Momentum Investing: An Application of Machine Learning to Factor Investing

Daniyal Kazimi*, Michael Lin, and Jeffrey Qiu

Abstract

Momentum investing is just one of numerous ways to apply machine learning to factor investing. This study aimed to develop a machine learning model based on momentum investing to see if we could accomplish a feat as old as the market itself: beating the market. While the model showed some promising results, we found that the rigidity of our methodology and the complexity of the market posed significant challenges. Specifically, the model struggled to account for certain market conditions and did not beat the market. These results suggest that momentum investing, on its own, may not be the optimal strategy for factor investing. However, further research could explore how momentum investing could be combined with other strategies to improve performance.

Keywords: Momentum investing, Factor investing, Machine learning, R

---

*We are grateful for Professor Tony S. Wirjanto (Department of Statistics and Actuarial Science and the School of Accounting and Finance, University of Waterloo) for his guidance on the completion of this paper as a submission for his course AFM 423: Topics in Financial Econometrics. Any errors and omissions are ours alone. Kazimi: School of Accounting and Finance, University of Waterloo, email: dkazimi@uwaterloo.ca. Lin: School of Accounting and Finance, University of Waterloo, email: michael.lin1@uwaterloo.ca. Qiu: School of Accounting and Finance, University of Waterloo, email: jeffrey.qiu@uwaterloo.ca.

1 Introduction

The purpose of this paper is to describe our methodology and report our findings regarding our endeavour to develop and evaluate a momentum-based investment learning model as an application of machine learning to factor investing.

Specifically, we sought to determine the optimal time horizon for momentum signals among daily, weekly or monthly signals, and to evaluate the model's performance under various conditions, such as economic recessions. The model would aim to buy a stock to capture positive momentum as profit and to sell a stock to during negative momentum to cut losses.

We hypothesized that the model would be generally successful in capturing positive momentum but would struggle during periods of market turmoil. This is because, generally speaking, when the market is doing well it will continue to do well which can be captured by momentum. However, during abnormal market conditions the market becomes unpredictable and we don't believe that tracking momentum will be a good method for determining the optimal action to take. To serve as a benchmark, we compared our model's performance with the opposite approach of momentum investing which is investing in an index of the market as a whole and holding the position.

2 Research Questions

In designing our experiment we strove to answer three research questions as stated below:

1. How do we address gaps in the datasets?
2. How do we ensure that the program can be executed in a reasonable timeframe?
3. Can this model be applicable to actual financial strategies?

First, addressing gaps is important as some data is only available on days the markets are open, and therefore can pose a problem for our weekly and monthly models as the observed date may not fall on one of those days.

Next, addressing program execution is important as longer runtimes make it more difficult to make adjustments/fix errors and make the model less accessible to other users.

Lastly, applicability is important because a model that depends on variables not available to a user (i.e. using future/current data to predict the past) or is unable to improve upon the results of a base case are not practical and therefore not useful to the user.

3 Variables and Measures

Looking at the variables used, we aimed to investigate the relationship between portfolio return, and the use of volatility/standard deviation predictors over different time frames. The response variable in this study was our portfolio value, which was measured by having an initial value of $1,000, and seeing how this value changes over time depending on the investment strategies tested. In the base case, the $1,000 was simply kept invested the entire duration of the S&P 500 index, a 'buy and hold' strategy. For the others, the amount was continually withdrawn and reinvested depending on the predictor variables. This response variable was used to collect data on how effective the investment strategy being tested was, mainly to see if the volatility/standard deviation predictors could outperform the base case. As for the predictor variables, there were various variables used in this study, including:

- S&P 500 Index Base Case: A simple $1,000 investment into the beginning of the 20 year measured period without any adjustments, this was generally used as a benchmark for other predictor variables.
- Standard Deviation Index 1 day/week/month: A portfolio with $1,000 to work with from the onset. This portfolio makes buy and sell decisions based on the standard

deviation of the market, in an attempt to buy into the market when there is positive momentum, and avoid investing into the market when there is negative momentum.

- Volatility Index 1 day/week/month: A portfolio with $1,000 to work with from the onset, that makes buy/sell decisions based on the volatility of the market, essentially trying to invest when volatility is at an ideal level.

4 Application of Machine Learning Approach to Factor Investing

Our application of machine learning to factor investing was to provide criteria so that the model can decide to do one of three options for every time period.

1. Buy
2. Hold
3. Sell

As a result, we also had to write the code to prevent any impossible decisions being made, such as a 'buy' decision being made when the amount given is already invested (there is no funds available to buy more), or a hold decision when the amount given is not currently invested (there is no stock to hold).

5 Experimental Methodology

In order to create our models we made use of several open-source R packages, those being tidyverse, lubridate, readxl, and quantmod. The bulk of our model depends on functions from tidyverse for machine learning and analytical purposes, with readxl and quantmod mostly being relied upon for data retrieval. We also adapted some programs produced by Tony Wirjanto, specifically those relating to neural networks. Our source code and all related files can be found in the Github repository. To address data gaps, we ran code blocks to substitute

gaps with a previous observation, as this ensures continuity in our models. To measure program execution length, we utilised time differentials (Functions from base R) between the start and ends of a code block. These measurements were taken while the program was run on an M1 Macbook Air; runtimes may vary significantly depending on one's computer hardware. Performance of each of the models are based on results compared to a base case 'buy and hold' strategy; these visualisations are presented in the next section.

6 Results and Discussion

In order to visualize the results of our models we created three line graphs, one for each model. These graphs plotted the value of each portfolio (Base, Volatility, and the STD) in USD across the entire time period we analyzed from 2003 to 2022. For visual clarity, we included two vertical axes to mark economic downturns, name the 2008 Financial Crisis and the 2020 COVID-19 pandemic. This allows us to more accurately observe any changes within these time periods of importance. Further, a horizontal axes was added at a value of $1,000 to help observe how far each portfolio has deviated from the initial investment.

Figure 1: Portfolio returns plot for momentum signal of 1 day

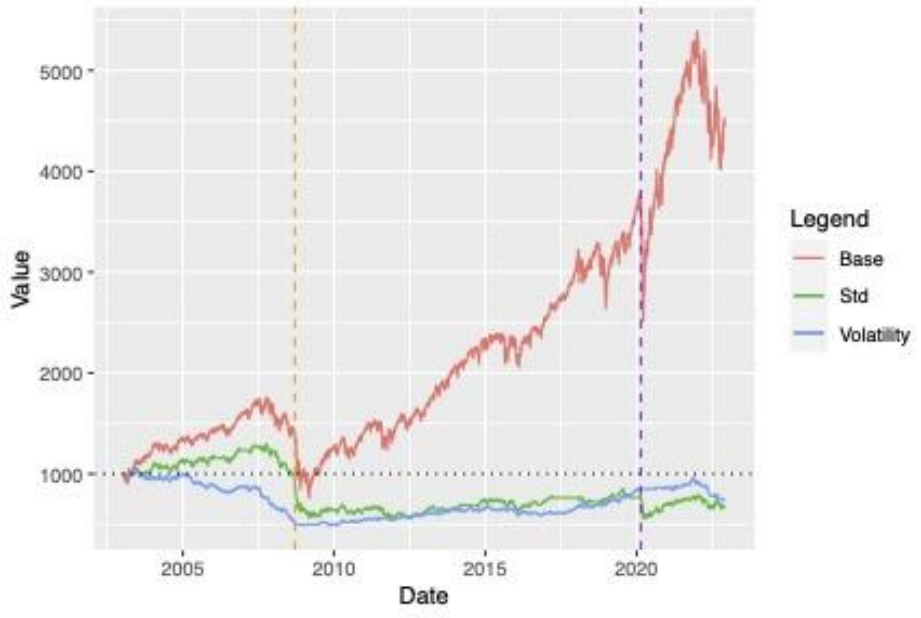


Figure 1 shows the change in portfolio values with the Volatility portfolio representing our momentum investing portfolio based on momentum signals for a timeframe of 1 day. We can observe that our Volatility portfolio never yields a positive return over the entire timeframe, let alone being able to match the base portfolio. That is to say, investing using this model would lose an investor money if they invested at the the start of the time period. The main dropoff in value was during 2007. Notably, the portfolio seems to gain value from 2008 onward. That is to say, the portfolio may have yielded a positive return if the investment was made beginning 2008. Instead, the minimal return could not offset the large initial loss.

In addressing our third research question, it is clear this model would likely not be applicable to actual financial strategies as its performance is even worse than if absolutely nothing was done. This is potentially an indication that looking at short-term timeframes for momentum signals are not an effective strategy for momentum investing. Alternatively, another possibility is that this model is especially sensitive to periods of market turmoil. However, further work would be required to confirm whether this is the case or if our model is unique in that way.

Figure 2: Portfolio returns plot for momentum signal of 7 days

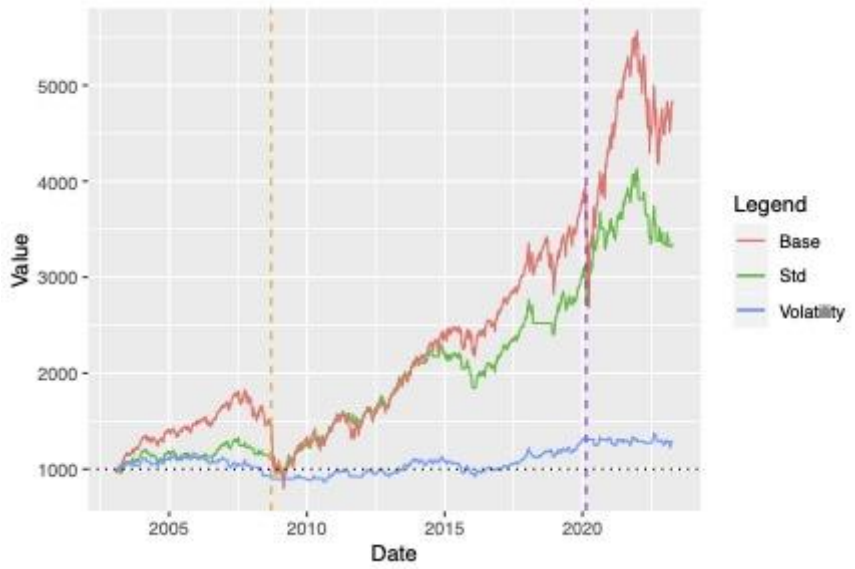


Figure 2 depicts the Volatility portfolio representing our momentum investing portfolio based on momentum signals for a timeframe of 7 days. The Volatility portfolio does end up yielding a positive return, albeit only by a slim margin and barely cresting a return of 40%. Given the large time horizon, it is difficult to consider this strong.

In addressing our third research question, it is unlikely this model would be applicable to actual financial strategies. If it were to be, a significant amount of additional work would need to be conducted in order to have it be viable. However, one key observation of note is that the Volatility portfolio appears much more resistant to periods of economic turmoil. During both the 2008 Financial Crisis and the 2020 COVID-19 pandemic, the volatility portfolio only experiences a minimal decrease in value compared to the large drops the Base portfolio, and the entire market, experienced. Thus, an area for future work could be drilling into the reason for this and possibly whether 7 day momentum signals could be viable for risk mitigation and hedging.

Figure 3: Portfolio returns plot for momentum signal of 30 days

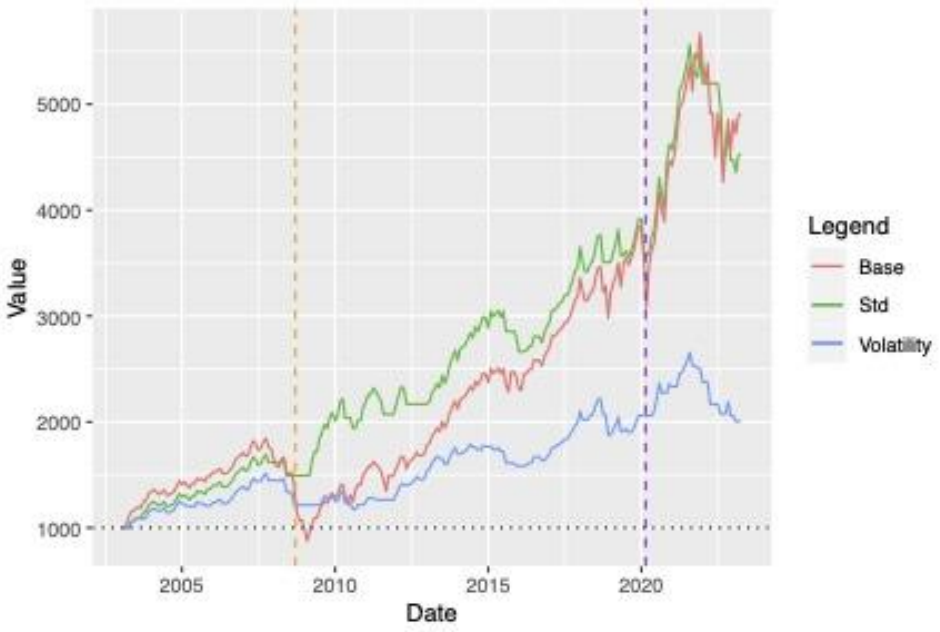
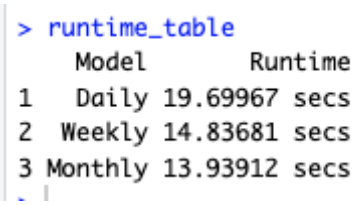


Figure 3 depicts the Volatility portfolio representing our momentum investing portfolio based on momentum signals for a timeframe of 30 days. We can see this model begins to show

reasonable promise, especially in comparison to the other two, and actually yields a fairly respectable return over the entire time period. While it is still significantly under that of the Base portfolio, this is a promising result nonetheless. Of note however, is that this model appears to be the most volatile in terms of the magnitude of change. Rises and falls are more drastic resulting in more distinct peaks and troughs. Notably, the standard deviation is also very high and is actually greater than the base portfolio for significant periods of time.

In addressing our third research question, this model is likely applicable to actual financial strategies if made more robust. The fact that it yielded a positive return is a positive sign and shows that there may be something worth looking into. An area for future work can be observing the correlation between longer momentum signal periods and return.

Figure 4. Runtime table for each model



```
> runtime_table
    Model       Runtime
1   Daily 19.69967 secs
2  Weekly 14.83681 secs
3 Monthly 13.93912 secs
```

Figure 4 depicts the runtime for each of our models. We can observe that the runtime for each are quite short with the Daily one taking the longest at 20 seconds and Monthly the shortest at 14 seconds. As these are all well under a minute, they can be considered acceptable and an indication that our model would be accessible to a wide range of hardware capabilities for the purposes of reproducibility and further work. The variation in times is also logical as more granular timeframes result in a higher frequency of data points and thus requires greater time to process.

Figure 5. Plot of three standard deviations

Figure 5 depicts the three different standard deviations we considered for our analysis. As can be observed, single STD results in the highest portfolio value. We determined the reason for this being that double STD ended up taking the action of "hold" far too often and thus the model rarely actually acted. Conversely, using half STD resulted in the model almost always taking the action of "sell". Thus, a single STD was a good intermediate point wherein the model had the most reasonable balance between all three actions.

7 Related Work

Projects 1 and 2 are highly related, therefore the reader is referred to our GPR #1 (Momentum Investing and Machine Learning Applications to Factor Investing, Kazimi et al., 2023) for related work.

8 Conclusions

In this paper, we discover that while momentum investing is a possible application of machine learning to factor investing, it may not be the ideal one and there is much room for further work and analysis. Of note, our experimental results indicted that longer periods of momentum signal correlate to a more positive return. Further, weekly momentum signals specifically seem to be especially resistant to periods of market turmoil. In consideration of further work to be conducted, these are both focus areas we would deem to be worth examining further.

References

Beaudan, P. & He, S. (January 30, 2019). Applying Machine Learning to Trading Strategies: Using Logistic Regression to Build Momentum-Based Trading Strategies. Available at SSRN: https://ssrn.com/abstract=3325656 or http://dx.doi.org/10.2139/ssrn.3325656

Borisenko, D. (July 23, 2019). Dissecting Momentum: We Need to Go Deeper. Available at SSRN: https://ssrn.com/abstract=3424793 or http://dx.doi.org/10.2139/ssrn.3424793

Calomiris, C. & Mamaysky, H. (2019). How news and its context drive risk and returns around the world. Journal of Financial Economics, 133(2), 299–336. Available at https://doi.org/10.1016/j.jfineco.2018.11.009

Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy with lubridate. Journal of Statistical Software, 40(3), 1-25. URL https://www.jstatsoft.org/v40/i03/.

Ryan JA, Ulrich JM (2023). _quantmod: Quantitative Financial Modelling Framework_. R package version 0.4.22, <https://CRAN.R-project.org/package=quantmod>.

Wirjanto, T. S. (2023). Topics in Financial Econometrics.

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse." _Journal of Open Source Software_, *4*(43), 1686. doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.

Wickham H, Bryan J (2023). _readxl: Read Excel Files_. R package version 1.4.2, <https://CRAN.R-project.org/package=readxl>.

Appendix

```
library(tidyverse)              # Activate the data science package
library(lubridate)              # Activate the date management package
library(readxl)
library(quantmod)

citation(package = "tidyverse")

#https://www.cboe.com/tradable_products/vix/
#https://fred.stlouisfed.org/series/SP500#0

#####
#Daily

data_spday <- read_csv('dayMetaData.csv')
data_spday <- data_spday %>%
  mutate(Date=as.Date(Date, format = "%m/%d/%Y"))

### Base Portfolio (Invest at start and hold)
data_spday <- data_spday %>%
  mutate(BasePortfolio=0)

for (i in 1:5000) {
  data_spday <- data_spday %>%
        mutate(BasePortfolio= if_else(Date == "2003-01-21", 1000,
                        lag(BasePortfolio)+(lag(BasePortfolio)*DailyReturn/100)))
}

###Portfolio Decision Based on Standard Deviation (Price Momentum)
start_timeD <- Sys.time()
#Standard Deviation and Buy/Hold/Sell (+1/0/-1) Decision Calculation
spday_bar <- median(data_spday$DailyReturn)
data_spday <- data_spday %>%
  mutate(neg_change = -sd(DailyReturn),
        pos_change = sd(DailyReturn))
data_spday <- data_spday %>%
  mutate(PortDecision = if_else(DailyReturn < neg_change, -1,
                        if_else(DailyReturn > pos_change, 1,0)),
        PortDecision = as.factor(PortDecision))
```

```
#Determining Ownership for Portfolio Calculation Purposes
data_spday <- data_spday %>%
  mutate(Owned=0)

for (i in 1:5000) {
data_spday <- data_spday %>%
  mutate(Owned = ifelse(Date == "2003-01-21", 0,
                        ifelse(lag(Owned) == 0 & (lag(PortDecision) == 0 | lag(PortDecision == -1)), 0,
                               ifelse(lag(Owned) == 0 & lag(PortDecision) == 1, 1,
                               ifelse(lag(Owned) == 1 & lag(PortDecision) == -1, 0, 1)))))
}

#Applying Returns if Index is 'owned' otherwise no change
data_spday <- data_spday %>%
  mutate(MomentumPortfolio=0)

for (i in 1:5000) {
  data_spday <- data_spday %>%
          mutate(MomentumPortfolio= if_else(Date == "2003-01-21", 1000,

if_else(Owned==1,lag(MomentumPortfolio)+(lag(MomentumPortfolio)*DailyReturn/100),
                                        lag(MomentumPortfolio))))
}
end_timeD <- Sys.time()
net_timeD <- end_timeD - start_timeD
net_timeD

### Portfolio decision based on volatility neural network
data_spday <- data_spday %>%
  mutate(Date=as.Date(Date, format = "%m/%d/%Y"))


getSymbols.FRED("VIXCLS",
          env = ".GlobalEnv",
          return.class = "xts")
vix <- fortify(VIXCLS)
colnames(vix) <- c("Date", "vix")

data_vixday <- merge(data_spday, vix, by = "Date")

delta <- 0.5
vix_bar <- median(data_vixday$vix)
data_vixday <- data_vixday %>%
  dplyr::select(Date, DailyReturn) %>%
  mutate(r_minus = (-0.02) * exp(-delta*(data_vixday$vix-vix_bar)),
          r_plus = 0.02 * exp(delta*(data_vixday$vix-vix_bar)))
data_vixday <- data_vixday %>%
  mutate(PortDecisionVix = if_else(DailyReturn < r_minus, -1, if_else(DailyReturn > r_plus, 1,0)),
          PortDecisionVix = as.factor(PortDecisionVix))

# Determining Ownership for Portfolio Calculation Purposes
data_vixday <- data_vixday %>%
  mutate(Owned=0)

for (i in 1:5000) {
  data_vixday <- data_vixday %>%
          mutate(Owned = ifelse(Date == "2003-01-21", 0,
                    ifelse(lag(Owned) == 0 & (lag(PortDecisionVix) == 0 | lag(PortDecisionVix == -1)), 0,
                               ifelse(lag(Owned) == 0 & lag(PortDecisionVix) == 1, 1,
```

```
                              ifelse(lag(Owned) == 1 & lag(PortDecisionVix) == -1, 0, 1)))))
}

#Applying Returns if Index is 'owned' otherwise no change
data_vixday <- data_vixday %>%
  mutate(VixPortfolio=0)

for (i in 1:5000) {
  data_vixday <- data_vixday %>%
          mutate(VixPortfolio= if_else(Date == "2003-01-21", 1000,

if_else(Owned==1,lag(VixPortfolio)+(lag(VixPortfolio)*DailyReturn/100),
                                       lag(VixPortfolio))))
}

DayPortfolios <- merge(data_spday, data_vixday, by = "Date")

DayPortfolios <- DayPortfolios %>%
  select(Date, MomentumPortfolio, BasePortfolio, VixPortfolio)

ggplot(DayPortfolios, aes(x = Date)) +
  geom_line(aes(y = MomentumPortfolio, color = "Std")) +
  geom_line(aes(y = BasePortfolio, color = "Base")) +
  geom_line(aes(y = VixPortfolio, color = "Volatility")) +
  labs(x = "Date", y = "Value", color = "Legend")+
  geom_hline(yintercept=1000, linetype="dotted")+
  geom_vline(xintercept = as.numeric(as.Date("2008-09-15")), color = "Orange", linetype = "dashed")+
  geom_vline(xintercept = as.numeric(as.Date("2020-02-20")), color = "Purple", linetype = "dashed")


#####
# Weekly
#Data Processing
weekmetadata <- read_csv('Weekmetadata.csv')
weekmetadata <- weekmetadata[-c(1,2),]
weekmetadata <- weekmetadata %>%
  mutate(Date=as.Date(Date, format = "%m/%d/%Y"))%>%
  mutate(Date=Date-2) #The data downloaded ends each week on the sunday, to make the merge
with VIX later easier, we subtract two to make it end on Fridays

### Base Portfolio
data_spweek <- weekmetadata %>%
  mutate(BasePortfolio=0)

for (i in 1:5000) {
  data_spweek <- data_spweek %>%
          mutate(BasePortfolio= if_else(Date == "2003-01-24", 1000,
                              lag(BasePortfolio)+(lag(BasePortfolio)*WeeklyReturn/100)))
}


### Portfolio Decision Based on Standard Deviation (Price Momentum)
start_timeW <- Sys.time()
sp_barweek <- median(data_spweek$WeeklyReturn)
data_spweek <- data_spweek %>%
  mutate(neg_change = -sd(WeeklyReturn),
          pos_change = sd(WeeklyReturn))
data_spweek <- data_spweek %>%
  mutate(PortDecision = if_else(WeeklyReturn < neg_change, -1,
                              if_else(WeeklyReturn > pos_change, 1,0)),
```

```
        PortDecision = as.factor(PortDecision))

data_spweek <- data_spweek %>%
  mutate(Owned=0)

for (i in 1:5000) {
  data_spweek <- data_spweek %>%
        mutate(Owned = ifelse(Date == "2003-01-24", 0,
                ifelse(lag(Owned) == 0 & (lag(PortDecision) == 0 | lag(PortDecision == -1)), 0,
                        ifelse(lag(Owned) == 0 & lag(PortDecision) == 1, 1,
                        ifelse(lag(Owned) == 1 & lag(PortDecision) == -1, 0, 1)))))
}

data_spweek <- data_spweek %>%
  mutate(MomentumPortfolio=0)

for (i in 1:5000) {
  data_spweek <- data_spweek %>%
        mutate(MomentumPortfolio= if_else(Date == "2003-01-24", 1000,

if_else(Owned==1,lag(MomentumPortfolio)+(lag(MomentumPortfolio)*WeeklyReturn/100),
                                        lag(MomentumPortfolio))))
}

end_timeW <- Sys.time()
net_timeW <- end_timeW - start_timeW
net_timeW

### Portfolio decision based on volatility neural network
getSymbols.FRED("VIXCLS",
        env = ".GlobalEnv",
        return.class = "xts")
vix <- fortify(VIXCLS)
colnames(vix) <- c("Date", "vix")

for (i in 1:8687) {
  vix <- vix %>%
        mutate(vix= if_else(is.na(vix), lag(vix), vix))
}

data_vixweek <- merge(data_spweek, vix, by = "Date")



delta <- 0.5
vix_bar <- median(data_vixweek$vix)
data_vixweek <- data_vixweek %>%
  dplyr::select(Date, WeeklyReturn) %>%
  mutate(r_minus = (-0.02) * exp(-delta*(data_vixweek$vix-vix_bar)),
        r_plus = 0.02 * exp(delta*(data_vixweek$vix-vix_bar)))
data_vixweek <- data_vixweek %>%
  mutate(PortDecisionVix = if_else(WeeklyReturn < r_minus, -1, if_else(WeeklyReturn > r_plus, 1,0)),
        PortDecisionVix = as.factor(PortDecisionVix))


data_vixweek <- data_vixweek %>%
  mutate(Owned=0)

for (i in 1:5000) {
  data_vixweek <- data_vixweek %>%
```

```
        mutate(Owned = ifelse(Date == "2003-01-24", 0,
                ifelse(lag(Owned) == 0 & (lag(PortDecisionVix) == 0 | lag(PortDecisionVix == -1)), 0,
                        ifelse(lag(Owned) == 0 & lag(PortDecisionVix) == 1, 1,
                        ifelse(lag(Owned) == 1 & lag(PortDecisionVix) == -1, 0, 1)))))
}


data_vixweek <- data_vixweek %>%
  mutate(VixPortfolio=0)

for (i in 1:5000) {
  data_vixweek <- data_vixweek %>%
        mutate(VixPortfolio= if_else(Date == "2003-01-24", 1000,
                        if_else(Owned==1,lag(VixPortfolio)+(lag(VixPortfolio)*WeeklyReturn/100),
                        lag(VixPortfolio))))
}

WeekPortfolios <- merge(data_spweek, data_vixweek, by = "Date")

WeekPortfolios<-WeekPortfolios %>%
  select(Date, MomentumPortfolio, BasePortfolio, VixPortfolio)

ggplot(WeekPortfolios, aes(x = Date)) +
  geom_line(aes(y = MomentumPortfolio, color = "Std")) +
  geom_line(aes(y = BasePortfolio, color = "Base")) +
  geom_line(aes(y = VixPortfolio, color = "Volatility")) +
  labs(x = "Date", y = "Value", color = "Legend")+
  geom_hline(yintercept=1000, linetype="dotted")+
  geom_vline(xintercept = as.numeric(as.Date("2008-09-15")), color = "Orange", linetype = "dashed")+
  geom_vline(xintercept = as.numeric(as.Date("2020-02-20")), color = "Purple", linetype = "dashed")



#####
# Monthly
monthlymetadata <- read_csv('Monthmetadata.csv')
monthlymetadata <- monthlymetadata %>%
  mutate(Date=as.Date(Date, format = "%m/%d/%Y"))

### Base Portfolio
data_spmonth <- monthlymetadata %>%
  mutate(BasePortfolio=0)

for (i in 1:5000) {
  data_spmonth <- data_spmonth %>%
        mutate(BasePortfolio= if_else(Date == "2003-02-01", 1000,
                        lag(BasePortfolio)+(lag(BasePortfolio)*MonthlyReturn/100)))
}

### Portfolio Decision Based on Standard Deviation
start_timeM <- Sys.time()
### Standard Deviation Calculation


sp_barmonth <- median(data_spmonth$MonthlyReturn)
data_spmonth <- data_spmonth %>%
  mutate(neg_change = -sd(MonthlyReturn),
        pos_change = sd(MonthlyReturn))
data_spmonth <- data_spmonth %>%
  mutate(PortDecision = if_else(MonthlyReturn < neg_change, -1,
```

```
                                if_else(MonthlyReturn > pos_change, 1,0)),
          PortDecision = as.factor(PortDecision))

data_spmonth <- data_spmonth %>%
  mutate(Owned=0)

for (i in 1:5000) {
  data_spmonth <- data_spmonth %>%
          mutate(Owned = ifelse(Date == "2003-02-01", 0,
                    ifelse(lag(Owned) == 0 & (lag(PortDecision) == 0 | lag(PortDecision == -1)), 0,
                              ifelse(lag(Owned) == 0 & lag(PortDecision) == 1, 1,
                              ifelse(lag(Owned) == 1 & lag(PortDecision) == -1, 0, 1)))))
}

data_spmonth <- data_spmonth %>%
  mutate(MomentumPortfolio=0)

for (i in 1:5000) {
  data_spmonth <- data_spmonth %>%
          mutate(MomentumPortfolio= if_else(Date == "2003-02-01", 1000,

if_else(Owned==1,lag(MomentumPortfolio)+(lag(MomentumPortfolio)*MonthlyReturn/100),
                                        lag(MomentumPortfolio))))
}

end_timeM <- Sys.time()
net_timeM <- end_timeM - start_timeM
net_timeM

### Portfolio decision based on volatility neural network
getSymbols.FRED("VIXCLS",
          env = ".GlobalEnv",
          return.class = "xts")

# convert to monthly data
vix_monthly <- to.period(VIXCLS, period = "months", indexAt = "firstof", OHLC = FALSE)

# fortify and rename columns
vix_monthly <- fortify(vix_monthly)
colnames(vix_monthly) <- c("Date", "vix")

vix_monthly <- vix_monthly %>%
  mutate(Date=as.Date(Date, format = "%m/%d/%Y"))

data_vixmonth <- merge(data_spmonth, vix_monthly, data_spmonth = "Date")

#Just incase there are back-toback N/A
for (i in 1:8687) {
  vix_monthly <- vix_monthly %>%
          mutate(vix= if_else(is.na(vix), lag(vix), vix))
}


delta <- 0.5
vix_bar <- median(data_vixmonth$vix)
data_vixmonth <- data_vixmonth %>%
  dplyr::select(Date, MonthlyReturn) %>%
  mutate(r_minus = (-0.02) * exp(-delta*(data_vixmonth$vix-vix_bar)),
          r_plus = 0.02 * exp(delta*(data_vixmonth$vix-vix_bar)))
data_vixmonth <- data_vixmonth %>%
```

```
  mutate(PortDecisionVix = if_else(MonthlyReturn < r_minus, -1, if_else(MonthlyReturn > r_plus, 1,0)),
         PortDecisionVix = as.factor(PortDecisionVix))


data_vixmonth <- data_vixmonth %>%
  mutate(Owned=0)

for (i in 1:5000) {
  data_vixmonth <- data_vixmonth %>%
         mutate(Owned = ifelse(Date == "2003-02-01", 0,
                  ifelse(lag(Owned) == 0 & (lag(PortDecisionVix) == 0 | lag(PortDecisionVix == -1)), 0,
                         ifelse(lag(Owned) == 0 & lag(PortDecisionVix) == 1, 1,
                         ifelse(lag(Owned) == 1 & lag(PortDecisionVix) == -1, 0, 1)))))
}


data_vixmonth <- data_vixmonth %>%
  mutate(VixPortfolio=0)

for (i in 1:5000) {
  data_vixmonth <- data_vixmonth %>%
         mutate(VixPortfolio= if_else(Date == "2003-02-01", 1000,
                         if_else(Owned==1,lag(VixPortfolio)+(lag(VixPortfolio)*MonthlyReturn/100),
                         lag(VixPortfolio))))
}

MonthPortfolios <- merge(data_spmonth, data_vixmonth, by = "Date")

MonthPortfolios<-MonthPortfolios %>%
  select(Date, MomentumPortfolio, BasePortfolio, VixPortfolio)

ggplot(MonthPortfolios, aes(x = Date)) +
  geom_line(aes(y = MomentumPortfolio, color = "Std")) +
  geom_line(aes(y = BasePortfolio, color = "Base")) +
  geom_line(aes(y = VixPortfolio, color = "Volatility")) +
  labs(x = "Date", y = "Value", color = "Legend")+
  geom_hline(yintercept=1000, linetype="dotted")+
  geom_vline(xintercept = as.numeric(as.Date("2008-09-15")), color = "Orange", linetype = "dashed")+
  geom_vline(xintercept = as.numeric(as.Date("2020-02-20")), color = "Purple", linetype = "dashed")

summary(MonthPortfolios)

###
#Sensitivity Analysis
#Half Stdev Portfolio
data_sensitivity <- data_spmonth %>%
  mutate(neg_changeHalf = -0.5*sd(MonthlyReturn),
         pos_changeHalf = 0.5*sd(MonthlyReturn))
data_sensitivity <- data_sensitivity %>%
  mutate(PortDecisionHalf = if_else(MonthlyReturn < neg_changeHalf, -1,
                         if_else(MonthlyReturn > pos_changeHalf, 1,0)),
         PortDecisionHalf = as.factor(PortDecisionHalf))

data_sensitivity <- data_sensitivity %>%
  mutate(OwnedHalfStd=0)

for (i in 1:5000) {
  data_sensitivity <- data_sensitivity %>%
         mutate(OwnedHalfStd = ifelse(Date == "2003-02-01", 0,
```

```
                    ifelse(lag(OwnedHalfStd) == 0 & (lag(PortDecisionHalf) == 0 | lag(PortDecisionHalf
== -1)), 0,
                                ifelse(lag(OwnedHalfStd) == 0 & lag(PortDecisionHalf) == 1, 1,
                                ifelse(lag(OwnedHalfStd) == 1 & lag(PortDecisionHalf) == -1, 0, 1)))))
}

data_sensitivity <- data_sensitivity %>%
  mutate(HalfPortfolio=0)

for (i in 1:5000) {
  data_sensitivity <- data_sensitivity %>%
          mutate(HalfPortfolio= if_else(Date == "2003-02-01", 1000,

if_else(OwnedHalfStd==1,lag(HalfPortfolio)+(lag(HalfPortfolio)*MonthlyReturn/100),
                                        lag(HalfPortfolio))))
}

#Double Stdev Portfolio
data_sensitivity <- data_sensitivity %>%
  mutate(neg_changeDbl = -2*sd(MonthlyReturn),
          pos_changeDbl = 2*sd(MonthlyReturn))
data_sensitivity <- data_sensitivity %>%
  mutate(PortDecisionDbl = if_else(MonthlyReturn < neg_changeDbl, -1,
                                if_else(MonthlyReturn > pos_changeDbl, 1,0)),
          PortDecisionDbl = as.factor(PortDecisionDbl))

data_sensitivity <- data_sensitivity %>%
  mutate(OwnedDblStd=0)

for (i in 1:5000) {
  data_sensitivity <- data_sensitivity %>%
          mutate(OwnedDblStd = ifelse(Date == "2003-02-01", 0,
                                ifelse(lag(OwnedDblStd) == 0 & (lag(PortDecisionDbl) == 0 |
lag(PortDecisionDbl == -1)), 0,
                                ifelse(lag(OwnedDblStd) == 0 & lag(PortDecisionDbl) == 1, 1,
                                        ifelse(lag(OwnedDblStd) == 1 & lag(PortDecisionDbl) == -1, 0, 1)))))
}

data_sensitivity <- data_sensitivity %>%
  mutate(DblPortfolio=0)

for (i in 1:5000) {
  data_sensitivity <- data_sensitivity %>%
          mutate(DblPortfolio= if_else(Date == "2003-02-01", 1000,

if_else(OwnedDblStd==1,lag(DblPortfolio)+(lag(DblPortfolio)*MonthlyReturn/100),
                                lag(DblPortfolio))))
}

colnames(data_sensitivity)[13] = "SinglePortfolio"

sensitivityPortfolios<- data_sensitivity %>%
  select(Date, SinglePortfolio, HalfPortfolio, DblPortfolio)

ggplot(sensitivityPortfolios, aes(x = Date)) +
  geom_line(aes(y = SinglePortfolio, color = "Single")) +
  geom_line(aes(y = HalfPortfolio, color = "Half")) +
  geom_line(aes(y = DblPortfolio, color = "Double")) +
  labs(x = "Date", y = "Value", color = "Legend")+
  geom_hline(yintercept=1000, linetype="dotted")+
```

```
  geom_vline(xintercept = as.numeric(as.Date("2008-09-15")), color = "Orange", linetype = "dashed")+
  geom_vline(xintercept = as.numeric(as.Date("2020-02-20")), color = "Purple", linetype = "dashed")

#####
#Program Execution Table
#The resulting outputs can vary slightly, but multiple runs have seen similar results

runtime_table <- data.frame(Model=c('Daily','Weekly','Monthly'),
                            Runtime=c(net_timeD,net_timeW,net_timeM))
runtime_table
```