Penalizing LSTM Output Size for Short-Term Stock Market Technical Analysis

Franklin Doane
dept. of Computer Science
Tennessee Technological University
Cookeville, United States
doanefranklin89@gmail.edu

***Abstract*—**

***Index Terms*—**

I. Introduction

II. Related Work

A. LSTMs for Market Prediction

Mei Sun, Qingtau Li, and Peiguang Lang, published work that was focused on using LSTM architecture for market price prediction [1]. Much like this work, they were aiming to use LSTM architecture for short-term price prediction; however, their focus was heavily on featuring engineering through the use of singular value composition (SVD). Qi Li, Norshaliza Kamaruddin, Siti Sophiayati Yuhaniz and Hamdan Amer Ali Al-Jaifi also have done work in this area [2]. Their work was on the use of LSTMs for price prediction. They aimed to augment it with the use of Symbolic Genetic Programming (SGP). While both of these use combine different methods with LSTM architecture to predict market prices, this work is distinct it its use of penalties for model logits.

B. Penalizing Models for Market Predictiton

Huifeng Jiang, Xuemei Hu, and Hong Jia explored different methods of penalizing logistic regressors in order to increase their performance as market indicators [3]. The authors explore penalties that apply to all model coefficients. In this work, the penalties being explored will apply only to model logits.

III. Size Penalty Ablation Test

A. General Methods

Dataset creation: The all datasets were created by downloading data consisting of daily OHLCV values for each stock being used. The list of stocks being used was determined by picking the 1500 stocks with the largest market cap that also have data for the time periods that the datasets intend to cover.

Training setup: The dataset is loaded and the day of the week is 1-hot-encoded into 5 separate features that are added. The data then gets cut into time series, each with a length of 50. The data from the day following each time series is used to create the label, which consist of the relative increase in high price and low price compared to the close price of the previous day. Each training example gets all of it's price data normalized together on a scale of -1 to 1. Similarly, the volume data gets normalized on a scale of -1 to 1. 15% of the training examples are sliced off of the back of the dataset to become the validation set. The ordering of the training set gets shuffled according to a random seed. The training data gets split into whole batches of size 200. The model is initialized with random parameters. The model architecture is depicted in "Fig. 1".

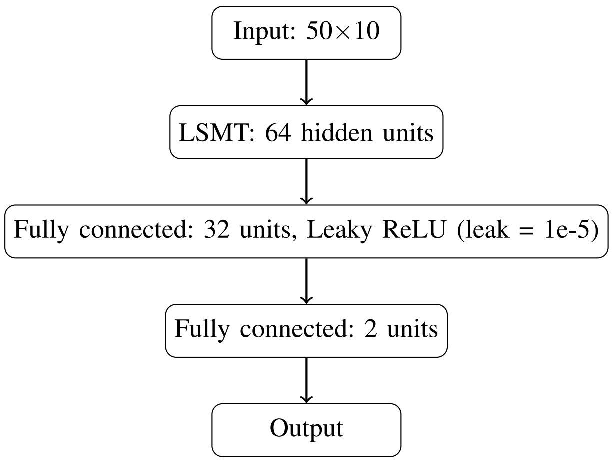

Fig. 1. Model architecture.

Training methods: The loss for each batch is calculated with formula "(1)" . The formula utilizes Mean Squared Error loss and a magnitude penalty for the logits.

$$Loss = mse(Y, \hat{Y}) + \lambda \sum_{i=1}^{B} (\hat{Y}_{i,1}) - \lambda \sum_{i=1}^{B} (\hat{Y}_{i,2}) \qquad (1)$$

Here the logit at index 0 is the high price that the model is predicting, and at index 1 is the low price. B represents the batch size, and λ is the size penalty coefficient. The loss is backpropagated for each batch, and the parameters are updated with the Adam optimizer. The learning rate for the optimizer starts at 1e-3, and is set by PyTorch's LowerOnPlateau scheduler. The metric tracked by the scheduler is validation loss. At the end of each epoch, validation occurs. The model predicts all of the examples in the validation set and loss is calculated. Unlike during training, the loss in validation is calculated as a pure Mean Squared Error loss with no penalties. The loss is then used to update the learning rate scheduler.

Testing: The testing dataset undergoes feature calculation and normalization in the same manner as the training set. An agent is created with a trading balance of \$100. The agent employs a simple trading algorithm that aims to only use strong positive signals. If the model predicts a high price increase $\geq 5\%$ and a low price increase $> -0.1\%$, the agent purchases the stock at the close price for that day. The agent's balance is evenly split between all of the stocks chosen by the model, as long as each getting a minimum of \$1. For each position that the agent opened, it closes the position at a price from the following day. The agent is attempting to sell the position a $\geq 5\%$ increase, but will stop-loss if the price is $\leq -0.1\%$. The outcome of the position is determined using the logic detailed in "Fig. 2". The price that each position would've sold for is used to update the agent balance. Each position is checked first to see if the trading parameters would have caused it to sell at the open price. Then it is checked to see if it would sell during the day. Note the outcome where an a position value of both $\leq -0.1\%$ and $\geq 5\%$ occur during a day. Since it is ambiguous which would've happened first when using daily candlestick data, it is considered a stop-loss in the interest of being conservative. Each position that is sold at or above the aim of a 5% increase is counted as a successful trade and a true positive from the agents perspective. Each position that is sold for less than or equal to -0.01% of the purchase price or that sells at the end of the day due to having no other opportunities to sell is considered a false positive. After process is repeated for every timestamp, the total balance increase is calculated for the whole time span, along with the rates at which each of the outcomes occur. These rates are used to calculate classification metrics like precision.

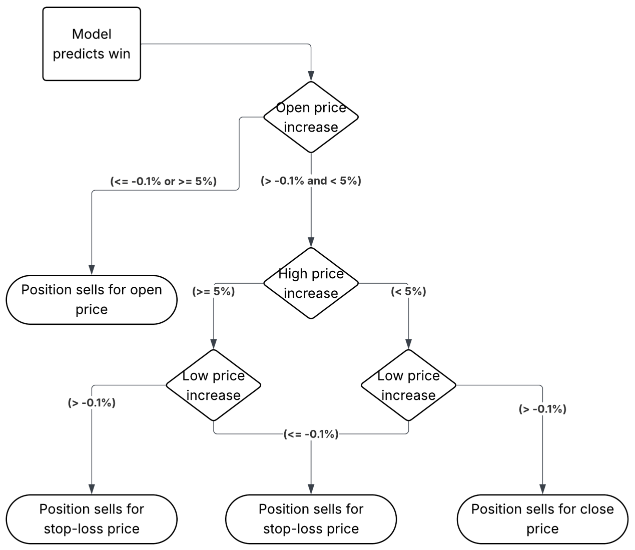

Fig. 2. Algorithm for determining price at which a position sells. The algorithm first checks to see if the position would sell at open. After this it examines mid-day prices to see if aim price or stop-loss price were hit. Note that if both of those occur, the outcome is ambiguous without more data and is given the stop-loss price in the interest of being conservative. If none of these occur, the stock sells at the close price.

B. Selecting the Size Penalty Coefficient

Using all of the methodology outlined in Section III-A, 5 models were created with different size penalty coefficients. The values ranged from 2×10^{-5} to 5×10^{-4}. The value 3×10^{-4} was selected because it had the highest precision and return when tested.

C. Test Methods

With the size penalty coefficient λ selected, the model was then tested to see how it performed against an ablated model. Using the general methods, 8 models were trained with the size penalty coefficient $\lambda = 3 \times 10^{-4}$. Each one was trained on market data from all 1500 stocks from 2013 through 2014. The collection of these models will be referred to as Group A. For each of these models, a corresponding model was trained with a size penalty coefficient $\lambda = 0$. This is done in order to remove the term's effect on the loss entirely. Each corresponding model used the same random seed for the training data shuffle, so as to have complete parity in every part of the training process except the size penalty. This collection of models will be referred to as Group B.

D. Test Results

After testing all of the models on data from the year 2016, it was found that 5 of the 8 penalized models from Group A had a higher return than their Group B counterparts. The distributions of the total return at the end of the year for both model groups can be seen in "Fig. 3".

As indicated in "Fig. 3", both the median and mean return were higher in Group A compared to Group B. The 2 groups had other performance differences. As intended, penalizing higher predictions lowered the positive rates for the agents using Group A models. "Fig. 4" illustrates this.

The higher selectivity exhibited in Group A is also correlated with a higher precision as demonstrated in "Fig. 5".

These results indicate that the logit size penalty is improving model performance; however, all of these results summarize the temporal dimension rather than exploring it. It is worth examining, as seen in "Fig. 6", how the performance changes over time.

This reveals that at many points during the evaluation period, the Group B models were performing better on average. Near the end of the year, the Group A models jump in performance may indicate that something about the market during the time period was favorable to the more selective models.

IV. Temporal Generalization Test

A. Methods

The temporal generalization test was conducted using all of the methods outlined previously in Section III-A. The goal was to conduct the same type of ablation test as in Section III-C, but to test on multiple datasets that span multiple time periods. For every consecutively 2 year period from 2013-2014 through

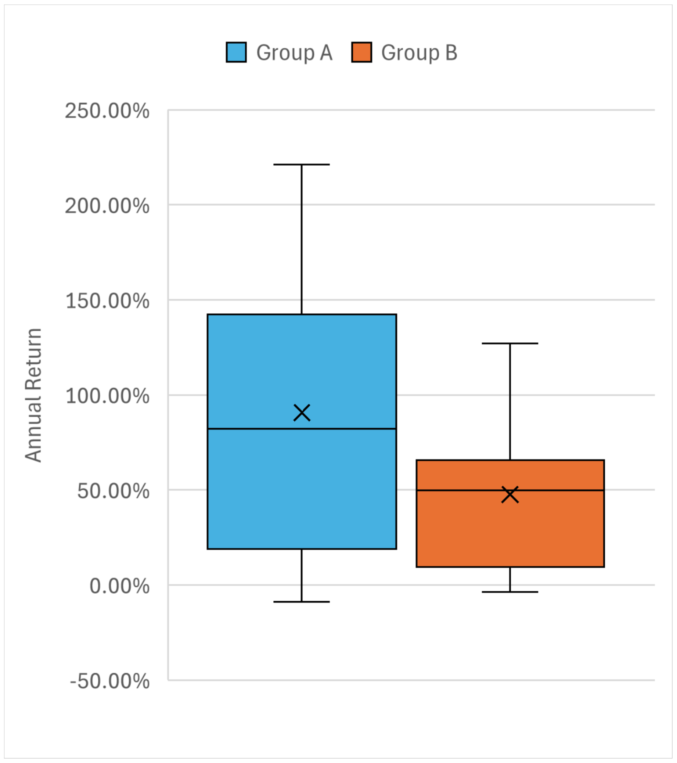

Fig. 3. Distributions of annual return from the year 2016 are shown for agents using penalized and un-penalized models in their evaluations on the data from the year 2016. Mean values are indicated with an X.

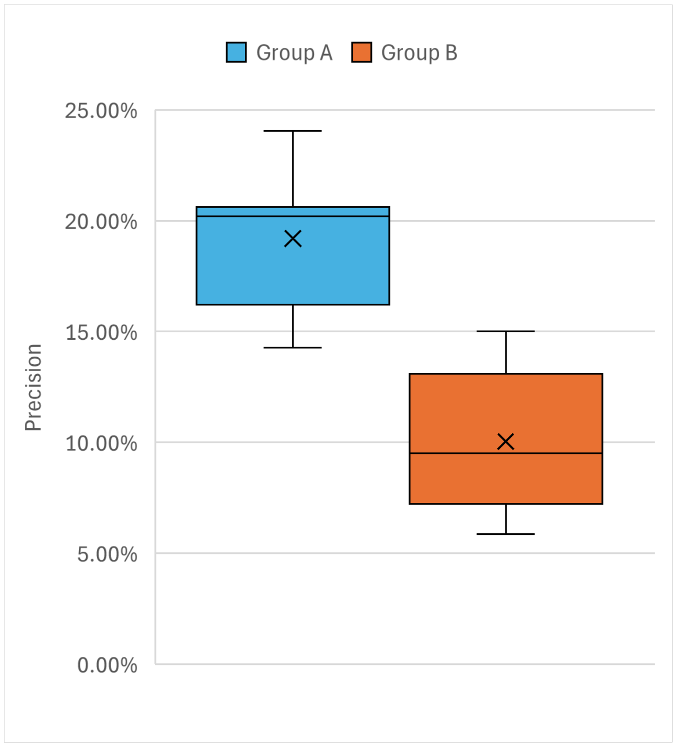

Fig. 5. Distributions of precision are shown for agents using penalized and un-penalized models in their evaluations on the data from the year 2016. Mean values are indicated with an X.

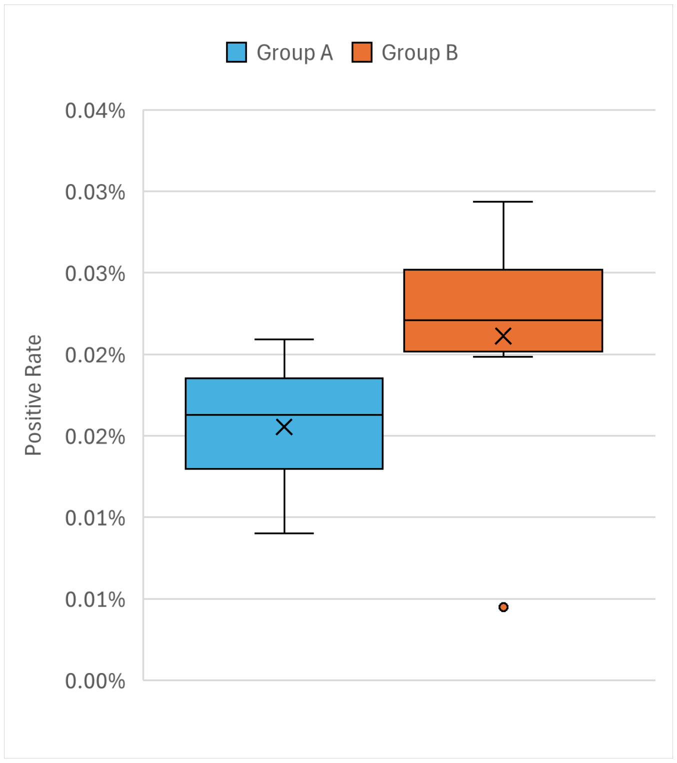

Fig. 4. Distributions of positive rates are shown for agents using penalized and un-penalized models in their evaluations on the data from the year 2016. Mean values are indicated with an X.

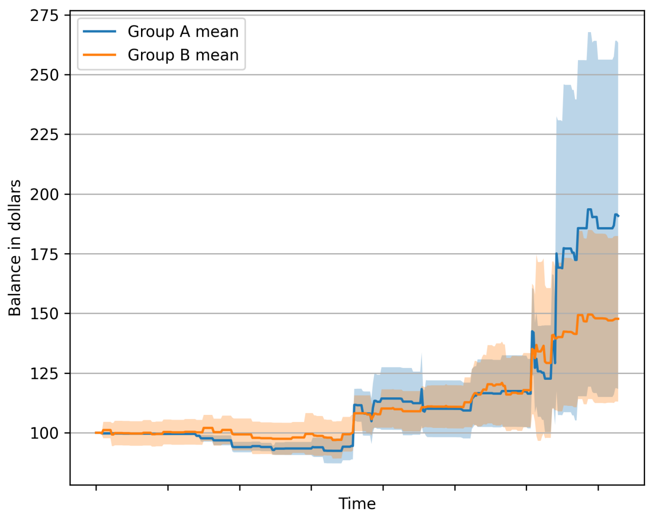

Fig. 6. The mean balances of the agents using Group A and Group B models are shown over the course of the evaluation year, 2016. The shaded area around each line represents the 95% confidence interval for the balance at that point.

2020-2021, a new model was trained on that data using the size penalty coefficient $\lambda = 3 \times 10^{-4}$. The collection of these models will be referenced as Group C. For each of these models, an ablated counterpart was created with identical training, except that the size penalty coefficient was set $\lambda = 0$. These models will be collectively referred to as Group D. All of the models in Group C and Group D had the random elements of their training seeded with the singular value that seeded training for the upper median models in both Group A and Group B in terms of annual return and precision. The models were tested on the 2nd year after the end of their training data, so as to avoid data leakage.

B. *Test Results*

Similar to the results in Section III-D, both the median and mean annual return were higher for the agents who used who were trained with penalties. This is shown in “Fig .7”.

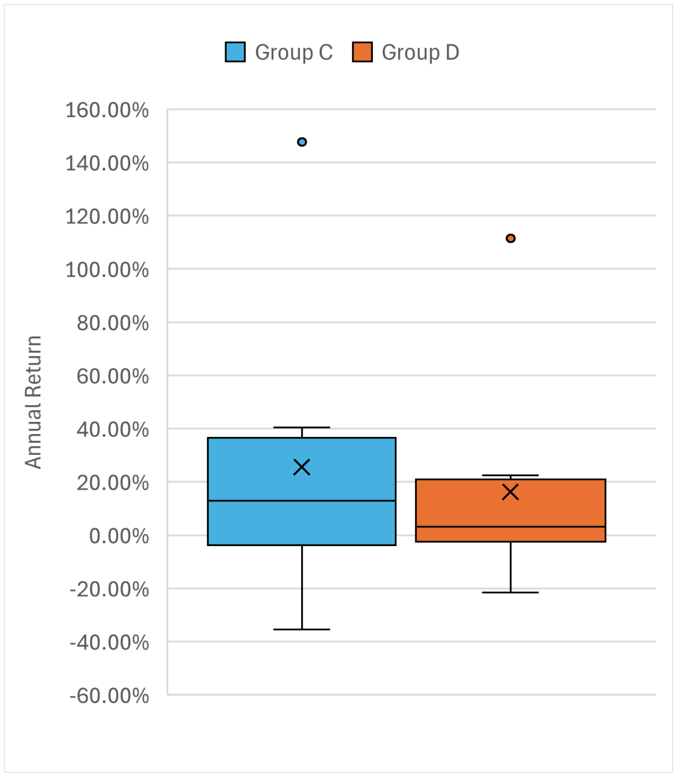

Fig. 7. Distributions of annual returns are shown for agents using the penalized and un-penalized models. Results are from evaluations ranging from 2016 through 2023. Mean values are indicated with an X.

Also similar to the previous results, the size-penalized models in Group C have a lower positive rate and a higher precision on average. “Fig .8” and “Fig .9” demonstrate this.

V. Results

The results from both tests indicate a performative advantage with using the size penalty loss terms during training. The mean annual return from using the models in Group A is 90.8% as opposed to the 47.7% from Group B. The values for both of these are larger and farther apart than the means for

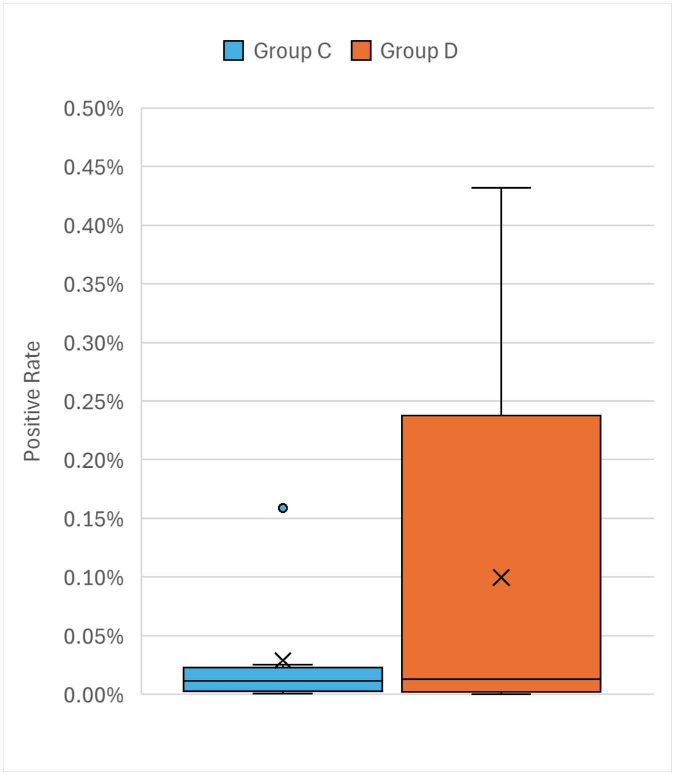

Fig. 8. Distributions of positive rates are shown for agents using penalized and un-penalized models. Results are from evaluations ranging from 2016 through 2023. Mean values are indicated with an X.

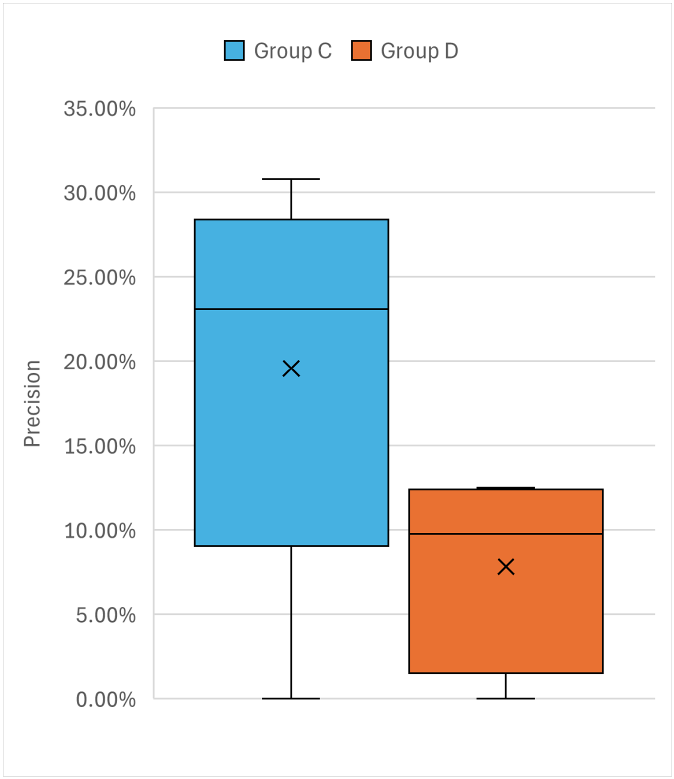

Fig. 9. Distributions of precision are shown for agents using penalized and un-penalized models. Results are from evaluations ranging from 2016 through 2023. Mean values are indicated with an X.

Groups C and D, which are 25.6% and 16.3% respectively. In both tests, the agents using size penalized models were seen to have a higher increase on average for each true positive. These findings are illustrated in “Fig. 10” and “Fig. 11”.

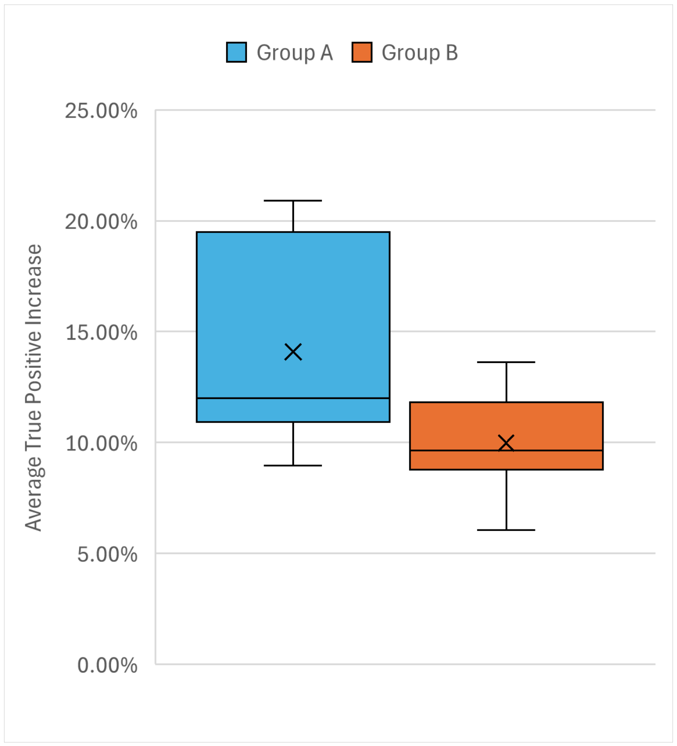

Fig. 10. Distributions of precision are shown for agents using penalized and un-penalized models from their evaluation on data from the year 2016. Results are from evaluations ranging from 2016 through 2023. Mean values are indicated with an X.

The fact that the average true positive increases became more similar between the 2 groups in the temporal generalization test could help to explain why the average annual return became more similar for them as well. The overall magnitude of these trade increases also suggests that the agents are making most of their profits when they sell stocks for an opening price which exceeds their 5% goal. Otherwise the average increase for a successful trade would be expected to be much closer to 5%.

VI. Conclusion

VII. Limitations

References

[1] M. Sun, Q. Li, and P. Lin, “Short-term stock price forecasting based on an SVD-LSTM model,” Intelligent Automation & Soft Computing, vol. 28, no. 2, pp. 369–378, Feb. 2021, doi: https://doi.org/10.32604/iasc.2021.014962.

[2] Q. Li, N. Kamaruddin, S. S. Yuhaniz, and H. A. A. Al-Jaifi, “Forecasting stock prices changes using long-short term memory neural network with symbolic genetic programming,” Scientific Reports, vol. 14, no. 1, p. 422, Jan. 2024, doi: https://doi.org/10.1038/s41598-023-50783-0.

[3] H. Jiang, X. Hu, and H. Jia, “Penalized logistic regressions with technical indicators predict up and down trends,” Soft Computing, Aug. 2022, doi: https://doi.org/10.1007/s00500-022-07404-1.

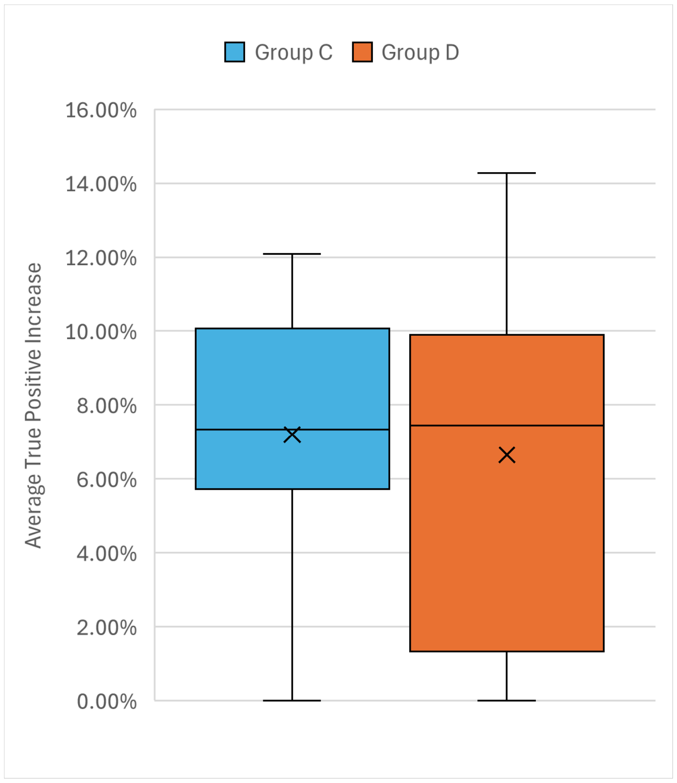

Fig. 11. Distributions of precision are shown for agents using penalized and un-penalized models. Results are from evaluations ranging from 2016 through 2023. Mean values are indicated with an X.