ORIE 5257: RFQ Assignment

Sagar Mehta spm255, Henry Lee hhl38, Conan Zai cz232

31 October 2020

Part I. Predicting Probability of Winning an RFQ

Approach

For the first part of the assignment, we decided to use a Neural Network (NN) based algorithm to evaluate the probability of winning an RFQ. In general, NNs have the flexibility to fit highly complex data due to their large number of free parameters (weights and biases between interconnected units). In our exploration, we found this to be true. NNs performed better than other classification algorithms and yielded the highest accuracy with both the training data and out-of-sample data. Specifically, we opted for a Multilayer Perceptron (MLP) classifier. The MLP is considered to be one of the more straightforward NN algorithms and involves a network composed of multiple layers of perceptron neurons. Because we only have 5,000 training samples, we tried to avoid any particularly complex NN variants. Our MLP model has 3 hidden layers, each of size 11, and uses a logistic sigmoid activation function. We chose the logistic activation function because we are interested in the probability of winning a trade; this probability (and the classification) has non-linear properties. Furthermore, the logistic activation function is bound between 0 and 1, normalizing the output of each neuron. We experimented with other non-linear activation functions such as tanH and ReLU functions as well and found them to yield similar results.

Feature Engineering

Out of the data provided to us, we chose to use the following columns to evaluate the probability of winning a trade: ‘Bond’, ‘Side’, ‘Notional’, ‘MidPrice’, ‘QuotedPrice’ and ‘Competitors’. The target we are attempting to predict is ‘Traded’. We combine the ‘MidPrice’ and ‘QuotedPrice’ into a single entity named ‘Diff’, which is (MidPrice-QuotedPrice)/MidPrice for Bids and (QuotedPrice-MidPrice)/MidPrice for Offers. We chose to omit ‘Time’ because consecutive time steps can represent different bonds being traded, which makes it difficult to extract any meaning from. We one-hot encoded the bond and counterparty features since they are categorical variables. We converted the binary feature side to ‘1’ for offer and ‘-1’ for bid. Then, we divided each feature by the max value for that feature to normalize.

Performance

Our MLP model performed well on both the training and out-of-sample data. As shown by the classification report of the model, our overall accuracy on the training set was around 90%, with 89% accuracy of predicting a ‘MISSED’ trade and a 92% accuracy of predicting a ‘DONE’ trade. On the out-of-sample set, we achieved an overall accuracy rate of 90%, which consists of a 90% accuracy of predicting a ‘MISSED’ trade and a 91% accuracy of predicting a ‘DONE’ trade.

Label	Training Sample	Test Sample
0 (Missed Trade)	0.89	0.90
1 (Successful Trade)	0.92	0.91

Figure 1: Accuracy of MLP Model on Training and Out-of-Sample Set

Shortcomings/Potential Improvements

The same reason that makes NNs so powerful is also one of its shortcomings. Through many layers of neurons, a NN is capable of learning arbitrary nonlinear functions very well. However, this leads to potential overfitting as new observations can lead to shifting model parameters. Outliers can cause the model to learn anomalies in the data that reduce performance. This could be overcome by incorporating a Bayesian approach through Decision Trees which would manage overfitting by splitting observations across leaves. Within the framework of our current model, we may be able to improve performance by solely focusing on features that carry the most information such as bond type and counterparty to see if we can eliminate some noise. More experimentation with the hyperparameters may also improve accuracy.

Part II. Quoting Optimal RFQ Prices

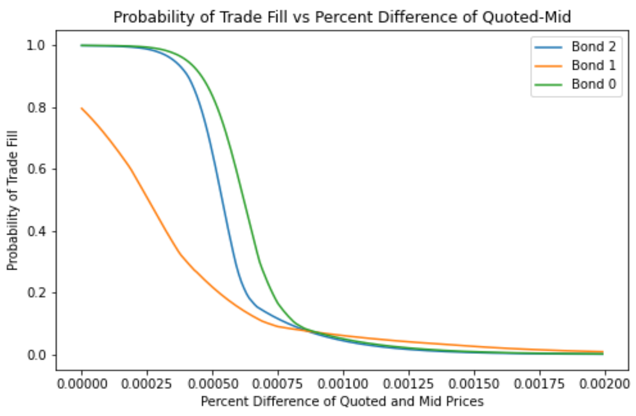


Figure 2 to right illustrates the probability of winning an RFQ as a function of the distance from mid price, expressed as a percentage. Bond 2, Bond 1, and Bond 0 correspond respectively to the first, second, and third new RFQs provided in the competition data. We observe that Bond 2 and Bond 0 have similar plots. Quoting exactly the mid price yields almost a 100% probability of winning the RFQ. There is a margin in which the probability remains around 100% and afterwards, it drops sharply.

However, with Bond 1, quoting even exactly the mid price yields only a probability of 80%, meaning there is no guarantee of winning this RFQ. From a distance of 0, the probability drops right away, with no margin.

Overview of Strategy

Our goal is to select a quote that will win the trade and be profitable. Naturally, the closer the quoted price is to the mid price, the higher the probability of winning the trade. However, if a market maker quotes too close to the mid, they risk suffering from "Winner's Curse" of overestimating the value and suffering a loss at the next tick, if the mid price moves through their quoted price.

Initially, we attempted to predict the next mid price in order to determine the optimal price to quote. We tried various regression models, including Random Forest Regression and Bayesian Ridge Regression. However, all of them performed relatively poorly compared to the actual mid prices on training and test data. They failed to capture the seemingly random fluctuations in mid price and eventually, we abandoned this approach. Intuitively, it is easy to appreciate the difficulty of this problem. Markets are volatile and predicting the next tick price move of a security is a colossal endeavor.

Rather than predict the next mid price, we reasoned that we could modify our MLP model from Part I to incorporate the 'PL' column as a label. Therefore, instead of using just the 'Traded' column as our target label, we created a new column named 'Success' which is a '1' only if the trade was won but the PnL for that trade was negative, '2' if the trade was won and profitable, and '0' otherwise.

After normalizing the features to have each feature obtain a maximum value of 1, we use a MLP classifier with 3 hidden layers, with layer sizes (11,11,11). This was arbitrarily selected through experimentation. We also experimented with activation functions such as ReLU and Logistic, and ultimately decided to choose tanH since it produces the highest precision for label '2' (executed trade and positive PnL).

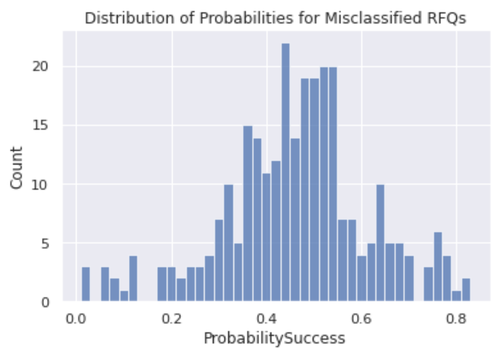
Label	**Training Sample**	**Test Sample**
0 (Missed Trade)	**0.85**	**0.88**
1 (Successful Trade with Loss)	**0.58**	**0.46**
2 (Successful Trade with Profit)	**0.57**	**0.56**

Figure 3: Accuracy of MLP Model on Training and Out-of-Sample Set

It is worth noting, however, that the precision of classifying a successful trade with profit is only around 60%. This is significantly lower than the accuracy of predicting a won or lost trade in Part I. In the following, we attempt to rectify our model's inaccuracy in predicting a "successful" trade.

Approach 1

The first approach analyzes the distribution of RFQ probabilities of getting a fill and it being profitable. First, we created 2 new columns "Success01" and "PredictedSuccess01" which convert "Success" and "PredictedSuccess" label values into binary label values. This is elaborated on in the included Jupyter notebook. Next, we filtered out only those RFQs that are misclassified (e.g. have either a predicted label of 1 and an actual label of 0, or a predicted label of 0 and an actual label of 1). We then plotted the distribution of probabilities of a won and profitable trade for all the misclassified RFQs.



From the histogram above, we see that the number of misclassified RFQs is insignificant for probability values greater than approximately 0.6. Based on this, we will only quote prices with the intent to trade if the probability is greater than 0.6. Even with this imposed condition, we would still be able to make about 27.1% (80/295) of all successful trades. Based on this number, we believe we can strike a good balance between trading frequency and overall trading PnL.

Approach 2
Our second approach is more quantitative since we aim to find the probability threshold which maximizes the total out-of-sample PnL. The algorithm iterates over all possible values of probabilities attached to the out-of-sample RFQs, and returns the optimal probability threshold. Our result shows that a maximum PnL of $70770.40 is achieved if we only trade RFQs with probabilities of greater than or equal to 0.3506.

Comparing both approaches, we observe that they produce dramatically different results. Instead of averaging the results, we think that the probability threshold produced via the second approach is more promising as it is more likely to give an overall positive PnL. Hence, we decided to use a probability threshold of 0.3506 to compute the quoted prices of our competition RFQs.

Derivation of Optimal RFQ Price
Given the desired probability threshold, we varied the "diff" (scaled distance from mid) to find the highest "diff" which meets our probability threshold of 0.3506. Then, we used this "diff" value to solve for the RFQ quoted price. The following formula is used to backsolve for the quoted price: $Diff = \left|\frac{QuotedPrice - MidPrice}{MidPrice}\right|$

The derived quoted prices for the 3 competition RFQ orders are listed below:

Bond	Bond_0	Bond_1	Bond_2
QuotedPrice	$110.90	$118.18	$134.50

Shortcomings/Future Improvements
Our model has room for improvement in predicting the probability of an RFQ quote being both executed and profitable. Unlike our model in Part I which yielded an accuracy of 90% for predicting whether an RFQ would be traded, this model has much lower accuracy. It follows that there is some insight in the data that presently eludes us. In future work, it would be reasonable to try to find the relationship between the type of bond and counterparty and the probability distribution of being successful on that trade. For example, some counterparties are specifically more knowledgeable about a certain product and in trading with them, you are more likely to lose as a market maker. It would also make sense to try and find the relationship between how the next mid price moves based on bond and counterparty types. Another possible improvement would be to differentiate between degrees of success in a trade. Currently, we have binary labels with profitable trades in one category and unprofitable trades and missed trades in another. Some RFQs have characteristics that yield a high probability of being extremely profitable whereas others straddle the line of breaking even. Categorizing these in an appropriate way could improve predictability. Aside from this, naturally more data and more detailed data would help, but that is not always available.