215
Invest in VBK	1.37	50	-0.64	50	0.3652

Decision	Avg Pos Return	Percent Pos Return	Avg Neg Return	Percent Neg Return	Estimated Utility
Invest in VEA	0.87	50	-0.64	33	0.2235
Invest in VWO	1.48	67	-1.49	33	0.4875
Invest in VPU	0.92	50	-1.18	50	-0.1283
Invest in BLV	0.31	67	-1.52	33	-0.2965

It is important to note that the chance of VTI returning positive and the chance of VTI returning negative will no necessarily add up to 1. There is a chance of VTI returning 0 return, and I will be sure to include this if it shows up in the data.

DECISION MAKING RULE

The rule the decision-maker will use to choose a preferred action $a*$ *from the menu A of possible actions.*

The model will find the utility function for each particular action and pick the action with the highest utility for each day of the year.

NOTES

1. I have been confused in the use of variables in class. I understand the methods and definitions in english, but get lost when we write variables on the board. In this document, all of the information is given in English. When we have our meetings about the projects, I want to discuss proper variable naming with you.

2. Unfortunatley, my decision making process does not actually work as is. The code is correct, but I am unable to find returns using my strategy and method that beat a typical hands-off buy-and-hold approach. Perhaps it is because my sample size is too small - perhaps it is because timing the market is actually impossible - if it were easy this probably would have been done before. I will keep working on my analysis but I want to discuss with you the possibility of having a result that doesn't actually improve performance from the benchmark.

MATLAB CODE

```
%%%%% SYS 6014 PROJECT - RYAN KRECHEL - NEW METHOD 2 %%%%%
clc; clear all; close all;

%Import all of the data and create ETF matrix to hold all data for later
ETF = zeros(3020,4,7);
VTI = csvread('VTI1.csv'); ETF(:,:,1) = VTI(:,:);
VXF = csvread('VXF2.csv'); ETF(:,:,2) = VXF(:,:);
VBK = csvread('VBK3.csv'); ETF(:,:,3) = VBK(:,:);
VEA = csvread('VEA4.csv'); ETF(:,:,4) = VEA(:,:);
VWO = csvread('VWO5.csv'); ETF(:,:,5) = VWO(:,:);
VPU = csvread('VPU6.csv'); ETF(:,:,6) = VPU(:,:);
BLV = csvread('BLV7.csv'); ETF(:,:,7) = BLV(:,:);
%Import a data set I created that sets up a "calendar" for my data
Dates = csvread('YearlyData.csv');

%This are the values I can change to manipulate the data my code processes
%I can search for whatever stretch I want to test my codes performance
StartYear=2008;
EndYear=2018;

%% Create a matrix to hold the averaged data for each day of the year
PosDays = zeros(366,4,7); PosDays(:,1:2,1) = Dates(:,:);
NegDays = zeros(366,4,7); NegDays(:,1:2,1) = Dates(:,:);
%Column 3 = percentage of days, Column 4 = average of days pos/neg

%% Populate the matrices
for fund = 1:7
    for day = 1:366
        %create temporary variables to use in assessing data as positive
        %and negative for each calendar day
        PosETFcount=0; PosETFtotal=0; NegETFcount=0; NegETFtotal=0; ZeroETFcount=0;
        for data = 1:3020
            %Check ETF and put in corresponding sheet of Pos and Neg Days
            if(ETF(data,1,fund)==PosDays(day,1,1) && ETF(data,2,fund)==PosDays(day,
2,1))
                %if between years of interest and value is positive
                if(ETF(data,3,fund)>=StartYear && ETF(data,3,fund)<=EndYear && ETF(
data,4,fund)>0)
                    %count number of days of positive return
                    PosETFcount=PosETFcount+1;
                    %keep the total value to get average at the end
                    PosETFtotal=PosETFtotal+ETF(data,4,fund);
                end
                %if between years of interest and value is negative
                if(ETF(data,3,fund)>=StartYear && ETF(data,3,fund)<=EndYear && ETF(
data,4,fund)<0)
                    %count number of negative days
                    NegETFcount=NegETFcount+1;
                    %keep the total value to get average at the end
                    NegETFtotal=NegETFtotal+ETF(data,4,fund);
                end
                %if between years of interest and value is 0
                if(ETF(data,3,fund)>=StartYear && ETF(data,3,fund)<=EndYear && ETF(
data,4,fund)==0)
                    %count number of net-zero days
                    ZeroETFcount=ZeroETFcount+1;
                end
            end
        end
        %add up how many trading days there were for those years
        TotalETFdays=PosETFcount+NegETFcount+ZeroETFcount;
        %only run the code below if the total trading days was >0
        %important because the code runs even for holidays which are never
        %trading days
        if TotalETFdays>0
            %calculate the percent of days return was positive
            PosDays(day,3,fund)=PosETFcount/TotalETFdays;
            %if there are any positive days, solve the average pos return
            if PosETFcount>0
                PosDays(day,4,fund)=PosETFtotal/PosETFcount;
            end
            %calculate the percent of days return was negative
            NegDays(day,3,fund)=NegETFcount/TotalETFdays;
            %if there are any negative days, solve the average neg return
            if NegETFcount>0
                NegDays(day,4,fund)=NegETFtotal/NegETFcount; %average positive
            end
        end
    end
end

%% Solve Utitlity Fucntions to make decisions
%create utility function vector to hold summary of data
Utility = zeros(366,12);
%put the dates in the last two columns
Utility(:,9:10) = PosDays(:,1:2);
for day = 1:366
    %calculate the utility for each etf on each day
    Utility(day,1)=PosDays(day,3,1)*PosDays(day,4,1)+NegDays(day,3,1)*NegDays(day,4
,1);
    Utility(day,2)=PosDays(day,3,2)*PosDays(day,4,2)+NegDays(day,3,2)*NegDays(day,4
,2);
    Utility(day,3)=PosDays(day,3,3)*PosDays(day,4,3)+NegDays(day,3,3)*NegDays(day,4
,3);
    Utility(day,4)=PosDays(day,3,4)*PosDays(day,4,4)+NegDays(day,3,4)*NegDays(day,4
,4);
    Utility(day,5)=PosDays(day,3,5)*PosDays(day,4,5)+NegDays(day,3,5)*NegDays(day,4
,5);
```

```
    Utility(day,6)=PosDays(day,3,6)*PosDays(day,4,6)+NegDays(day,3,6)*NegDays(day,4
,6);
    Utility(day,7)=PosDays(day,3,7)*PosDays(day,4,7)+NegDays(day,3,7)*NegDays(day,4
,7);
    %note that column 8 is "do not trade" and always has a utility=0
    %solve maximum utility and identify the decision choice
    %column 11 holds the etf number, 12 holds the estimated utility for
    %that day
    [Utility(day,12) Utility(day,11)]=max(Utility(day,1:8));
    if sum(Utility(day,1:7))==0
        Utility(day,11) = 8;
    end
end

%sum up the total estimated utility for the "strategy" comprised of each
%decsision for the whole year
TotalEstimatedUtility = sum(Utility(:,12))

%% Check actual return for each year using the defined strategy
%create matrix to hold the data
YearReturns = zeros(366,19);
%fill columns 1 and 2 with the date information
YearReturns(:,1:2)=Utility(:,9:10);
%go through each day for every year to find the actual return of
%theoretically following the strategy just solved for
for year = 8:19
    for day = 1:366
        for data = 1:3020
            %make sure the code accesses the correct data by date
            if(ETF(data,1,1)==YearReturns(day,1) && ETF(data,2,1)==YearReturns(day,
2) ...
                    && ETF(data,3,1)==2000+year && Utility(day,11)<8)
                %populate value with what would have been chosen given the
                %strategy was followed exactly
                YearReturns(day,year) = ETF(data,4,Utility(day,11));
            end
            %if the strategy chose option 8, "do not invest", automatically
            %set the return to 0
            if(Utility(day,11)==8)
                YearReturns(day,year)=0;
            end
        end
    end
end

%How did the strategy do for each year searched from StartYear to EndYear
PerformanceByYearPast=zeros(2,EndYear-StartYear+1);
PerformanceByYearPast(1,:)=StartYear-2000:EndYear-2000;
for year = 1:length(PerformanceByYearPast)
```

```
    PerformanceByYearPast(2,year)=sum(YearReturns(:,StartYear-2000+year-1));
end
%Print it out
PerformanceByYearPast

%check the years in the future after EndYear
%this only runs if your search used data before 2019
if(EndYear<2019)
    PerformanceByYearFuture=zeros(2,2019-EndYear);
    PerformanceByYearFuture(1,:)=EndYear-2000+1:2019-2000;
    for year = 1:2019-EndYear
        PerformanceByYearFuture(2,year)=sum(YearReturns(:,year+EndYear-2000));
    end
    PerformanceByYearFuture
end


%% Clear the Random and Unimportant Variables
clear data day fund value year Dates
clear NegETFcount NegETFtotal PosETFcount PosETFtotal TotalETFdays ZeroETFcount
clear VTI VXF VBK VEA VWO VPU BLV
```