Tesla Volatility Trading Strategy

Yu-Wei Vincent Yeh
University of California, Los Angeles
vincentywyeh@gmail.com

Abstract

The aim of this project is to devise a systematic trading strategy for Tesla that promises low-risk and sustainable return by observing the relationship between Tesla's implied volatility and historical volatility. This project consists of two stages:

1. Stage one: Technical analysis techniques were utilized to analyze Tesla's daily charts from 01/01/18 through 09/25/20 — the main strategy was devised at this stage.
2. Stage two: Previously devised strategy was implemented in Python using QuantConnect's LEAN Engine, an open-source algorithmic trading software. At this stage, trades were simulated on a much more frequent scale and in a less rigorous manner — this will be explained thoroughly in section II part C. Note that this algorithm should only be considered as a rough demonstration of the strategy devised in stage one.

Motivation

In recent years with Tesla's media exposure skyrocketed, its stock began to gain extreme volatility, creating enticing yet risky investment opportunities. However, what came with the increasing volatility was a considerably rare behavior among its implied volatility and average historical volatility.

Implied volatility is reflected in option pricing — the higher the implied volatility, the more expensive the options are. In most instances, implied volatility would be higher than the historical volatility, meaning that buyers who long the options are paying more for what would be a smaller stock movement in the future. Due to the large and unpredictable swings of Tesla stocks, this phenomenon is seldom true. For Tesla, average historical volatility is "intertwined" with implied volatility — the two values continuously fluctuate and surpass one another. This rare behavior therefore creates a potentially low risk but profitable pattern that will be demonstrated later.

I. Stage One: Technical Analysis, Trading Strategy and Demonstration

A. Technical Analysis and Strategy

a. Thoughts on Historical Volatility and Moving Averages:

In the trading of various assets, moving averages are the most fundamental yet useful indicator for signaling how strong a trend is. For instance, when a candle stick exceeds or drops below, say 20 period moving average, this could signify the beginning or the end of a trend. Similar idea could be extended towards what average historical volatility stands for with respect to implied volatility.

Suppose we let average implied volatility be a signaling line, if it is surpassed by average historical volatility of the past 30 days, not only does this mean that the stock is currently gaining volatility, the option price could be falling with the lowering of implied volatility — or perhaps just not rising as rapidly as the stock gaining volatility. With average historical volatility rising above implied volatility, this could also indicate that the options are priced relatively fairly.

b. Intuitions and Strategy:

Since volatilities only tells the range that the stock could potentially fluctuate in the future or have already fluctuated without having much to say about the direction the stock is moving, a financial instrument that could generate profit when the stock goes either direction is desirable — some such instruments include strangles and straddles, but for this project only straddles were used.

Even though constructing a straddle would cost twice as much as simply getting into one of the contracts, at times when the stock price has the potential to make significant moves in the near future, we could lock in profits. Assuming the market is expecting a major event, option price would be high, and

that's undesirable for option buyers. As a result, the goal is to discover an optimal entry point that guarantees fair option value and an optimal exit point where we either take profit or cut loss.

As discussed in part a, the promising event that could suggest that the options are priced fairly is the moment when average historical volatility rises above the implied volatility. Thus, we would like to obtain a straddle at that very moment and wait for a dramatic movement of Tesla stock to take place. It would later be shown in part B's demonstration that the stock would have finished making its biggest moves approximately at the instant when average historical volatility and implied volatility diverged to extreme — this would therefore be the exit signal. In cases where they don't diverge significantly, another exit strategy that would be shown in part B is needed.

B. Demonstration: Two Types of Transactions (Each Labeled 1~18)

a. Type one (Transaction 1, 2, 4, 5, 6, 11, 12, 14, 15, 17 and 18):

As shown in Figure 1, whenever historical volatility exceeds implied volatility, we purchase a straddle. The options in the straddle should have a strike price that's close to the current stock price and matures in approximately 1 to 1.5 months. It can be observed that oftentimes by the second arrow, Tesla stock would have already made substantial moves when average historical volatility diverges significantly from implied volatility. Thus, we sell/exercise the straddle. In most instances such as transaction 1, 2, 4, 5, 6, 11, 12 and 17, we could generate large profit. As for 14, 15 and 18, we would either lose a small amount of money or have modest gains.

b. Type two (Transaction 3, 7, 8, 9, 10, 13 and 16):

Similar to type one, whenever historical volatility exceeds implied volatility, we also get a straddle with the same parameters. However, in contrast to type one, the divergence between average historical volatility and implied volatility didn't exceed the observable threshold. Thus, the time when average historical volatility once again dives below implied volatility would also be the signal for us to exit the position. In most instances such as 3, 7, 8, 9 and 10, we would either lose a small amount of money or have modest gains. As for 13 and 16, we would have substantial gains similar to that of type one transactions.

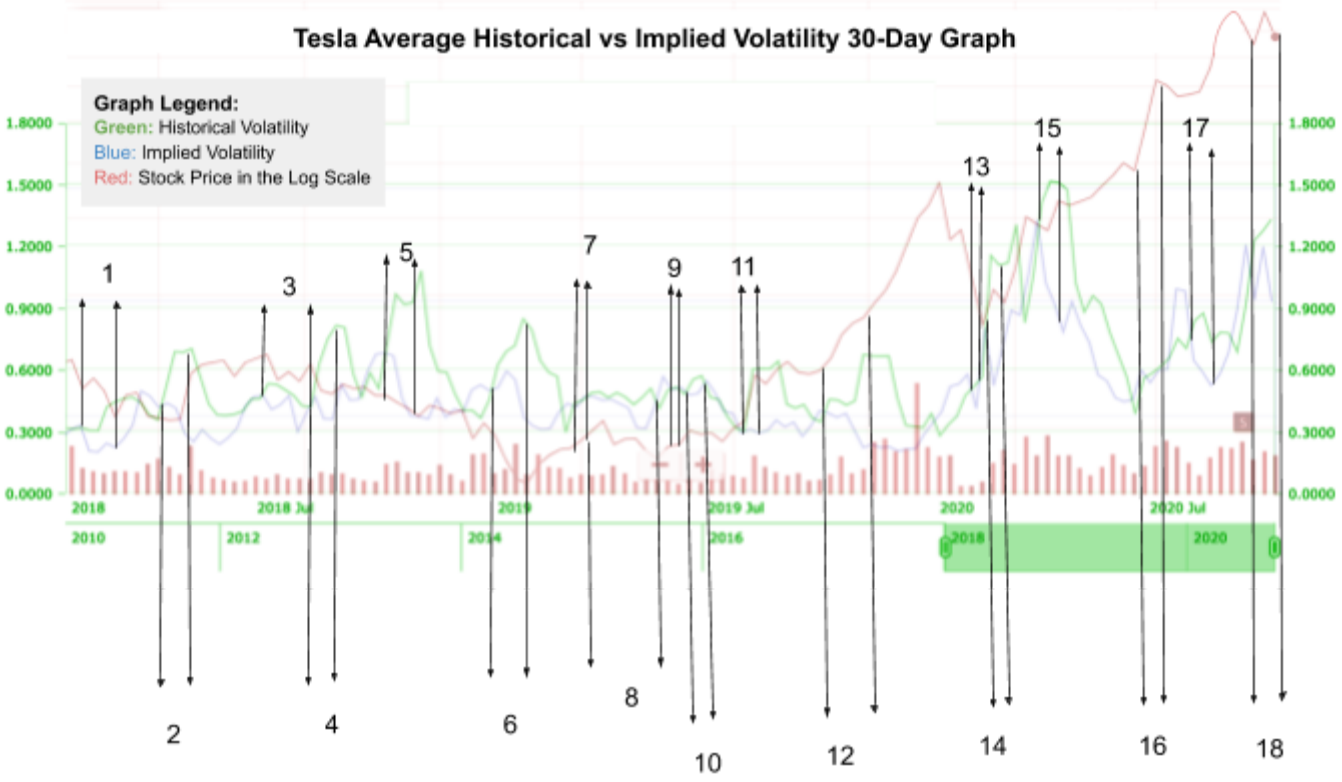

Figure 1. An overlaying graph of 30-day average historical volatility (green) and 30-day implied volatility (blue) from AlphaQuery along with the stock price (red) in the log scale from YahooFinance. (01/01/18 ~ 09/25/20)

II. Stage Two: Implementation of Strategy Through LEAN Engine

A. Algorithm and Performance Display:

```python
import numpy as np
import pandas as pd
from datetime import timedelta
from scipy import stats
from math import floor
from numpy import sqrt,mean,log,diff

class VolatilityTradeAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2017, 1, 1)
        self.SetEndDate(2020, 9, 25)
        self.SetCash(1000000)
        self.numdays = 30
        self.equity = self.AddSecurity(SecurityType.Equity, "TSLA", Resolution.Minute).Symbol
        self.option = self.AddOption("TSLA", Resolution.Minute)
        self.symbol = self.option.Symbol
        self.equity.hist_window = RollingWindow[TradeBar](self.numdays)

        #set our strike/expiry filter for this option chain
        self.option.SetFilter(-1, 0, timedelta(1), timedelta(2))

        #self.SetBenchmark("TSLA")

    def OnData(self,slice):
        #if TSLA data aren't there, return
        if not slice.Bars.ContainsKey("TSLA"): return
        self.equity.hist_window.Add(slice["TSLA"])
        price = pd.Series([float(i.Close) for i in self.equity.hist_window],
                          index = [i.Time for i in self.equity.hist_window])
        if len(price) < self.numdays: return
        aveVol = self.vol(price)

        for i in slice.OptionChains:
            chains = i.Value
            # sorted the optionchain by expiration date and choose the furthest date
            expiry = sorted(chains,key = lambda x: x.Expiry, reverse=True)[0].Expiry
            # filter the call and put contract
            call = [i for i in chains if i.Expiry == expiry and i.Right == OptionRight.Call]
            put = [i for i in chains if i.Expiry == expiry and i.Right == OptionRight.Put]
            # sorted the contracts according to their strike prices
            call_contracts = sorted(call,key = lambda x: x.Strike)
            if len(call_contracts) == 0: return
            #choose the contract with the highest strike price (the price that's the closest to current stock price)
            self.call = call_contracts[0]
            #find corresponding call to create a staddle
            for i in put:
                if i.Strike == self.call.Strike:
                    self.put = i
            #average of implied volatility between call and put, which is roughly half way between
            IVave = (self.call.ImpliedVolatility+self.put.ImpliedVolatility)/2
            #buy straddle when historical volatility rises beyond IV, meaning that options prices are fair while volatility is relatively large
            if not self.Portfolio.Invested:
                if IVave<aveVol:
                    self.Buy(self.call.Symbol, 10)
                    self.Buy(self.put.Symbol, 10)
            elif aveVol - IVave > 0.2:
                self.Liquidate()
            elif IVave > aveVol and self.Portfolio.Invested:
                self.Liquidate()

    #calculates historical volatility for length x amount of time (similar to moving average)
    def vol(self,x):
        constant = np.column_stack([np.ones(len(x)), x])
        r = diff(log(constant))
        r_mean = mean(r)
        diff_square = [(r[i]-r_mean)**2 for i in range(0,len(r))]
        std = sqrt(sum(diff_square)*(1.0/(len(r)-1)))
        volatility = std*sqrt(self.numdays)
        return volatility
```

Figure 2. Trading algorithm created through the LEAN Engine on QuantConnect Algorithm Lab

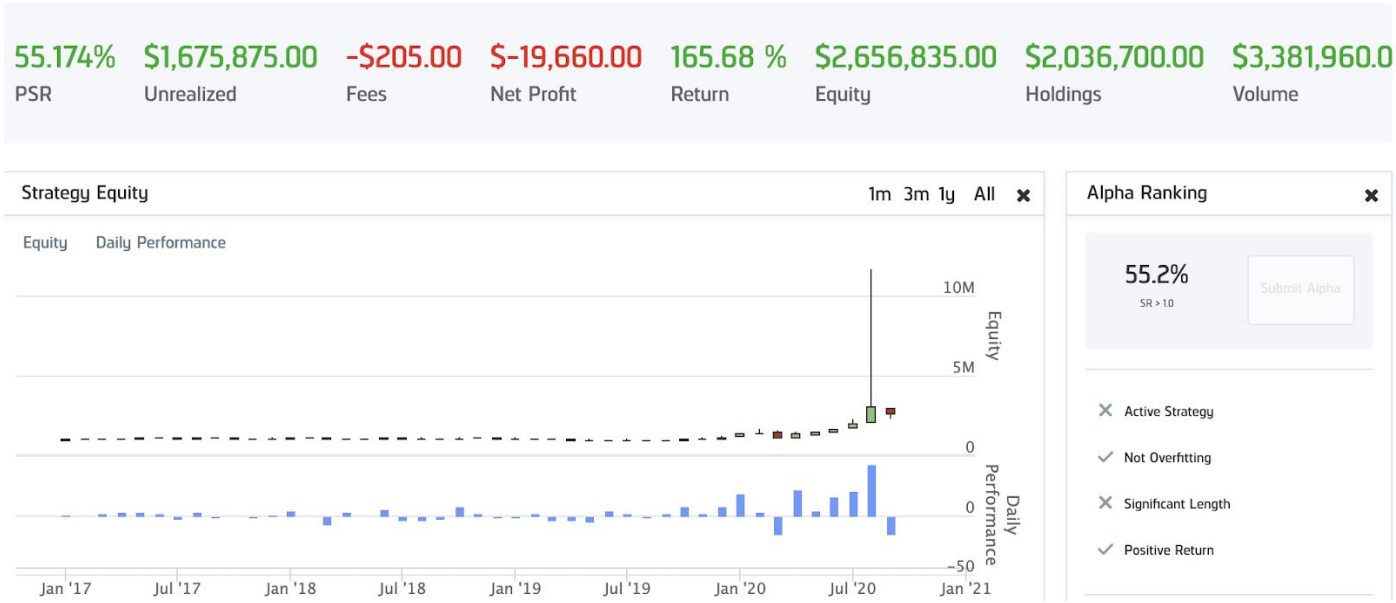

Figure 3. Screenshot of the strategy's performance starting from 01/01/17 through 09/25/20

Figure 4. Screenshot of the strategy's performance starting from 01/01/17 through 08/27/20

B. How the Algorithm Works:

a. Algorithm (Figure 2):

Some minor properties of the program include Initialize(), a constructor that sets the parameters such as trading time frame, monetary amount, securities type considered in the model, and the rolling window that would be used in OnData() to assist trading, and Vol(), a helper function that calculates average historical volatility. Since OnData() contains the main trading strategy, this is what we will be focusing on.

With option expiry date and strike price range already being filtered in Initialize(), OnData() first picks the correct options with the same strike price and expiry date to construct the straddle, then decides whether to get into the position. In this case, we construct the straddle by first picking the call option that has a strike price that's closest to the current stock price, then choose the corresponding put option that has the same parameters. After obtaining implied volatility from option prices and calculating average historical volatility with Vol(), decisions in regards to whether to purchase the straddles can be made, and the strategy is simple: Whenever average historical volatility (denoted by aveVol variable) exceeds implied volatility (denoted by IVave), we would like to purchase 10 units of straddle, and whenever the average historical

volatility becomes 0.2 larger than or dips below implied volatility, we get out of the position by liquidating all holdings of the portfolio.

The average historical volatility is calculated using thirty one-minute stock data, and the implied volatilities of the options in the straddle are mostly two-day implied volatilities (since the options expire in two days). These mass discrepancies in the use of time frame in calculation will be explained in part C of this section — this is the critical factor that may have significantly reduced the stability and rigorousness of the algorithm.

b. Performance (Figure 3 and 4):

Starting from 01/01/17 through mid-2019, the algorithm reached a maximum gain of nearly 15% and a maximum loss of 5%, which is a stable though mediocre performance. Since mid-2019, the algorithm started to create massive gains. During the stock market crash in March, the gain has dropped from 50% to a mere 6%. Thereon, with Tesla becoming even more volatile and the stock market recovering, up till 09/25/20 the gain has exceeded 165% (Figure 3). The performance graph through 08/27/20 is also included to exclude extreme values on 08/28/20 due to stock split (Figure 4).

C. Weaknesses of the Algorithm:

a. LEAN Engine Constraints:

Due to LEAN Engine only providing option price data by minutes resolution, the original plan was to produce average historical volatility data using daily stock prices, then purchase straddles sometime during the trading session when appropriate. However, the program only allows for one data resolution, so even if stock price data resolution is set to daily, the algorithm would be fed data in minutes resolution. This led to the eventual decision of trading on an extremely high frequency with options expiring within two days instead of deploying the daily trading strategy devised in part I.

b. Volatility Calculation:

Originally, calculating volatility with minutes data and extending it to desired period length would be mathematically correct. However, Tesla's fluctuations, even in minutes, are overwhelmingly large, and multiplying them by such time span would significantly skew the result. If the average two-day volatility were to be properly calculated, the minute standard deviation would have been multiplied by a factor close to the square root of 780 based on the basics of volatility calculation (there are 390 minutes in a trading day, so two days would contain 780 minutes). However, due to the overly high fluctuations in minute scale, conventional calculations are abandoned, leading to the decision of multiplying by a factor of square root of 30 (30 was originally used in attempt to implement the strategy in part I, but after having the program backtested, this seemed to be a more sensible option).

c. Threshold Flaws:

The threshold in such trading programs could be loosely constructed at times. In this case, 0.2 is an approximation from the observable threshold limit that would've been set if the original strategy were to be implemented.

III. Conclusion

Even though the algorithm does in a way benefit from the uprise of Tesla stocks in the recent year, results produced by the algorithm do confirm that the strategy could be feasible — with proper algorithm and trading system, I believe my original strategy could be significantly more effective and secure greater returns. Despite the fact that investing in Tesla stocks in 2017 and selling them in 2020 would have produced significantly better returns than the algorithm, this strategy does promise stability and was proven to be occasionally profitable.

Due to constraints of LEAN Engine, my strategy was implemented in a much riskier fashion, and the algorithm created should only be viewed as a rough implementation of my original idea. For possible future study, I would like to explore other algorithm options that could simulate and execute my original daily strategy, and perhaps incorporate it into real-life trading scenarios.