Table of Contents

Introduction:

In finance, technical analysis is somewhat of a pseudoscience for predicting price movements. The efficient market hypothesis states that asset prices reflect all available information and asset prices cannot be predicted. This does not stop a lot of people from trying. Technical Analysis can be traced back to the Dutch financial markets in the 1600s. The candlestick techniques used in this analysis were developed by Homma Munehisa, a grain farmer in Japan, in the 1700s. This is a highly controversial topic, but as the processing power of computers has increased vastly, the ability to apply advanced statistical techniques have led to breakthroughs in technical analysis.

A technical analyst looks to predict trends, patterns and prices of assets. In Economics, all metrics used to predict any asset price are considered "lagging indicators". Lagging indicators confirm trends and changes in trends. Prior to the wide use and explosion of computation power, analysts used patterns/shapes as trend analysis to try and predict price patterns. These were innovated and explored by Ralph Nelson Elliot in *Nature's Law: The Secret of the Universe.* The patterns Elliot explored included head and shoulders patterns, double top/bottom to predict reversals. He mostly charted triangles in seasonal and stochastic processes and tried to gauge the strength and oscillations of the occurrences. With more computing power, these trend chart analyses have blended into a more mathematical approach. Popular methods used to try to predict and capture trend reversals include moving averages and RSI (relative strength). A more mathematical approach to Elliot's ideas can be explored.

There is still a large divide between uses of fundamental and technical analysis, however more people are adopting the use of both techniques in their research. Nassim Nicholas Taleb has a much more pessimistic view, in his book *The Black Swan* he reiterates the unpredictability of everything due to the impact of rare "Black Swan" events. As usual, maybe the truth lies somewhere in the middle. Maybe Elliot was onto something with the use of triangles, such is the basis for k-means clustering.

EDA and Models:

This model attempts to use machine learning to help improve a simple exponential moving average (EMA) trading strategy. The original strategy is an "EMA Crossover" Strategy. The goal of this strategy is to trigger a buy order when the short ema is greater than the long ema, and trigger a sell order when the short ema is below the long ema. In this analysis a one week and a two-week EMA is used. We

expect to see more frequent crossovers as the price of the asset is so volatile. This strategy is then combined with a machine learning clustering algorithm, in this case K-means. The K-means algorithm sees its origin in signal processing, and partitions data points/observations by minimizing the variance between points and ultimately assigns the cluster as a function of the distance to its mean (or centroid) depending on a predefined distance measure. (Josh Starmer, Statquest)

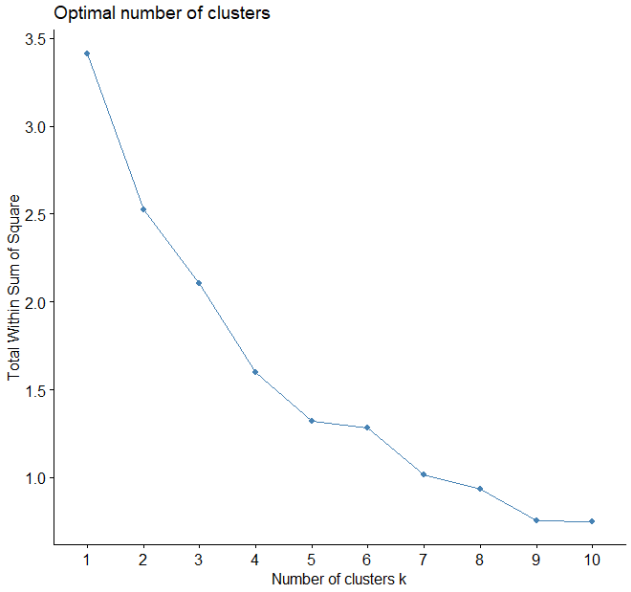
The price data was pulled from the “Binance API” (Binary-Finance) which is one of the largest cryptocurrency exchanges. The data was partitioned and exported to a csv for reproducibility. Buy and sell signals were developed (this can be referenced in the code in the appendix).

The image below represents a plot of the price (with candlesticks) as well as the EMAs. The red circles represent the crossovers, or when the short ema crossed above or below the long ema.

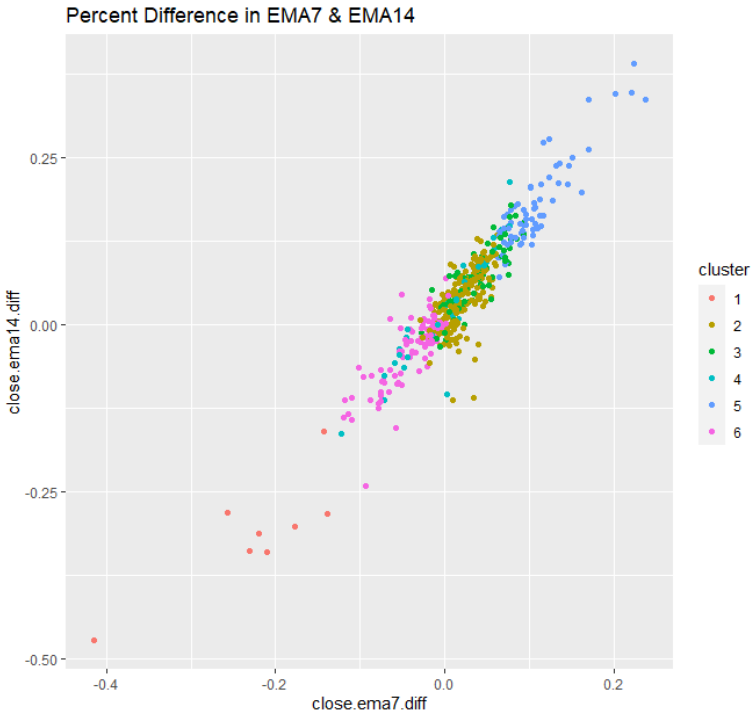
The clustering algorithm will be applied so that it clusters the percent difference in price on both sides of the ema if there is a crossover. This will enhance the decision-making ability of the computer as it will be able to continually check cluster percent change differences to detect a significant up trend to optimize a buy in price and/or the start of a downtrend to detect an optimal point to sell. The “candles” or colored price lines are simply (green if price closed above previous close, red if price closed below previous close, and the gray represents the variance throughout the day (high or low)).

The data was normalized/standardized by percent change. And converting buy and sell signals to binary(yes/no) decisions. As seen in the previous image the data is biased upwards, I did not perform a log/power transform because I thought it was important to see the full effect of the price changes and relative difference from the close prices.

PCA analysis. From the PCA analysis below we see an inflection point around five or six clusters. This visually represents the minimization of the weighted sum of squares or differenced data. This report used six clusters in the analysis.

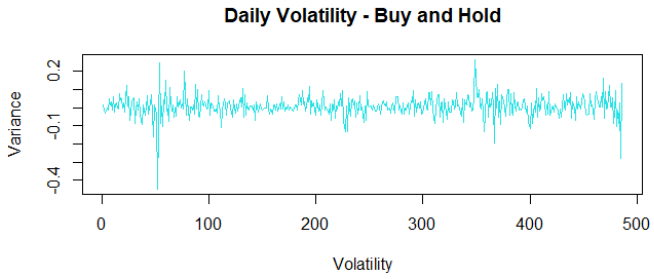


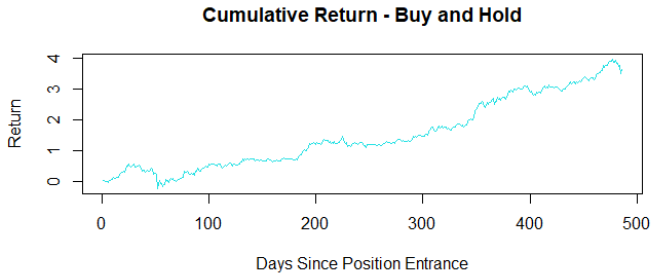
k-means clustering on financial data. The plot on the left shows the clusters assigned from the k-means algorithm. In this analysis all of the clusters associated with negative differences between the ema7 and ema 14 were assigned to the sell logic. Where as any cluster associated with large positive differences between the ema 7 and ema 14 were assigned to the buy/ or an entry signal to a position. This may have been better optimized by leaving out the middle clusters.



Results:

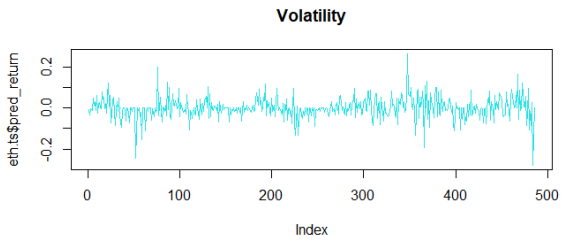
It was necessary to calculate daily returns in provide a cumulative sum of returns to test the algorithms performance (technical term is backtesting.) The machine learning application was tested between a buy and hold strategy and strict implementation of the ema crossover strategy. Please reference the following charts

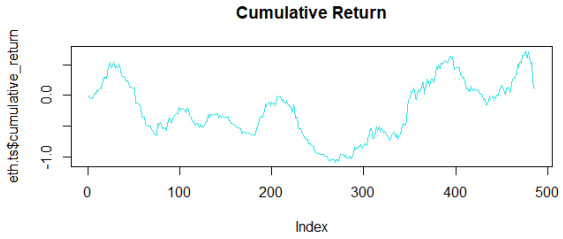


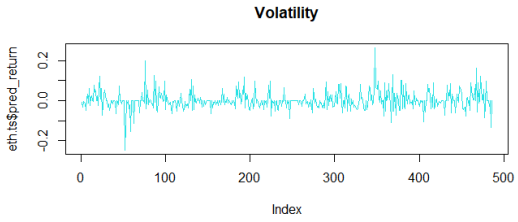


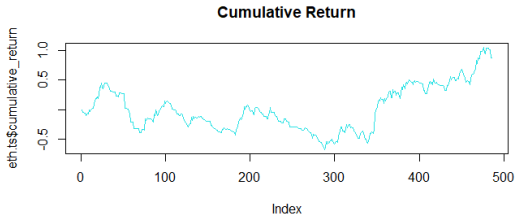
The plot on the left is a visual representation of the volatility, one interesting point in the graph is the significant 40% downturn. Since the position entrance Ethereum has increased nearly 400%.

The plot on the right is the two ema crossover strategy. The large downturn at the end of the dataset significantly impacted the cumulative return. The return over the period was 11.75%









The plots on the left represent the return and volatility of the ema strategy enhanced with the K-means clustering algorithm. The cumulative return is 87.30% over the period.

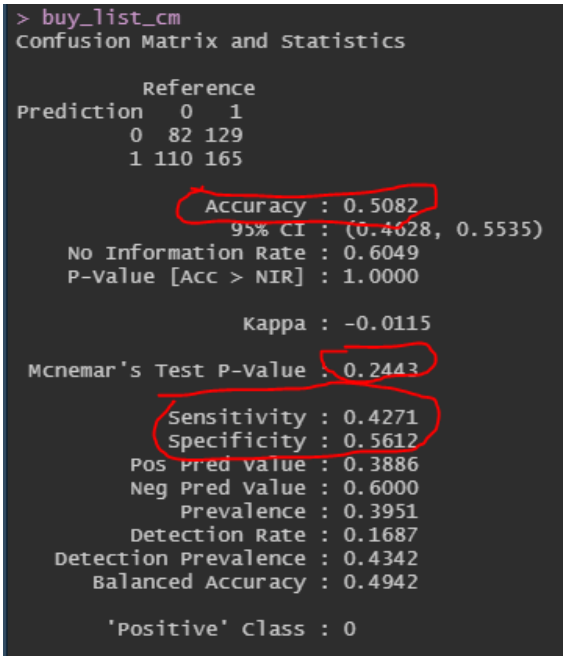
The simple ema crossover strategy was able to decrease the variance but its return is pathetic compared to the buy and hold strategy. The downside risk is mitigated slightly as the downtrend is recorded at a little over -20% compared to the buy and hold strategy of around -40%.

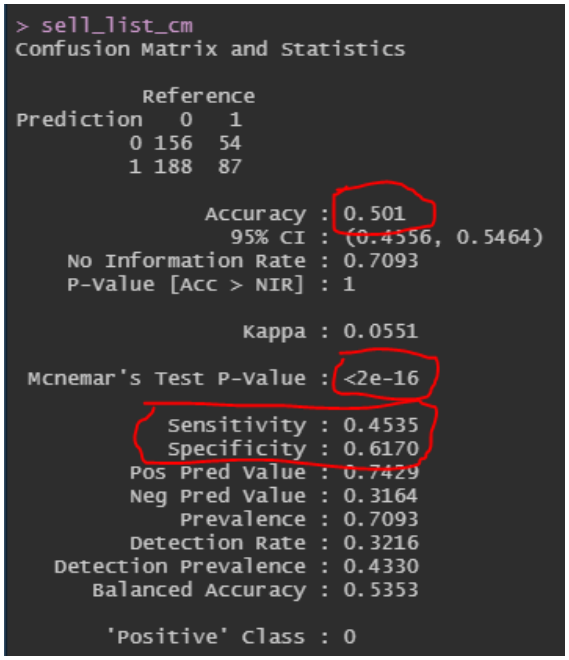
The application of the K-means algorithm decreased some variance while generating returns similar to the buy and hold strategy (still about 300% less). The ema strategy enhanced with the K-means clustering did statistically improve the selling ability and reduced the risk of downside loss, therefore increasing the efficiency of the crossover. This is shown below by performing analysis on each strategies' respective confusion matrix. The accuracy of the two ema crossover strategy models were both proven significant. As in both buy signals and sell signals generated had predictive quality. However, the accuracy to buy was slightly above 50%, whereas the model to sell was poor, underperforming a coin flip.

```
> buy_list_cm
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  47 164
         1  75 200

               Accuracy : 0.5082
                 95% CI : (0.4628, 0.5535)
    No Information Rate : 0.749
    P-Value [Acc > NIR] : 1

                  Kappa : -0.0526

 Mcnemar's Test P-Value : 1.254e-08

            Sensitivity : 0.38525
            Specificity : 0.54945
         Pos Pred Value : 0.22275
         Neg Pred Value : 0.72727
             Prevalence : 0.25103
         Detection Rate : 0.09671
   Detection Prevalence : 0.43416
      Balanced Accuracy : 0.46735

       'Positive' Class : 0
```

```
> sell_list_cm
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 164  46
         1 200  75

               Accuracy : 0.4928
                 95% CI : (0.4474, 0.5382)
    No Information Rate : 0.7505
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0494

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.4505
            Specificity : 0.6198
         Pos Pred Value : 0.7810
         Neg Pred Value : 0.2727
             Prevalence : 0.7505
         Detection Rate : 0.3381
   Detection Prevalence : 0.4330
      Balanced Accuracy : 0.5352

       'Positive' Class : 0
```

When enhanced with the K-means clustering, the buy signal accuracy did not change, and also was not proven significantly in the Mcnemar's Test with a high p-value of .2443. The sensitivity and specificity values also increased significantly however as mentioned, it is not statistically significant. What is interesting is the results from the sell list confusion matrix. The accuracy improved by almost a percentage point. At 50.1% it does technically perform better than a coinflip. The sensitivity and specificity improved ever so slightly.



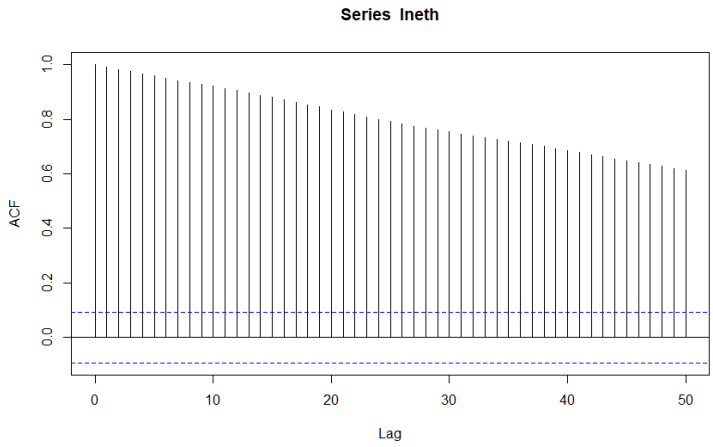


While the statistical tests don't really seem to prove tremendous improvement. I am hopeful different ensembles of methods could outperform a simple technical analysis strategy. Especially a Bayesian approach. In this use case the model did visually show a tremendous impact in minimizing loss on paper and decreased the variance overall, the enhanced method improved the return by nearly 76%. It would be interesting to see this applied to other financial assets or a longer time series given the relative influx in retail traders and popularity in cryptocurrency.

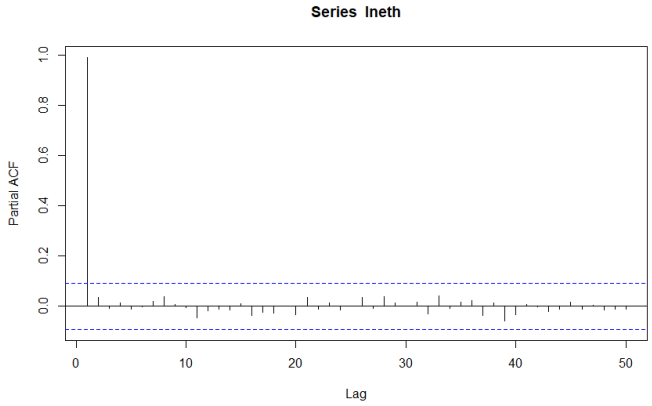
This may also prove very useful in application in portfolio management. The Sharpe Ratio developed by William Sharpe is basically the standard deviation of returns compared to a risk-free return. It views the assets return compared to excess return by the asset. In a portfolio of assets, it could work similarly to anomaly detection and automate the selling process of an asset that is under performing, or losing its edge. If an asset becomes riskier based on its Sharpe ratio, it can be sold sooner.

Forecasting

The second technique used in this analysis is an Autoregressive Integrated Moving Average (ARIMA). Since asset prices are considered random, or a stochastic random walk, this technique is used. In order to build an efficient and useful model, the data must be confirmed stationary. This process and be done in multiple ways. Common practice is to use an autocorrelation function as pictured in the diagram below.



When reviewing this function, it “tails off” showing that it descends gradually representing how correlated each lag is to the previous lag. The partial autocorrelation function (PACF) below “cuts off” or immediately after lag one there is zero correlation. This points in the direction of a AR(1) model and suggests that the data is stationary after a log transformation.



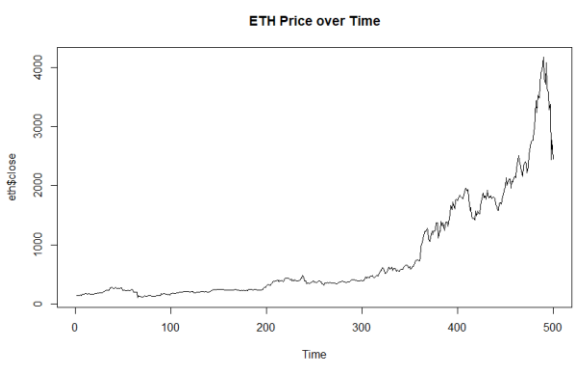
There are more dependable methods of checking for stationarity of the mean. One example is the Augmented Dicky-Fuller test. The ADF test was run on the original log transformation of the data, and then run on the log differenced data. The null hypothesis states that the time series is not stationary, the alternative hypothesis states that the data is stationary.

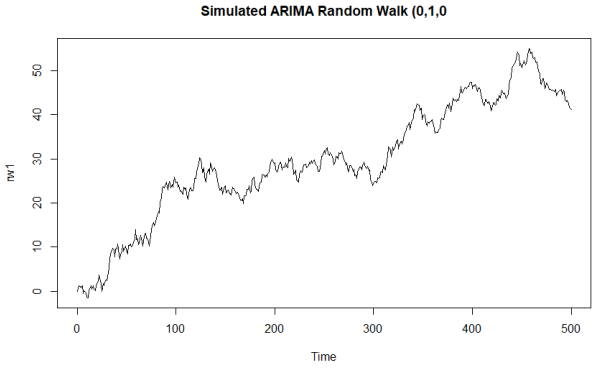
```
        Augmented Dickey-Fuller Test

data:  lneth
Dickey-Fuller = -1.9122, Lag order = 7, p-value = 0.6149
alternative hypothesis: stationary
```

The output shown above is from the ADF Test, the large p-value of 0.6149 indicates that it is not stationary and we cannot reject the null hypothesis.

```
        Augmented Dickey-Fuller Test

data:  difflneth
Dickey-Fuller = -8.1655, Lag order = 7, p-value = 0.01
alternative hypothesis: stationary
```

After taking the log difference and re-running the ADF Test, the low p-value of .01 indicates that we can reject the null hypothesis and accept the alternative that the time series is stationary.





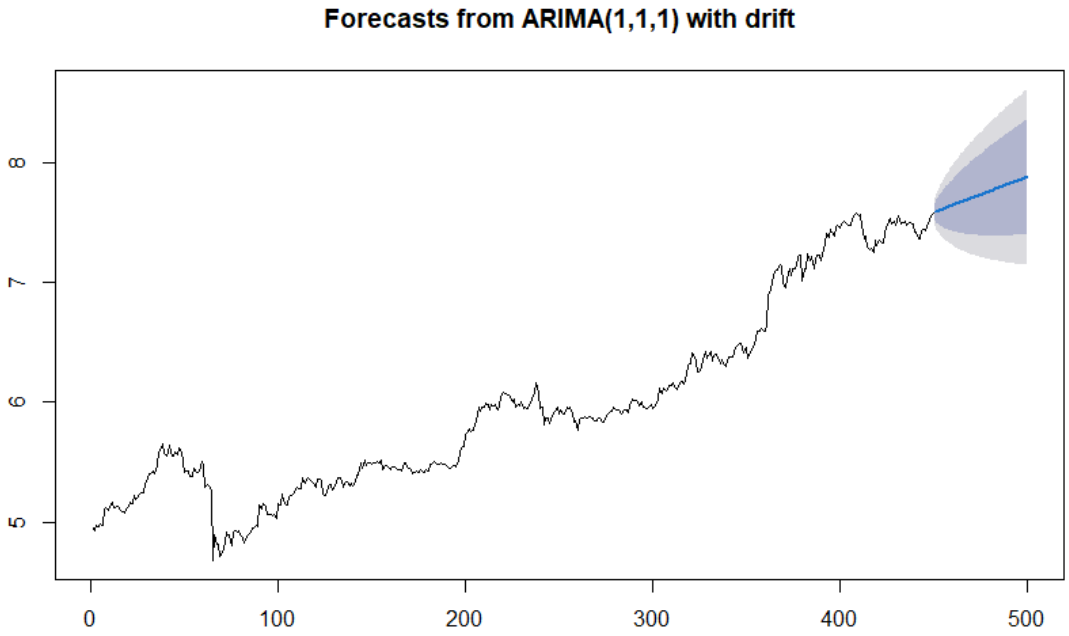
The justification for using the difference to achieve stationarity can be represented visually above. The plot on the right is a simulated ARIMA model of order (0,1,0). Or one difference, which negates the random walk. The Random Walk is based on the price yesterday and an error term epsilon that makes up the difference in price. To generate an ARIMA models p and d values for non-seasonal data (ARIMA(p,d,q) the Auto Arima function was used. The output was and ARIMA(1,1,1). Also known as a model with one lagged auto regression, one difference and the number of lagged forecast errors. I.e. its coefficient is weighted by “white noise.” Wold decomposition states that any stationary time series may be represented by a linear combination of white noise (David Stoffer, datacamp).

```
Series: eth.ARIMA
ARIMA(1,1,1) with drift

Coefficients:
          ar1     ma1   drift
      -0.8292  0.7248  0.0059
s.e.   0.0958  0.1174  0.0025

sigma^2 estimated as 0.003138:  log likelihood=658.4
AIC=-1308.79   AICc=-1308.7   BIC=-1292.37
```

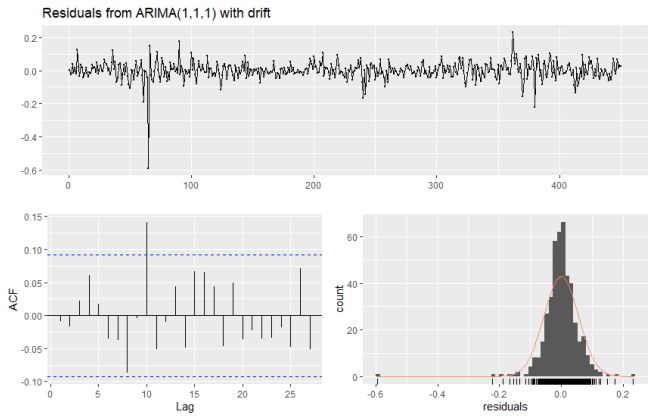
```
Call:
arima(x = eth.ARIMA, order = c(2, 1, 1))

Coefficients:
          ar1     ar2     ma1
      -0.8385  -0.0022  0.7372
s.e.   0.1498   0.0598  0.1421

sigma^2 estimated as 0.003156:  log likelihood = 655.61,  aic = -1303.22
```

```
Call:
arima(x = eth.ARIMA, order = c(1, 1, 2))

Coefficients:
          ar1     ma1      ma2
      -0.8354  0.7340  -0.0023
s.e.   0.1024  0.1126   0.0527

sigma^2 estimated as 0.003156:  log likelihood = 655.61,  aic = -1303.22
```

The other models both have an AIC lower than the ARIMA(1,1,1) model, but the forecast was nearly the same. The forecast was made with the ARIMA(1,1,1) as it was a simpler model and gave a useful prediction.



The above plot represents an ARIMA(1,1,1) forecast with a drift. The drift coefficient is the change of the mean of the random changes. The above plot is in term of the logarithm. The mean percent error when comparing the forecasted price to the actual price is .1415 (exponentiated to return to actual price and then the analysis was performed). The blue line is called the “point forecast” where the light gray and light blue are a visual representation of a confidence interval of future samples. The 80% and 95% interval.



The residuals shown on the right appear to be "good". The goal is to have random residuals as this means that the equation built for the ARIMA model describes the linear relationship that only random error remains. The plot of the residuals appears random or white noise, which is a good sign. And the ACF shows the majority of the residuals lying within the significance bounds. Showing low correlation threshold. The mean of the distribution appears to be zero, with some large negative outliers. This is not surprising as we see significant volatility in the asset price. However, the Ljung-Box test has a low p-value below an alpha level of .05, at .02456 we fail to reject the null hypothesis that the residuals do have autocorrelation. And we accept the alternative hypothesis that the data is not independently distributed.

The following is a comparison between a naïve forecast and a mean forecast.

```
> accuracy(naive_f, test)
                    ME      RMSE       MAE       MPE      MAPE     MASE       ACF1
Training set   4.629118  81.47298  35.02126 0.3886423  3.907711  1.00000 -0.2565207
Test set     329.120000 741.05689 551.15080 7.1109860 17.540964 15.73761         NA
> accuracy(mean_f, test)
                     ME      RMSE       MAE        MPE      MAPE     MASE
Training set -3.348877e-15  859.2823  677.1446 -133.30040 164.55871 1.000000
Test set      1.997771e+03 2105.2158 1997.7709   70.35924  70.35924 2.950287
```

The output shows the naïve forecast significantly outperforms the mean forecast as measurements, such as the MAPE and RMSE are much lower.

Conclusion:

In conclusion, this strategy uses the above simple moving average crossover strategy but is enhanced by k means clustering. The clusters capitalize on the percent change variability in the closing price of the stock and each respective moving average. A larger percent change may indicate a reversion to the mean. When the 7 day and the 14 day moving averages crossover and it belongs to a cluster in which the closing price where there is positive difference 7 day moving average and 14 day moving average (i.e. closing price is above moving average) it signals a purchase order. Inversely, if the EMAs vary greater percent wise negatively from the closing price it triggers the sell clause as we would expect a trend reversal. Using the WSS the minimal differences could start to cluster/recognize price movements and larger swings, as seen in the cluster output.

Ultimately it decreased volatility and helped lock in returns. In a margin account this could prove effective as when automated it can lock in returns. A triple EMA crossover could increase the complexity of the strategy which may lead to less and more accurate signals, as well as choosing different clusters to become actionable (As well as leaving some clusters out of actionable trades).

The forecasted models used did a decent job capturing the price trend. However, there is significant upward bias (which is partially removed by making the time series stationary) however in the forecast

the price shows a longer-term increasing trend. A model may be more useful if it includes a coefficient for drift relating to the volume or popularity of an asset. Other ARIMA orders may do a better job removing residuals, or using a larger time period may help smooth the auto-correlation. In the future it will be interesting to forecast using exponential moving averages and possibly integrate the two machine learning models employed in this project.

Works Cited

https://learn.datacamp.com/

Courses used: Forecasting in R, Visualizing Time Series Data in R, ARIMA Models in R

https://www.udemy.com/course/cryptocurrency-trading-using-machine-learning-with-r/learn/lecture/15909010#overview

Courses used: Bitcoin Trading using Machine Learning in R

https://www.stat.pitt.edu/stoffer/tsa4/tsaEZ.pdf

David Stoffer – Time Series: A data Analysis Approach Using R

https://rpubs.com/kevinTongam/arimaforecast

Rob J. Hyndman – Forecasting: Principles and Practice

Hastie, Tibshirani, Friedman – The Elements of Statistical Learning

https://www.youtube.com/user/joshstarmer

StatQuest with Josh Starmer – link to youtube channel - https://www.youtube.com/c/joshstarmer/about

Kahn Academy

Sentdex – link to youtube channel - https://www.youtube.com/results?search_query=sentdex+you

Appendix

```
################################################
# IST707, Standard Homework Heading
#
# Student name: Ryan Dean
# Final Project


#accessing libraries
library("Quandl")
library("tidyverse")
library("quantmod")
library("dplyr")
library("ggplot2")
library("xts")
library("quantmod")
library("factoextra")
#renaming dataset from csv generated
eth.ohlcv <- ethOHLCV052121
#removing index column
eth <- eth.ohlcv[,-c(1:2)]
#was running into issues with the acf and pacf plots, I think they were scaled to seconds
#had to use this approach
dates <- seq(as.Date("2020-01-07"), length = 500, by = "days")
#convert to time series
eth <- xts(x=eth, order.by = dates)
#checking parameters of dataset
length(eth)
head(eth, n = 10)
tail(eth, n = 10)
```

```
str(eth)
#extracting core
e_core <- coredata(eth)
class(e_core)
#index
e_index <- index(eth)
class(e_core)
#loading quantmod to plot ema
library(quantmod)
chartSeries(eth$close, theme = "black")
addEMA(7, col = "yellow")
addEMA(14, col = "pink")
#Feature/Signal Engineering
#assigning as time series
eth.ts <- data.frame(eth)
# exponential moving averages (1 week and 2 week)
ema7 <- EMA(eth.ts$close, n = 7)
ema14 <- EMA(eth.ts$close, n = 14)
## adding if/else logic columns
#ema 7 crosses above 14 buy
eth.ts$ema7.greater.ema14 <- ifelse(ema7 > ema14, 1, 0)
#ema  7 crosses below ema 14
eth.ts$ema7.less.ema14 <- ifelse(ema7 < ema14, 1, 0)
#close above ema 7
eth.ts$close.greater.ema7 <- ifelse(eth.ts$close > ema7, 1, 0)
#bull or bear candle
eth.ts$candle.today <- ifelse(eth.ts$close > eth.ts$open, 1, 0)
#bull or bear candle next day
eth.ts$candle.tommorrow <- lead(eth.ts$candle.today, n=1)
```

```
#percent change function
pct_delta <- function(x, y) {
  change <- (x - y)
  pct_change <- (change/y)
}
# 1 day return
eth.ts$daily_return <- pct_delta(eth.ts$close, eth.ts$open)
# 1 day lagged return
eth.ts$tommorrow_return <- lead(eth.ts$daily_return, n=1)
#ema close pct diff
eth.ts$close.ema7.diff <- pct_delta(eth.ts$close, ema7)
eth.ts$close.ema14.diff <- pct_delta(eth.ts$close, ema14)
#Omitting NAs
eth.ts <- na.omit(eth.ts)
#k-means cluster
set.seed(1234)
#elbow method for optimal clusters
fviz_nbclust(eth.ts[,12:13], kmeans, method = "wss")
#five clusters of the percent change difference between the close price and 7 day ema and 14 day ema
km1 <- kmeans(eth.ts[,12:13], 6)
#assigning a cluster to the column
eth.ts$cluster <- as.factor(km1$cluster)
#plotting clusters
par(mfrow=c(1,2))
ggplot(eth.ts, aes(close.ema7.diff, close.ema14.diff)) + geom_point()
cluster <- ggplot(eth.ts, aes(close.ema7.diff, close.ema14.diff, col = cluster)) + geom_point()
cluster + ggtitle("Percent Difference in EMA7 & EMA14")
#ETH behavior over past 486 days
par(mfrow=c(2,1))
```

```
eth.ts$eth.return <- cumsum(eth.ts$daily_return)

plot(eth.ts$daily_return, main="Daily Volatility - Buy and Hold", xlab="Volatility", ylab="Variance", type = "l", col = 5)

plot(eth.ts$eth.return, main="Cumulative Return - Buy and Hold", xlab="Days Since Position Entrance", ylab="Return", type = "l", col = 5)

# trading rules

rule <- 1


if (rule == 2) {



 #prediction w/ basic technical ema strat - long when price is above ema7

 eth.ts$buy_list <- ifelse(eth.ts$ema7.greater.ema14 == 1, 1, 0)

 eth.ts$sell_list <- ifelse(eth.ts$ema7.less.ema14 == 1, 1, 0)


 # predicted Return

 eth.ts$pred_return <- ifelse(eth.ts$candle.tommorrow == 1 & eth.ts$buy_list == 1,

                              eth.ts$tommorrow_return,

                              ifelse(eth.ts$candle.tommorrow == 0 & eth.ts$buy_list == 1,

                                   eth.ts$tommorrow_return,

                                   ifelse(eth.ts$candle.tommorrow == 1 & eth.ts$sell_list == 1,

                                        -eth.ts$tommorrow_return,

                                        ifelse(eth.ts$candle.tommorrow == 0 & eth.ts$sell_list == 0,

                                             -eth.ts$tommorrow_return, 0))))


 #cumulative return

 eth.ts$cumulative_return <- cumsum(eth.ts$pred_return)


} else { #long above ema7 when cluster has large differences in ema14, sell when negative changes
```

```
 #buy_list

 eth.ts$buy_list <- ifelse(eth.ts$ema7.greater.ema14 == 1 & eth.ts$cluster == 2 | eth.ts$cluster == 3 |
eth.ts$cluster == 5, 1, 0)

 #sell_list

 eth.ts$sell_list <- ifelse(eth.ts$ema7.less.ema14 == 1 & eth.ts$cluster == 1 | eth.ts$cluster == 4 |
eth.ts$cluster == 6, 1, 0)

 # predicted Return

 eth.ts$pred_return <- ifelse(eth.ts$candle.tommorrow == 1 & (eth.ts$cluster == 2 | eth.ts$cluster == 3
| eth.ts$cluster == 5) & eth.ts$buy_list == 1,

                      eth.ts$tommorrow_return,

                      ifelse(eth.ts$candle.tommorrow == 0 & (eth.ts$cluster == 2 | eth.ts$cluster == 3 |
eth.ts$cluster == 5) & eth.ts$buy_list == 1,

                          eth.ts$tommorrow_return,

                          ifelse(eth.ts$candle.tommorrow == 1 & (eth.ts$cluster == 1 | eth.ts$cluster == 4 |
eth.ts$cluster == 6) & eth.ts$sell_list == 1,

                              -eth.ts$tommorrow_return,

                              ifelse(eth.ts$candle.tommorrow == 0 & (eth.ts$cluster == 1 | eth.ts$cluster ==
4 | eth.ts$cluster == 6) & eth.ts$sell_list == 0,

                                  -eth.ts$tommorrow_return, 0))))


 #cumulative return

 eth.ts$cumulative_return <- cumsum(eth.ts$pred_return)


}

#plots of returns and volatility

par(mfrow=c(2,1))

plot(eth.ts$pred_return, type = "l", col = 5)

title("Volatility")

plot(eth.ts$cumulative_return, type = "l", col = 5)

title("Cumulative Return")
```

```
#Confusion Matrix and Accuracy Measures - Using Caret

#confusion matrix of buying accuracy

buy_list_cm <- confusionMatrix(factor(eth.ts$candle.tommorrow), factor(eth.ts$buy_list))

#adding a bear candle column with similar logic as to not have to change logic above

#bear candle

eth.ts$candle.todayinv <- ifelse(eth.ts$close < eth.ts$open, 1, 0)

#bull or bear candle next day

eth.ts$candle.tommorrowinv <- lead(eth.ts$candle.today, n=1)

#confusion matrix of selling accuracy - basically saying (If bear candle tomorrow and predicted to sell today, what is the accuracy)

sell_list_cm <- confusionMatrix(factor(eth.ts$candle.tommorrowinv), factor(eth.ts$sell_list))

#output confusion matrix for

buy_list_cm

sell_list_cm

#cumulative return for a strategy

eth.ts$cumulative_return

library(MASS)

library(tseries)

library(forecast)

#Renaming dataset from csv

eth.ohlcv <- ethOHLCV052121

#removing index column

eth <- eth.ohlcv[,-c(1:2)]

#was running into issues with the acf and pacf plots, I think they were scaled to seconds

#had to use this approach as the crypto market technically never "closes". These time periods are 24 hours apart.

dates <- seq(as.Date("2020-01-07"), length = 500, by = "days")

#convert to time series

eth <- xts(x=eth, order.by = dates)
```

```
class(eth)
#natural log of eth price (log transformation)
lneth <- log(eth$close[1:450])
lneth
#ACF, PACF, and Dickey-Fuller Test
acf(lneth, lag.max = 50)
#gradual decent of correlation
pacf(lneth, lag.max = 50)
#Simulated random walkto justify using one difference
set.seed(12345)
rw1 <- arima.sim(model = list(order = c(0,1,0)), n = 500)
ts.plot(rw1, main = "Simulated ARIMA Random Walk (0,1,0")
rw_diff <- diff(rw1, n = 1)
ts.plot(rw_diff)
ts.plot(eth$close, main = "ETH Price over Time")
#immediate drop in correlation at lag 0 to 1, it is stationary
#differencing because of PACF plot
difflneth <- diff(lneth, 1)
difflneth
#adf test on logged and logged differenced series
lneth <- na.omit(lneth)
difflneth <- na.omit(difflneth)
ln <- adf.test(lneth)
ln
#reject null hypothesis of non-stationarity
diff <- adf.test(difflneth)
diff
#Time Series and auto-arima
eth.ARIMA <- ts(lneth)
```

```
fitlneth <- auto.arima(eth.ARIMA)
fitlneth
plot(eth.ARIMA, type="l", main = "ETH Log Price Returns")
exp(lneth)
#other models to compare
model2 <- arima(eth.ARIMA, order=c(2,1,1))
summary(model2)
#model (1,1,2)
model3 <- arima(eth.ARIMA, order=c(1,1,2))
summary(model3)
#forecasted values from ARIMA
forecastedvalues <- forecast(fitlneth, h=50)
forecastedvalues
plot(forecastedvalues)

forecastedvalues <- as.numeric(forecastedvalues$mean)
finalforecast <- exp(forecastedvalues)
finalforecast

#Percentage Error
df <- data.frame(eth$close[451:500], finalforecast)
headings <- c("actual price", "forecasted price")
names(df) <- headings
attach(df)
percent_error <- ((df$'actual price'-df$'forecasted price')/(df$'actual price'))
percent_error
mean(percent_error)
view(df)
#make sure residuals are random
```

```
checkresiduals(fitlneth)

#naive forecast
neth <- naive(lneth, h=50)
autoplot(neth)

#training and test data sets for different forecasting methods
eth_fc <- eth$close
train <- subset(eth_fc, end = 450)
test <- eth_fc[451:500]
naive_f <- naive(train, h = 50)
mean_f <- meanf(train, h = 50)

accuracy(naive_f, test)
accuracy(mean_f, test)

#time series cross validation - forecasting evaluation on a rolling origin lol
error <- tsCV(train, forecastfunction = naive, h = 10)
mse <- colMeans(error^2, na.rm = TRUE)
#taken directly from datacamp - Forecasting in R
data.frame(h=1:10, MSE = mse) %>% ggplot(aes(x=h, y=MSE)) + geom_point()
```