MSc Research Paper

Course: WQU Capstone-0218-A

Name: Emeagi Michael

Title: A Holistic Comparison of High and Low Frequency Trading

WorldQuant University

Supervisor: Professor Ritabrata Bhattacharyya

Key Words: Comparison, Frequency, High, Low, Trading.

Abstract

This research will try to use a simple strategy to compare High and Low frequency trading. The strategy will use the 10 and 30 simple moving averages and the Random Forest algorithm to improve the prediction power of the trading system. The strategy will be used to trade these assets – ETF Gold, Apple, Eurusd starting January, 2000 to January, 2018. For each asset, there will be two datasets with different time timeframes. The first dataset will be in daily timeframe and will be used for the low frequency strategy and the second in a minute time frame, and this would be used for the high frequency strategy. To obtain high frequency data without much difficulty, the daily datasets would be resampled into the one minute time frame. Metrics such as Total Profit earned during the backtest period, Cumulative Average Growth Rate, Sharpe Ratio and Maximum Drawdown. These metric would compared for high and low frequency trading with same strategy to see the dataset with the best return and minimal risk over the tested period.

Table of Content

Chapter One

Introduction

The Financial markets with fully electronic execution and similar electronic communication networks developed in the late 1980 and 1990. In the U.S., decimalization, which changed the minimum tick size from 1/16 of a dollar (US$0.0625) to US$0.01 per share, may have encouraged algorithmic trading as it changed the market microstructure by permitting smaller differences between the bid and offer prices, decreasing the market makers' trading advantage, thus increasing market liquidity. In 1998, the US Securities and Exchange Commission (SEC) authorized electronic exchanges paving the way for computerized High Frequency Trading. HFT was able to execute trades 1000 times faster than a human. And since that time high frequency trading (HFT) has become widespread.

However, the rise of HFT has provoked a range of reactions from highly supportive to highly negative, with a few holding a neutral view. Before reaching any conclusions, market observers and participants should consider the findings of empirical, qualitative research supported by grounded and knowledgeable qualitative insight. On the other hand, low frequency trading referred to buy and hold and also trading strategies that worked with datasets that come in larger time frames like daily, weekly and month. This research paper will through data collection and simulation compare these two trading approaches or strategies by computing important metrics like profitability, Sharpe ratio, and cumulative growth average rate.

This research will be organised as follows: Chapter 2 presents how datasets would be collected for analysis. Chapter 3 describes how the collected data would processed and the trading strategy applied. Chapter 4 presents and interprets the result. Chapter 5 draws conclusion and make recommendations for future research.

Chapter Two

Data Collection

Assets for this project are chosen in way to cover at least three varieties of asset classes – commodities, equities and currencies. The assets are:

1. Gold ETF with ticker symbol 'SPY"
2. Apple stock with ticker symbol 'AAPL
3. Euro versus Dollar with ticker symbol 'EURUSD

Python programming language was used for all simulations. Datasets from these assets were classified into two groups-high frequency data and low frequency data.

2.1 High Frequency Data

High frequency data refers to any time series data with updates that occur in less than fractions of a second. Oftentimes the term is applied to market data from stock or financial products. It can be related to stock prices which are today recorded at seconds or less frequency and investors are creating tools which determine the prediction of next movement and flow of such data to decide whether to invest in stock A vs. Stock B in Market X vs. Market Y.

For simplicity, this project will regard one minute historical prices of the chosen assets as the high frequency data. Minute historical prices could be downloaded from sites that render financial services information like Yahoo and Google Finance. But in this project, resampling technique was applied to daily historical prices to obtain one minute data prices. The python code for downloading the daily historical prices before applying resampling technique is given below:

```
import pandas_datareader.data as web
df= web.DataReader('AAPL', 'yahoo','2010-01-01')
```

Line one imports a sub package in pandas that will allow line two download the historical prices of Apple stock from 2010-01-01 till date from Yahoo.

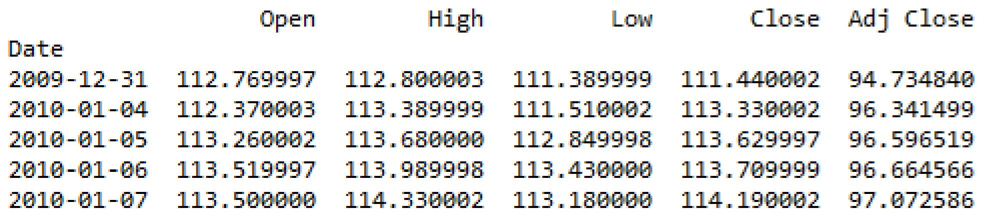
2.2 Resampling Technique

Resampling describes the process of frequency conversion over time series data. It is a helpful technique in various circumstances as it fosters understanding by grouping together and aggregating data. It is possible to create a new time series from daily price data that shows the weekly or monthly close prices. On the other hand, real-world data may not be taken in uniform intervals and it is required to map observations into uniform intervals or to fill in missing values for certain points in time. These are two of the main use directions of resampling: binning and aggregation, and filling in missing data. Downsampling and upsampling occur in other fields as well, such as digital signal processing. The python code used for downsampling the daily historical prices is shown below.

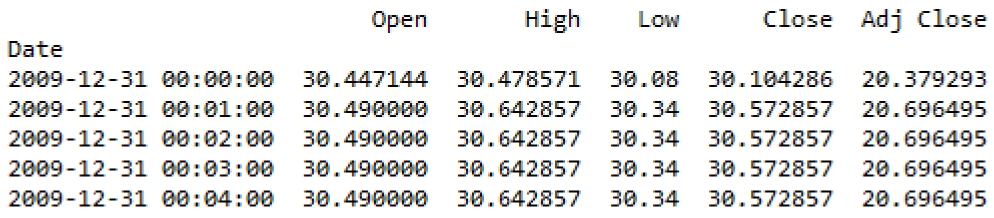
data = df.resample('T').bfill()

The ***bfill ()*** means backward fill and it means it wants to use previous values to replace nan values.

A sample header of daily historical price values in pandas dataframe is shown and also a dataframe when the data has been resampled.



```
                  Open        High         Low       Close   Adj Close
Date
2009-12-31  112.769997  112.800003  111.389999  111.440002  94.734840
2010-01-04  112.370003  113.389999  111.510002  113.330002  96.341499
2010-01-05  113.260002  113.680000  112.849998  113.629997  96.596519
2010-01-06  113.519997  113.989998  113.430000  113.709999  96.664566
2010-01-07  113.500000  114.330002  113.180000  114.190002  97.072586
```



```
                          Open       High    Low      Close  Adj Close
Date
2009-12-31 00:00:00  30.447144  30.478571  30.08  30.104286  20.379293
2009-12-31 00:01:00  30.490000  30.642857  30.34  30.572857  20.696495
2009-12-31 00:02:00  30.490000  30.642857  30.34  30.572857  20.696495
2009-12-31 00:03:00  30.490000  30.642857  30.34  30.572857  20.696495
2009-12-31 00:04:00  30.490000  30.642857  30.34  30.572857  20.696495
```

The first table show daily historical prices and second is a resampled prices in minutes. The time stamp clearly shows the time in minutes.

2.3 Low Frequency Data

Low frequency data uses daily, weekly, monthly or even less frequent data. Low frequency trading ignores market microstructure and bases strategies solely on trends and charts. Sourcing for low frequency data is not as challenging as high frequency data. For the low frequency strategy, the data were just used for the strategy just the way they were obtained from the web with the pandas. The difference with the high frequency data was that the daily historical prices were not resampled.

Chapter Three

Data Processing

The historical prices for high and low frequency data are already in a pandas data. This dataframe is further built on by adding more columns as required by the strategy. Columns with the moving average prices, Random Forest generated signals and other necessary columns required for implementing the strategy.

3.1 Simple Moving Average

Simple Moving Average (SMA) is an arithmetic moving averages calculated by adding the closing prices of the securities for a number of time periods and then dividing this total by the number of time periods. Many traders watch for short-term averages to cross above the long-term averages to signal the beginning of an uptrend. Short-term averages can act as levels of support when the price experiences a pullback.

A simple moving average is customizable in that it can be calculated for a different number of time periods, simply by adding the closing prices of the security for a number time periods and then dividing this total by the number of time periods, which gives the average prices of the security over the time period. A simple moving average smoothes out volatility, and makes it easier to view the price trend of a security. If it is pointing down, it means that the security price is decreasing. The longer the timeframe for the moving averages, the smoother the simple moving average. A short-term moving average is more volatile, but its reading is closer to the source data.

Moving averages are an important analytical tool used to identify current price trends and the potential for a change in an established trend. The simplest form of using a simple moving average in analysis is using it to quickly identify if a security is in an uptrend or downtrend. Another popular, albeit slightly more complex analytical tool, is to compare a pair of simple moving averages with each covering different time frames. If a shorter-term simple moving average is above a longer-term average, an uptrend is expected. On the other hand, a long-term above a shorter-term average signals a downward movement in the trend.

3.2 Slow Moving Average

The strategy for this project will make use of two moving averages. The moving average with a period of 30 would be considered as the slow moving average. It is regarded as the slow moving average because it was calculated with a longer period than the fast moving average. A column was created for the 30 period moving averages by adding another column with column name “ma_30” to the original pandas dataframe that has the Open, High, Low, Close historical prices. The python code for generating series of these moving averages is given below.

data['ma_30'] = data.Close.rolling(window=30).mean()

From the code above, the **data[‘ma_30’]** is creating a column with name ‘ma_30’ to the original dataframe with asset historical prices with name **‘data’**. The moving averages were created with the Close prices and ‘window = 30’ shows that it a 30 day/period moving averages.

3.3 Fast Moving Average

This moving average is the same with the slow except that it uses a period of 10. The averages are generated with the column name “ma_10” and added to the main pandas dataframe. The python code for generating the fast moving averages is given as

data['ma_10'] = data.Close.rolling(window=10).mean()

This is code is the same as the is the same for the slow moving averages except that the period was changed to 10 and column name for the averages changed to ‘ma_10’

3.4 Trading Strategy

A Trading Strategy is a fixed plan that is designed to achieve a profitable return by going long or short in the market. The main reasons that a properly researched trading strategy helps are its verifiability, consistency, and objectivity. For every trading strategy one needs to define assets to trade, entry/exit points and money management rules.

The strategy for this project will make a trading decision with the following:

(a) Slow and Fast moving averages with periods of 30 and 10 respectively.

(b) Random Forest Algorithm for higher signal prediction accuracy.

3.5 Signal Generation

A signal column is created and added to the original pandas dataframe. The signal generator assigns a 1(which indicates a buy signal) when the fast simple moving average crosses above the slow moving average and a -1 when the it crosses below the slow moving average. A 0 is assigned when non of these conditions are met. The python code that achieved this is shown below:

data['signal'] = np.where((data['ma_10']>data['ma_30']) , 1, 0)
data['signal2'] = np.where((data['ma_10']<data['ma_30']) , -1,data['signal'])

np.where is a conditional code in numpy that assigns values to some set conditions. This could be used to backtest the strategy but in order to enhance the signal accuracy, the random forest algorithm is used to generate another column of signals which would be used in the actual backtest.

3.6 Random Forest Algorithm

Random Forest is a machine learning technique. It is an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for the decision trees' habit for overfitting to their training set.

In this project, the Random Forest algorithm would be used as a classifier. It will classify the decisions into 1,-1 and 0. Like all machine learning prediction algorithm, the Random Forest generates signals with the features of the dataframe listed below

1. 10 period simple moving average
2. 30 period simple moving average
3. Open
4. High
5. Close

6. Low

To predict signals, the data is split in two: the Training Set and Test Set in the ratio of 75% to 25% respectively. Thus, 75% of the data is trained with the Random Forest algorithm while the remaining 25% is used to test. The python code responsible for this function is shown below.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
y = data['signal2']
X=data[['ma_30','ma_10','Close','Volume','High','Low']]
# Split data into Training and Test sets
split_percentage = 0.75
split = int(split_percentage*len(data))
# Train data set
X_train = X[:split]
y_train = y[:split]
# Test data set
X_test = X[split:]
y_test = y[split:]
clf_random_f = RandomForestClassifier(n_jobs=2, random_state=0)
clf_random_f.fit(X_train, y_train)
accuracy_train_rf = accuracy_score(y_train, clf_random_f.predict(X_train))
accuracy_test_rf = accuracy_score(y_test, clf_random_f.predict(X_test))
print('\nTrain Accuracy:{: .2f}%'.format(accuracy_train_rf*100))
print('Test Accuracy:{: .2f}%'.format(accuracy_test_rf*100))
data['Pred_Sig_Random_F'] = clf_random_f.predict(X)
data['Strategy_Return_RF'] = data.Return * data.Pred_Sig_Random_F
```

There comment section of the code with a light dark font gave some explanation of what that section of the code is doing.

3.7 Backtesting

Backtesting is the process of testing a strategy on a relevant historical data to ensure its viability before the trader risks any actual capital. A trader can simulate the trading of a strategy over an appropriate period of time and analyze the results for different levels of profitability and risk. If the results meet the necessary criteria that are acceptable to the trader, the strategy can then be implemented with some degree of confidence that it will result in profits. If the results are less favourable, the strategy can be modified, adjusted and optimized to achieve the desired results, or it can be completely scrapped.

When done correctly, backtesting can be an invaluable tool for making decision on whether to utilize a trading strategy. The sample time period on which a backtest is performed is critical. The duration of the sample time period should be long enough to include periods of varying market conditions including uptrends, downtrends and range-bound trading. Performing a test on only one type of market condition may yield unique results that may not function well in other market conditions, which may lead to false conclusions. The sample size in the number of trades in the test results is also crucial. If the sample number of trades is too small, the test may not be statistically significant. A sample with too many trades over a long period may produce optimized results in which an overwhelming number of winning trades coalesce around a specific market condition or trend that is favourable for the strategy. This may also cause a trader to draw a misleading conclusion.

A backtest should reflect reality to the best extent possible. The trading cost may otherwise be considered to be negligible by traders when analyzed individually but may have significant impact when the aggregate cost is calculated over the entire backtesting period. The costs include commissions, spreads and slippage, and they could determine the difference between weather a trading strategy is profitable or not. Perhaps, the most important metric associated with backtesting is the strategy's level of robustness. This is accomplished by comparing the results of an optimized backtest in a specific sample time period   with the results of a backtest with the same strategy and settings in a different sample time period. If the results are similarly profitable, then the strategy can be deemed to be valid and robust, and it is ready to be implemented in time markets. If the strategy fails in out-of-sample comparisons, then the strategy needs further development, or it should be abandoned altogether.

This project was backtested with a virtual money of $100,000 for both the high and low frequency simulations. Three percent commission was factored for every single trade with zero slippage assumed for all trades. The backtest was run on the entire historical data downloaded. For the Gold ETF and Apple stock, the period of backtest started from 2010 till date. For EURUSD, period used for backtesting started from 01/01/2013 and ended in 02/07/2018. A new dataframe with the name “portfolio” was created for the purpose of analyzing the backtest results. The backtest code for the high and low frequency simulations are shown below with detailed comments.

```
#Backtesting
# Set the initial capital
initial_capital= float(100000.0)

# Create a DataFrame `positions`
positions = pd.DataFrame(index=data.index).fillna(0.0)

# Buy a 100 units of financial instrument
positions['Stock'] = 100*data['signal2']
# Initialize the portfolio with value owned
portfolio = positions.multiply(data['close'], axis=0)

# Store the difference in shares owned
pos_diff = positions.diff()

# Add `holdings` to portfolio
portfolio['holdings'] = (positions.multiply(data['close'], axis=0)).sum(axis=1)
```

```
# Add `cash` to portfolio and subtract 3% commission per trade
portfolio['cash'] = initial_capital - (pos_diff.multiply(data['close'], axis=0)).sum(axis=1).cumsum() - 0.03*(pos_diff.multiply(data['close'], axis=0)).sum(axis=1).cumsum()

# Add `total` to portfolio
portfolio['total'] = portfolio['cash'] + portfolio['holdings']

# Add `returns` to portfolio
portfolio['returns'] = portfolio['total'].pct_change()

# Print the last lines of `portfolio`
print 'Total money returned by the LFT backtest is ',portfolio.iloc[-1]['total']
print portfolio
#calcualte CAGR
cagr = portfolio.iloc[-1]['total']/initial_capital
print 'CAGR is ', cagr

# calculate Mean Return
mean_ret = np.mean(portfolio['returns'])
print 'Mean Return is ', mean_ret

#Calculate Gain to Pain Ratio
gpr = np.sum(portfolio['returns'])/sum(n < 0 for n in portfolio['returns'])
print 'Gain to Pain Ratio is ',gpr
# Plotting Portfolio Returns and identifying peak periods showing drawdown
i = np.argmax(np.maximum.accumulate(portfolio['total']) - portfolio['total']) # end of the period
j = np.argmax(portfolio['total'][:i]) # start of period

plt.plot(portfolio['total'])
plt.plot([i, j], [portfolio['total'][i], portfolio['total'][j]], 'o', color='Red', markersize=10)
```

Chapter Four

Presentation of Result

In this section, the performance of the high and low frequency data set would be analyzed by observing the metrics explained in the sections below.

4.1. Sharpe Ratio

Sharpe Ratio is the measure of risk-adjusted return of a financial portfolio. A portfolio with a higher Sharpe Ratio is considered superior relative to its peers. It is a measure of excess portfolio return over the risk free rate relative to its standard deviation. Normally, the 90-day Treasury bill rate is taken as the proxy for risk free rate.

The formula for calculating the Sharpe ratio is $[R(p) – R(f)]/s(p)$

Where

$R(p)$: Portfolio return

$R(f)$: Risk free rate of return

$S(p)$: Standard deviation of the portfolio

4.2 Compound Annual Growth Rate (CAGR)

This metric is a business and investing specific term for the geometric progression ratio that provides a constant rate of return over the time period. CAGR is not an accounting term, but it is often used to describe some element of the business, for example revenue, units delivered, registered users, etc. It dampens the effect of volatility of periodic returns that can render arithmetic means irrelevant. It is particularly useful to compare growth rates from various from various data sets of common such as revenue growth of companies in the same industry.

The formula for CAGR is

$\text{CAGR} = (EV/BV)^{1/n} – 1$

Where:

EV = Investment’s ending value

BV = Investment’s beginning value

n = Number of periods (months, years, etc)

4.3 Gain to Pain Ratio

The Gain to Pain Ratio is calculated by dividing the sum of periodic portfolio gains by the absolute value of monthly portfolio losses. A Gain to Pain Ratio of 1 or better is quite good, a ratio of 2 or better is very good and a ratio of 3 or better is excellent.

4.4 Mean Return

Mean Return is the expected value or mean of all likely returns of investments comprising of a portfolio. It attempts to quantify the relationship between the risk of a portfolio of securities and its return. It assumes that while investors have different risk tolerances, rational investors will always seek the maximum rate of return for every level of acceptable risk tolerance. It is the mean or expected return that the investors try to maximize at each level of risk.

4.5 Total Amount Returned by Portfolio at the End of Backtest

This is just the total dollar value returned at the end of the backtest. This value will say whether the strategy is profitable or not at the end of the backtest period.

4.6 Graphical Presentations of Returns Showing Drawdowns

A graphical representation will provide a deep insight of what the portfolio returns look like clearly showing drawdown for risk evaluation.

4.7 Performance Metrics of HFT and LFT Simulations

The matrix below clearly shows the performance metrics for the HFT and LFT simulations.

	Low Frequency Data Set			**High Frequency Data Set**		
Assets / **Metric**	**Gold ETF**	**Apple Stock**	**EURUSD**	**Gold ETF**	**Apple Stock**	**EURUSD**
Sharpe Ratio	-3.56	-5.4	-656.31	-54.09	-93.37	-1268.84
CAGR	-2.23%	0.42%	0.02%	2.90%	1.80%	0.02%
Gain to Pain Ratio	0.000058	0.0000677	0.000000311	0.000202	0.000144	8.43E-09
Mean Return	-0.000029	0.0000322	- 0.000000133	4.42E-08	3.33E-08	4.99E-11
Total Amount Returned by Strategy	$83,461.76	$103,397.92	$100,110.45	$125,231.44	$116,163.24	$100,110.30

Figure 1 Performance Metrics Tables for High and Low Frequency Trading Simulations

In the table above, the same financial instruments in both the Low and High Frequency Data Sets have identical colours for a clear visual comparison. The strategy did not fare well with both data sets, however, Low frequency data set had a better Sharpe ratio. The CAGR for all assets under high frequency data set were better than their low frequency data sets. Thus, the assets showed growth even though it was in small amount. Gain to Pain Ratio doesn’t look good for the simulation of both data sets but the high frequency data set appeared to have better values with the exception of EURUSD. Low frequency data set has an edge in this metric which was expected since the trades lasted longer because of the daily period. Finally, the high frequency data set returned more cash at the end of backtest period with all assets in the category(high frequency data sets) outperforming the same assets in the low frequency data set category.

4.8 Low Frequency Data Simulations Showing Returns and Maximum Drawdowns

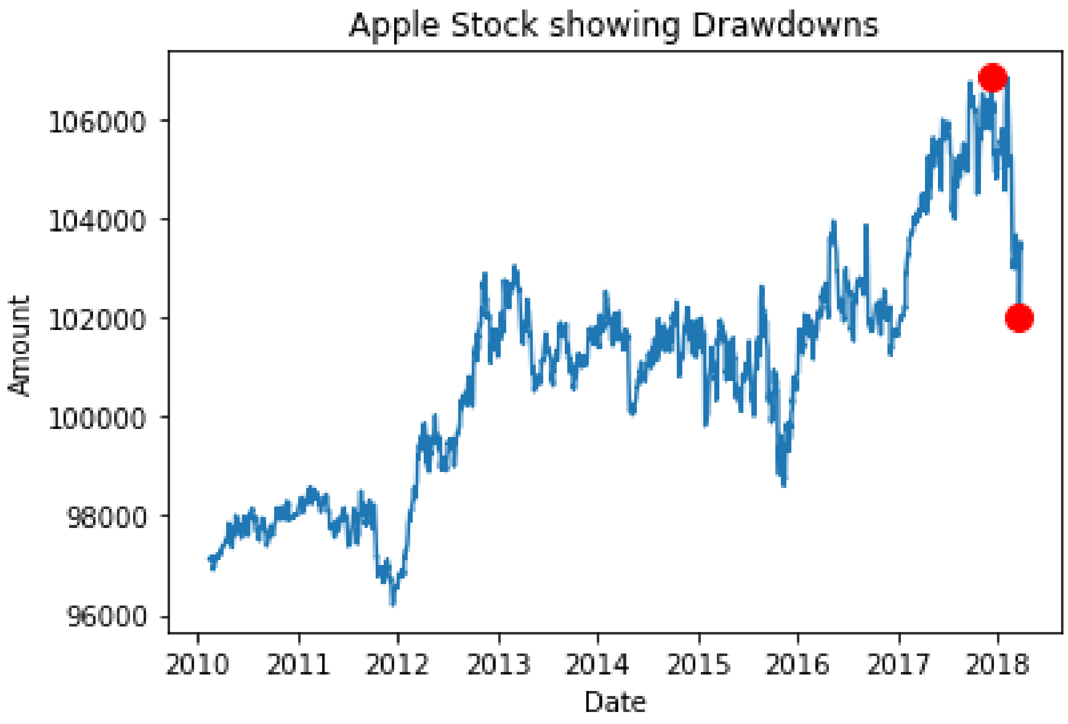


Figure 2 Low Frequency Data – Apple Returns Showing Maximum Drawdown

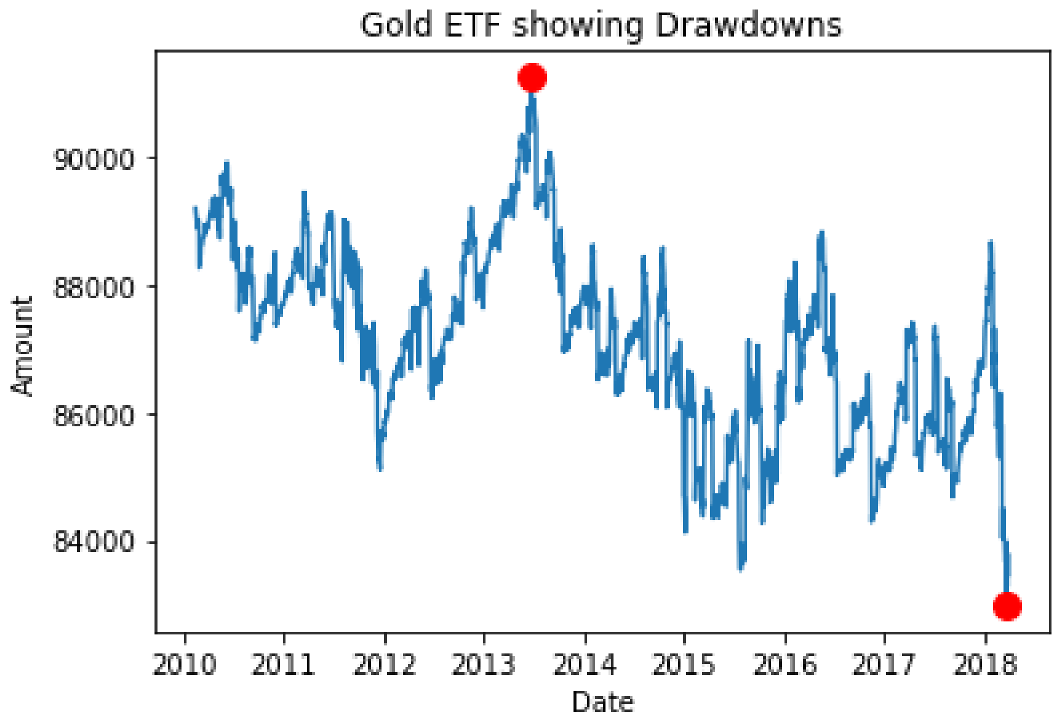


Figure 3 Low Frequency Data – Gold ETF Returns Showing Maximum Drawdown

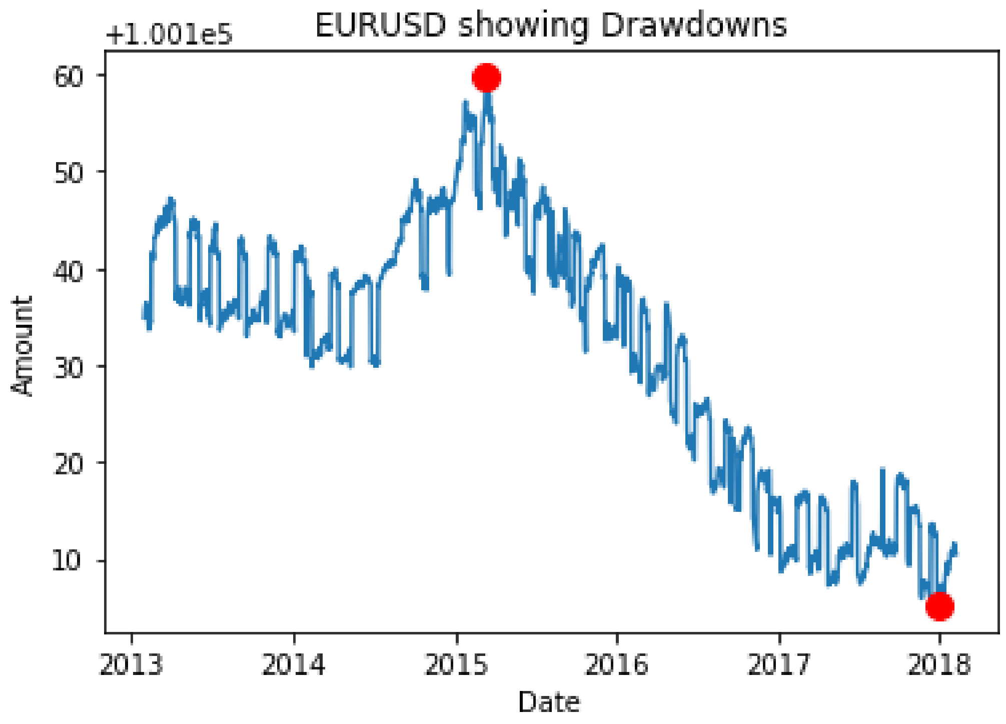


Figure 4 Low Frequency Data – EURUSD Returns Showing Maximum Drawdown

4.9 High Frequency Data Simulations Showing Asset Returns and Maximum Drawdowns

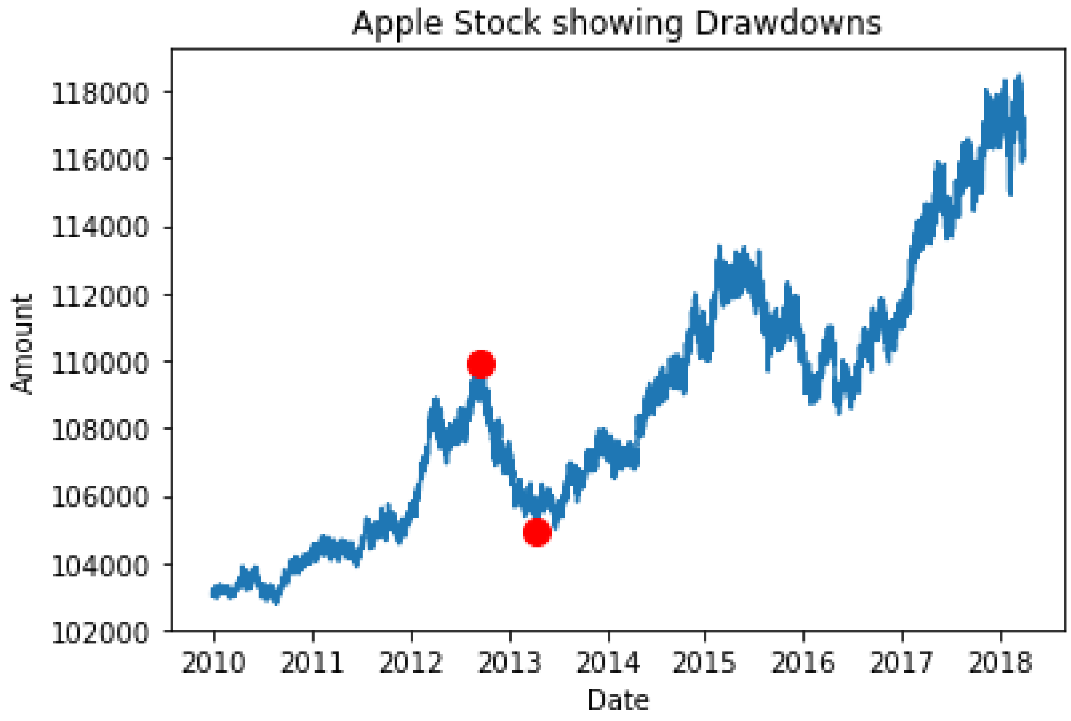


Figure 5 High Frequency Data – Apple Returns Showing Maximum Drawdown

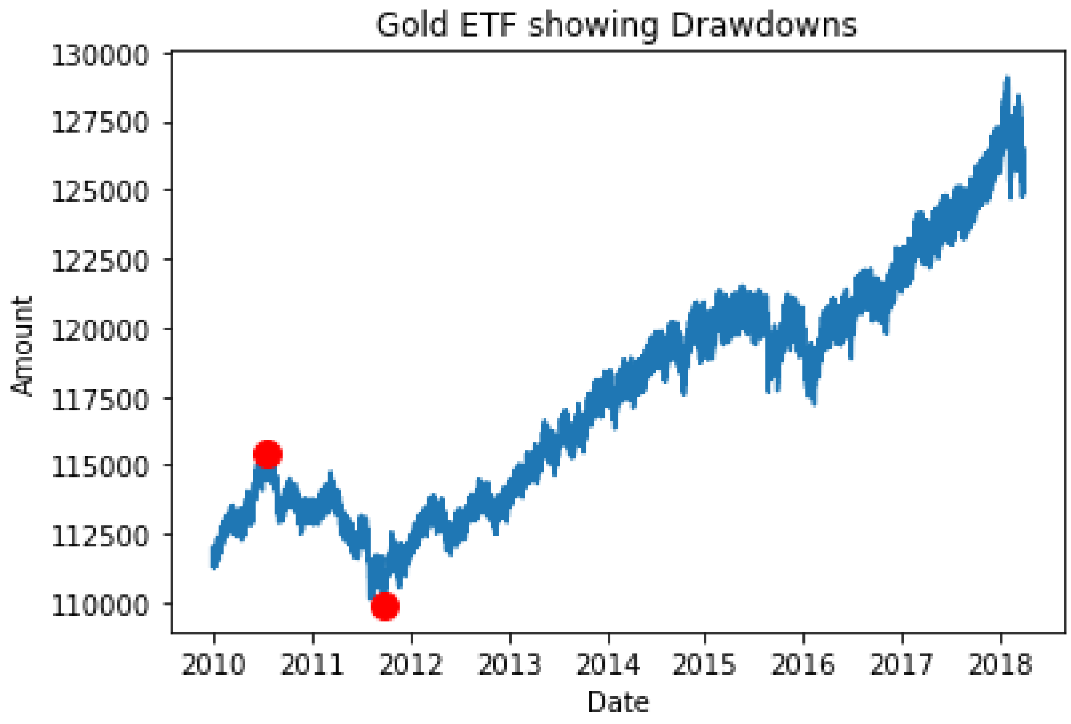


Figure 6 High Frequency Data – Gold ETF Returns Showing Maximum Drawdown

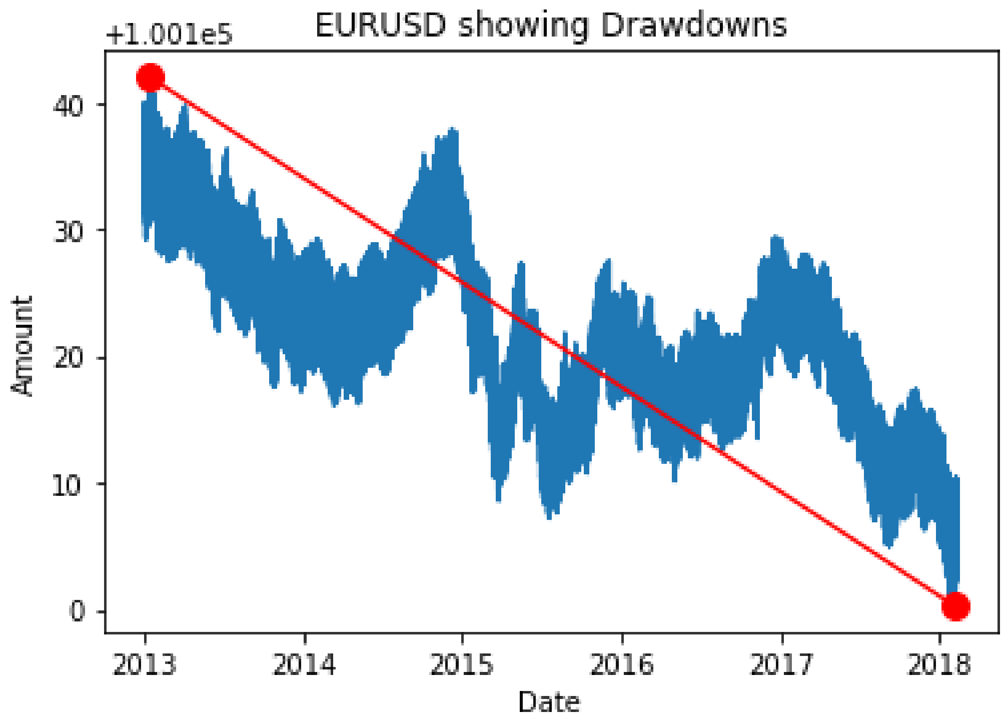


Figure 7 High Frequency Data – EURUSD Returns Showing Maximum Drawdown

Chapter Five

Conclusion

The results for both simulations were presented in Chapter four with some key metrics computed to access the performance of the high and low frequency data set. Sharpe ratios for both data sets were bad with the low frequency data set a lot better when compared with the high frequency. For Compound Annual Growth Rate (CAGR), the high frequency data set showed slight growth even though it was very small. On the same metric, the low frequency data set performed poorly with Gold ETF being the worst. Gold ETF experienced a negative growth rate over the entire 8 years period the backtest was run. Gain to Pain Ratio did not take a different direction when compared to other metrics measured. High frequency data set also outperformed the low frequency data in this area with the exception EURUSD. The most important metric measured during the trading simulations was the profitability of the strategy on both data sets. Both high and low frequency data sets returned profit for each asset during the entire backtest with the exception of Gold ETF under low frequency data set which returned a loss. On a more holistic comparison, the high frequency data sets were more profitable than the low frequency data sets with the same strategy.

The strategy chosen to test the performance of these two groups of data set was very simple and lacked complexity. The sole aim of this project was to know which style of trading would provide better returns over the same period of time despite the fact so many assumptions were made. The high frequency trading appeared to be the most profitable on almost all fronts.

High frequency trading is gaining a lot of popularity and developing rapidly with a lot of quantitative trading firms investing fortunes on high frequency trading technologies in order to increase portfolio returns. This project could form a platform for developing very profitable quantitative trading strategy by applying latest cutting edge technology in the industry.

Reference

Sam, Ifidon, E., Elizabeth, Ifidon, I., (2007). *Basic principles of research methods*. Benin City, Edo: Goodnews Express Communications.

History of Algorithmic Trading, HFT and News Based Trading.(2015, June 2). Retrieved from https://www.quantinsti.com/blog/history-algorithmic-trading-hft/

Evans, T.(2015, November 5). How do financial advisors execute trades? Retrieved from https://www.investopedia.com/ask/answers/110515/how-do-financial-advisors-execute-trades.asp

Backtesting a Moving Average Crossover in Python with Pandas.(2014, January 21). Retrieved from https://www.quantstart.com/articles/Backtesting-a-Moving-Average-Crossover-in-Python-with-pandas

Resampling Time Series Data with Pandas.(2016, May 23). Retrieved from https://www.benalexkeen.com/resampling-time-series-data-with-pandas/

Momentum Based Strategies for Low and High Frequency Trading.(2015, November 30). Retrieved from https://www.quantinsti.com/blog/momentum-based-strategies-for-low-and-high-frequency-trading/