Automating Options Trading

Rishi Mody
University of Massachusetts Amherst
Amherst MA
rmody@cs.umass.edu

Abstract

Recent times have seen major banks, hedge funds, mutual funds and stock brokers embrace the use of technology, computers and programming. It has led to the markets reorganizing themselves. Traders leverage technology to not only analyze their data to help make better decisions but also automate the trading process. Access to markets has become easier and faster using algorithmic trading. A high level description of algorithmic trading includes making trading decisions, submitting orders and managing said submitted orders.

Large orders can be executed using pre-programmed trading instructions accounting for variables such as time, price, and volume to send small slices of the order out to the market over time. It is widely used because these institutional traders need to execute large orders in markets that cannot support all of the size at once. Thus it is mainly used to minimize cost, market impact and risk during the execution of an order. Also known as black box trading, these encompass trading strategies that are heavily reliant on complex mathematical formulas and high-speed computer programs.

1. Introduction

Algorithmic Trading provides tons of advantages over manual trading: -

- Trades are executed at the best prices.
- Instant and accurate trade order placement.
- The timing of a trade is more precise and instantaneous.
- Reduces transaction costs.
- Can take into account multiple factors at the same time.
- Less manual errors because of reduced human intervention.

These advantages have seen algorithmic trading being adopted across the board by multiple institutions including all sorts of traders short term, medium and long term investors, pension funds, mutual funds, market makers, speculators etc.

Algorithmic Trading has certain minimum requirements for it to work. They include:

- Ability to program the required trading strategy into computer code.
- Network connectivity and access to trading platforms for placing the orders.
- Access to market data feeds on the basis of which the algorithm will make decisions.
- The ability and infrastructure to backtest the system once its built before using it.
- Available historical data for backtesting.

In this project, I explore these different aspects and requirements with a focus on options in particular. I further simulate possible options trades using straddles as a strategy. The bid-ask average and stock price are deliberately added to the tick data. Using this data, straddles are sold and settled in a short term. Multiple runs are simulated by changing the time period of settlement and spread, following which the most optimal combination for an option is found. This is stored and along with the original data is used as input to the machine learning model which would predict the parameters to maximize profit at any given instant.

2. Problem Statement

Intraday tick based minute data is provided for options available during the period of January 2016 - June 2017. This includes features like bid price, ask price, strike, expiration date, time bought, date of option. Using this data and by making possible additions, the best time period of settlement and spread is to be predicted using machine learning models. Thus automate the process of options trading, when the model is supplied with real time data.

3. Related Work

The field of finance has been an elite club where one would require a lot of financial power as well as contacts to succeed. With the increasing involvement of technology, a college graduate in his dorm room can design an algorithm to trade stocks, futures, options, currency etc. Various online platforms provide data, forums as well as infrastructure for one to backtest the algorithm. Quantopian provides some really good blogs [7] which describe how one can use machine learning to create and improve algorithms.
Investopedia [5] is an online resource which provides articles and tutorials which not only describe basic financial tools, instruments but also algorithmic trading basics and strategies.
Very few papers address algorithmic trading directly. Domowitz and Yegerman (2005)[2] study execution costs for a set of buy-side institutions, comparing results from different algorithm providers. Chaboud et al. (2009)[3] study algorithmic trades in the foreign exchange market and focus on its relation to volatility.

In 2009, a paper published out of UC Berkeley [4] aimed to examine algorithmic trades and their role in price discovery of almost 30 stocks listed on the DAX. They manage to explain about algorithmic trades and what these algorithms try to focus on when provided with financial data. This has been useful to understand more about the use of technology in the stock market.

Jason Leung, of MIT published a report on the application of machine learning in the stock market [6] which provided a lot of useful insights and understanding into the use cases and nuances of machine learning in the field of finance.

4. Dataset

The dataset has been provided by AlgoSeek, a firm with expertise in providing intraday US market data that is useful to academia, hedge funds, banks and individuals worldwide.
I have access to over a year and half of data starting January 4^{th}2016 of tick based minute data of options of QQQ, SPX and SPY. This data consists of multiple details of an option at each given instant. The features include strike price, expiration date, the bid and ask prices at the start of the minute, the end of the minute even the high and low prices during the minute. It provides information on if and when a trade happened, the volume, the bid and ask of the trade. For the project all the models have been run on the QQQ dataset.

4.1. Pre-processing

The data provided was well divided into multiple directories. Each directory denoted the date when a particular set of options could be bought. Each having multiple .csv files containing all the above mentioned features. Each of these multiple file indicated the date when the options would expire. The range of the validity period was from end of the week to a year later. The focus for this project is to understand the behaviour of options over the small term as longer duration would require the incorporation of factors including but not limited to volatility, market conditions etc. Thus for a short term we consider only those files which provide an expiration of up to a month. This gives us 4 files for each of the 246 days of the year 2016. These files have to be combined to form a single .csv file which would make the entire task of simulating trades as well as feeding it to the machine learning model more straight forward. The combining task was handled by writing a simple python script which would read each .csv file and write it into the big file.

The final data file contains almost 60 features of which many end up almost redundant data therefore some of them can be removed. We will be considering data at each single minute during the intraday trading hours, hence features which indicate the bid and ask at the opening of the minute can be removed and the one at the closing can be considered for all calculation purposes. Some important features such as strike, bid/ask price can be missing for some instances, these instances are completely removed. Also for simplicity we only consider those instances which are at the hour mark times.

4.2. Processing

Features are deliberately added to final data so as to be useful for the straddle strategy with which we will be simulating our trades. Firstly, we calculate the bid-ask spread for each of the option instances and add it to the data. We then estimate the stock price of the option and keep the price constant over each minute being considered. The stock price is estimated using the following formula:

$$S = C^{mid} - P^{mid} + X$$

where S - Stock Price
X - Strike price
C^{mid} - bid-ask average of the call with the above strike
P^{mid} - bid-ask average of the put with the above strike

For a particular minute, we use the call and put pair which has the same strike and least difference in the bid-ask average. This creates the final database to be used to experiment with.

5. Methods

When trading in options, one must have a sound and robust strategy. These strategies are instrumental in reducing the risk and maximizing the profit for a trader. There a variety of strategies to which one can adhere and use, some of them include covered call, married put, straddle, protective collar etc. For this project I have focused on using straddle as a strategy specifically short straddles. A straddle is when an investor holds a position in both a call and put both having the same strike price and expiration data. The investor must pay both premiums but this strategy almost guarantees a profit to him as long as the stock price changes significantly.

We simulate short straddles by selling at the money call and put options having the same expiration and stock price. We focus on short term trading hence for simplicity we aim to settle within a period of 1 hour after shorting or at the end of the day. For straddles sold a the end of the day, the only settling period considered is at 10 am the next day. There are certain conditions applied on the type of straddle as well. We account for a spread of 0, \$0.5 and \$1 on either side of the strike price. This gives us 6 combinations at which we judge a particular option. An option is assigned a label denoting the combination that provides the most profit overall through it.

Table 1: Label corresponding to time period and spread

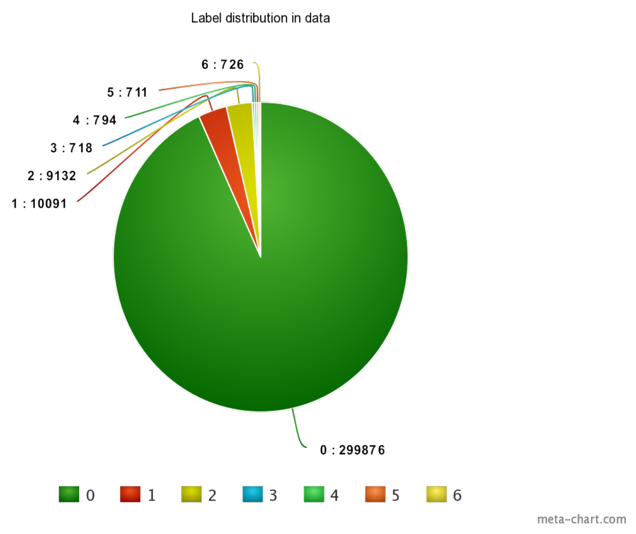
Timeperiod of Settlement	Spread	Label
After an hour	0	1
End of the day	0	2
After an hour	\$0.5	3
End of the day	\$0.5	4
After an hour	\$1	5
End of the day	\$1	6
Do not sell in the first place		0

Thus each of the options irrespective of it being a put or call is provided with a label.

This combination of option data and the label forms our data for the machine learning model. The number of features is also reduced to accommodate 36. The data is then divided into a train and a test set. We supply the train data to multiple ML models and try and predict the labels for the test data. Through this machine learning process we will be able to predict given the set of existing conditions and properties of an option what would be the most optimal strategy to use to maximize profit.

Table 2: Data Statistics

Label	Count
1	10091
2	9132
3	718
4	794
5	711
6	726
0	299,876
Total	322, 048



The train and test set themselves are divided such that train_x and test_x contain all the features and information while train_y and test_y contains the labels.

The entire data is heavily biased towards the label 0 i.e. that particular option must not be sold. So when we divide the data into train and test I ensure that each label is well represented in both the sets. To feed the data to the ML model, we must convert all the strings of information to integer or decimal values.

5.1. Models:

We evaluate the following models and feed them with the above selected features:

i) **K-Nearest Neighbors:**

KNN is a non-parametric classifier that classifies a new instance by using the majority vote of its K neighbors, which are located at a minimum distance from this instance. For a distance $d : R^d \times R^d \rightarrow R$, x = new instance, $N_k(x)$ = K Nearest Neighbours, II is an identity function, thus KNN classification is expressed as:

$$f_{KNN}(x) = arg \quad max_{y \in Y} \sum_{i \in N_k(x)} II[y_i = y]$$

KNN has been used as it is simple and powerful. It does not require tuning of complex parameters to build a model.

It is robust to training data noise. The cost of the learning process is zero and new training examples can be easily added. Also it gives good training accuracy.

ii) **Random Forest:**

Random Forest is an ensemble classifier that combines multiple decision trees and thus overcomes the decision trees limitation of overfitting. It involves bootstrap aggregating/bagging, is highly accurate and has a fairly quick prediction time.

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(\hat{x})$$

RF has been used as it provides one of the best accuracies among all algorithms. It runs efficiently irrespective of the size of the database. It also has an effective method to estimate missing data and its accuracy is unaffected when a large proportion of data is missing.

iii) **SVM:**

A Support Vector Machine is a discriminative classifier. Its decision boundary is same as the Logistic Regression boundary. Its decision boundary is of the form:

$$f_{SVM}(x) = sign(w^T x + b)$$

Machine Learning models work better when we can find a better separation boundary across node categories. SVM tries to find such a clear separation using the trick of basis expansion which projects the input feature into a large dimensional space using kernelization as an optimization trick for computation. The loss function of SVM tries to maximize this margin.

6. Experiments

6.1. Hyperparameter optimization:

Hyperparameter optimization is the process of tuning the hyperparameters of a learning model to improve its performance. It depends a lot on the type and amount of data provided. To find the optimal set of hyperparameters I have done 5-fold crossvalidation through GridSearchCV. This splits the training set randomly into 5 parts. Now, one block is chosen for validation V while the rest are used for training . Every block individually must once be a part of validation. For each loop of a block being a part of V, we must loop over the range of hyperparameters, and finding out the accuracy of the data on . We can use any machine learning algorithm for this. Once we have stored these, we must find the average accuracy of each hyperparameter across the 5 folds. The one with the highest average accuracy is denoted as the most optimal value of hyperparameter in that range to be used on the original training data using that particular algorithm.

Hyperparameters of each model fed into the grid are:

i) **K-Nearest Neighbors:**

- n_neighbors : [1,2,5,7,10]
- weights : [uniform, distance]
- algorithm : [auto, ball_tree, kd_tree, brute]
- metric[euclidean,manhattan,chebyshev]

ii) **Random Forest:**

- n_estimators [10, 50, 100]
- criterion ['gini', 'entropy']
- max_depth:[5,25]
- min_samples_split:[2,5].

iii) **SVM:**

- kernel:['rbf','poly']
- C:[1, 2.5, 5]
- degree:[2,3]

I experimented with random values at first to try and get a sense of the range in which the hyperparameters giving the best performance lie. After which in the next iteration I tried to get the most optimal values in and around that range.

7. Results

7.1. Evaluation Metrics

Evaluating a model is the most important task in a data science project which delineates how good the predictions are. There are multiple metrics through which one can judge the performance of a model. I have used two of them, namely Accuracy and F-score.

7.1.1 Accuracy

Accuracy is the most intuitive evaluation metric. It involves counting the predictions which are actually correct when compared with the labels from the test_y. This count is divided by the total number of predicitions giving the accuracy. Therefore,

$$Accuracy = \frac{Correctly\, predicted\, labels}{Total\; Predicted\, labels} * 100$$

Many a times, having high accuracy does not correspond to having a better model. Accuracy is a precise metric of evaluation only when the dataset is well balanced, with each label getting an approxiamtely equal presence in the dataset. As mentioned before, this dataset is more favourable to the label '0' thus we explore the use of F1-Score as an alternative evalution metric.

7.1.2 F-Score

The F-Score is derived using precision and recall evaluation metrics.

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$Precision = \frac{True\,Positive}{True\,Positive\, + False\,Positive} * 100$$

Recall is the ratio of correctly predicted positive observations to the all observations in actual class.

$$Recall = \frac{True\,Positive}{True\,Positive\, + False\,Negative} * 100$$

F-Score takes into account the weighted average of Precision and Recall. Therefore it takes into account both false positives and false negatives. It is more useful than accuracy especially when the dataset is unevenly balanced.

$$F1 - Score = 2 * \frac{Recall\, * \,Precision}{Recall\, + Precision} * 100$$

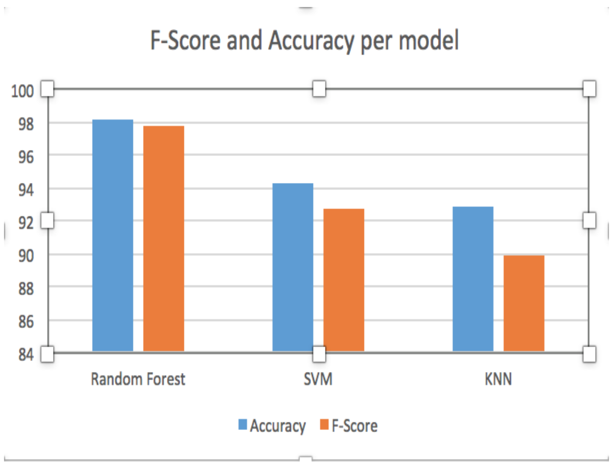
7.2. Model performance

Table 3: Best Hyperparameters per model

Model	Hyperparameters
Random Forest	{'min_samples_split': 5, 'max_depth': 25, 'n_estimators': 50, 'criterion': 'gini'}
KNN	{'n_neighbors': 7, 'weights': 'uniform', 'p': 2}
SVM	{'kernel': 'poly', 'C': 2.5, 'degree': 3}

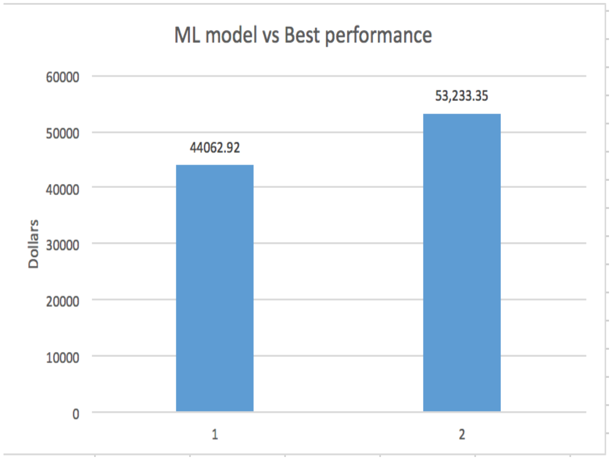
Table 4: Performance of Models

Model	Accuracy	F-Score
Random Forest	98.20%	97.82%
KNN	92.90%	89.96%
SVM	94.32	92.72



Analyzing the accuracy/ f-score does not help us in understanding the actual performance of the model. A better way would be to analyze how well the model performs at a given instant and compare it with the actual best performance of that instant in dollar terms. To simulate this I have stored the respective profits or losses made through the process. Thus we try to compare the performance the ML model with the ideal performance we have through the test data.

The following image shows us the ratio of gains obtained if we select the ML model to run and execute orders versus the profit obtained through the best decision giving maximized gains at each order which we had calculated while assigning the labels. Since the model does not have 100% accuracy it loses out on some opportunities to make profit and ends up selecting parameters that might lead to a loss.



Random Forest provides the best results of the three models. This is an expected outcome as Random Forest is a collection or ensemble of decision trees. A decision tree is a tree-like graph structure of decisions and their possible consequences. it includes chance event outcomes, resource cots and utility. It is useful in displaying conditional control statements. They are most commonly used to identify a strategy most likely to reach a goal, which is the main task undertaken in this project.

8. Conclusion and Future Work

We can leverage machine learning models, given the right set of model parameters and realization of the input data to better understand the volatile nature of options and try to predict their movement. Obtaining options data can be troublesome as very few companies keep a track of them. Most of them focus on stocks and futures. On obtaining the data, cleaning it and adding the required additional features takes a lot of time. The code runs for a very long time and it requires a lot of memory. To run the machine learning models, having an idea of the kind of data that one possesses makes it simpler to choose the type of model to run. Once the model/s have been decided then over multiple iterations one can find the model that would work best for the current strategy and dataset.

For the current project we have looked at only straddles as a strategy and focused on short term trades. The current model focuses only on the trades at the hour mark, it can be generalized into looking at trades at every minute of a trading day. As an extension, we could focus on options with expiry dates that are more than a month away. It would require the inclusion of more features such as volatility which could be incorporated using the Black-Scholes model[1]. Another possible extension would be to incorporate multiple strategies and the machine learning model would identify patterns when a straddle would be a better option as compared to a covered call or a married put. Thus when the model comes across an option available at the current time, it will decide the strategy in real time and accordingly execute an order. This model can be made more sophisticated by including a module for risk management. It would handle cases when due to unforeseen circumstances or a mistake by the model would lead to losses, and try to minimize or reverse them.

Acknowledgement

I would like to express my immense gratitude to Prof. Nikunj Kapadia, UMass Amherst for his support and for allowing me to work on this project as an independent study. His constant guidance and ideas not only helped me understand the financial details better but also guided the direction of this project.

References

[1] F. Black and M. Scholes. The pricing of options and corporate liability. https://www.cs.princeton.edu/courses/archive/fall09/cos323/papers/black_scholes73.pdf.

[2] I. Domowitz and H. Yegerman. The cost of algorithmic trading: A first look at comparative performance. https://www.researchgate.net/publication/237547529_The_Cost_of_Algorithmic_Trading_A_First_Look_at_Comparative_Performance.

[3] C. et al. Rise of the machines: Algorithmic trading in the foreign exchange market. https://www.federalreserve.gov/pubs/ifdp/2009/980/ifdp980.pdf.

[4] T. Hendershott and R. Riordan. Algorithmic trading and information. http://people.stern.nyu.edu/bakos/wise/2009/papers/wise2009-3b2_paper.pdf.

[5] Investopedia. Automated trading. https://www.investopedia.com/articles/trading/11/automated-trading-systems.asp.

[6] J. W. Leung. Application of machine learning: Automated trading informed by event driven data. https://dspace.mit.edu/bitstream/handle/1721.1/105982/965785890-MIT.pdf.

[7] T. Wiecki. Machine Learning on Quantopian. https://www.quantopian.com/posts/machine-learning-on-quantopian.