Leveraging Tree-Based Regression Models for Intraday Trading Strategy

Final Paper - Seminar Data Science for Marketing Analytics

Gergely A. Marias

07.01.2024

Disclaimer:

The content presented in this paper is for informational purposes only and should not be considered as financial advice. Any ideas, solutions, or strategies discussed are purely hypothetical. I am not a financial expert, and the information provided does not constitute endorsement or recommendation for any specific trading or investment activities.
Readers are strongly advised to conduct thorough research and seek advice from qualified financial professionals before making any investment decisions. The strategies discussed in this paper are not guaranteed to yield positive results, and past performance is not indicative of future outcomes. By reading this paper, the readers acknowledge that any decision to apply the discussed strategies is made at their own discretion and risk. I neither encourage nor support anyone to replicate the strategies outlined in this paper, and any such attempts are undertaken entirely on the reader's own will.

INTRODUCTION

In today's economic climate, where inflation is on the rise, I find it crucial to explore additional investment opportunities and alternative ways to generate income. The stock market stands out as a viable option for both long-term investors and short-term traders. My focus in this paper is on day trading only, a strategy characterized by closing all positions before the market's daily conclusion.

Therefore, I formulate this research question:

What is the potential financial yield associated with day trading TESLA stock, utilizing XGBoosting regression models' predictions?

Let me break down the trading strategy I am examining. First, I am dividing the trading day into two equal segments. I analyze the NY Stock Exchange, which operates from 9:30 AM to 4:00 PM (UTC-5) thus, I divide this 6 hours and 30 minutes. The first 3 hours and 15 minutes provide the data that my models will use to predict the remaining trading period. The strategy involves initiating a buy position every day exactly at 12:45 PM and setting a target price for Take Profit based on the models' predictions. The predictions will be an expected price movement in percentage for the rest of the day, from 12:45 PM up until 15:59 PM. For instance, at 12:45 PM TESLA price is 100$ and my model says 1.2% increase on the price. This means I buy 10 stocks for 100$ and set a Take Profit at 101.2$. If the price reaches this target, it automatically closes my position, thus it sells 10 stocks, each for 101.2$ and makes in total 10.2$ profit on that single trade. I do not count on commission fees during my research, and I use hypothetical 1000$ on every single trade.

To implement this approach, I use the *minute-opening* prices at 12:45 PM to start the buy position. The algorithm predicts the maximum price movements for the rest of the day and I set this as my predicted target price where I automatically close my position. However, if the price does not reach my predicted target price, then the position will automatically close at the end of the day, on the average of the *minute-highest and -lowest* prices for that last minute, at 15:59 PM. This strategy is tailored for stocks with high volatility, where significant price movements create patterns that my predictive models hopefully can leverage. For this analysis, I have chosen TESLA (TSLA: NYSE) as my subject stock, known for its notable volatility within the market.

DATA

I have delved into the minute-by-minute data of TESLA, spanning from January 2, 2014, to December 22, 2023, covering a total of 2520 trading days. After cleaning up the data, I managed to retain 2492 trading days, which is noteworthy as it includes almost all the days with complete minute data. (without any NAs)

The focus of my observations revolves around trading days, and the features encompass all the minutes from 9:00 AM up until 12:45 PM for each day. This provides me with 2492 observations and 195 features (195 minutes) to work with for building models.

Breaking it down, I have categorized my data into three sets:

1. The first set includes all the opening prices for specific minutes.
2. The second set includes all the highest prices for those minutes.
3. The third set includes all the lowest prices for those minutes.

When it comes to building my model, I use only the opening prices. To test out my strategy by closing at the end of the day, I turn to the highest and lowest price points. Here is a peek at the beginning of my dataset that contains all the minute opening prices: (Dataset 1.)

Table 1: First 5 observations and first 9 features from my Dataset 1.

Date	X93000	X93100	X93200	X93300	X93400	X93500	X93600	X93700
2014.01.02	149.8	149.15	148.68	148.4	148.51	148.445	147.81	148.07
2014.01.03	150.38	150.2999	150.63	151.364	150.52	150.95	152.17	151.73
2014.01.06	150.18	150.28	150.236	149.85	149.7	149	149	148.77
2014.01.07	147.62	147.61	147.97	148.22	147.52	147.844	148.2	148.1655

*X93000 = 9:30AM

However, if somebody looks at the TESLA's price chart on 02.01.2014. will see an opening price: 9.99$ and not 149.8$ as it is shown in my dataset. The reason behind it is that TESLA completed a 5-for-1 stock split in 2020 and a 3-for-1 split in 2022. So, if we do the calculation, we can get the right price:

149.8$ / 5 / 3 = 9.99$

Because of these stock splits, I realized it would make no sense to train my model on the actual prices when I would like to use tree-based models, but the price differences in percentage could be interesting. Thus, I cannot directly use the data in this form. I am interested in the percentage difference between one minute and the next, understanding how the price fluctuates between two consecutive minutes. To capture this, I have created a new dataset named "FirstHalf" which contains the date and the first 195 price differences in percentage from the first half of the day:

Table 2: First 5 observations and first 8 features from my transformed dataset.

Date	X1	X2	X3	X4	X5	X6	X7
2014.01.02	-0.43391	-0.31512	-0.18832	0.074124	-0.04377	-0.42777	0.175901
2014.01.03	-0.05327	0.219628	0.487287	-0.5576	0.285676	0.808215	-0.28915
2014.01.06	0.066587	-0.02928	-0.25693	-0.1001	-0.4676	0	-0.15436
2014.01.07	-0.00677	0.243886	0.168953	-0.47227	0.219631	0.240794	-0.02328

(FirstHalf$X1 = Dataset1$X93100 / Dataset1$X93000) %

The target variable following this idea which is specifically aimed at determining the maximum yield achievable (maximum price difference in percentage) for the remaining trading duration by initiating a buy position at 12:45 PM. This figure is obtained by calculating the maximum price in the second half of a trading day divided by the price at 12:45 PM.

(Target variable = maximum price from 12:46 PM to 15:59 PM / price at 12:45 PM) %

In summary, the training dataset contains 2492 observations, and each comprises 195 data points representing percentage differences, plus the target variable indicating the maximum attainable yield when initiating a buy position at 12:45 PM.

DATA PREPARATION

The .CSV file I imported is not organized correctly, with each row representing minute-by-minute data. To address this, I must rearrange the dataset so that each row encompasses an entire day, consisting of 390 minutes along with the corresponding date. Additionally, as previously stated, I use only the first half of each day, and the data points represent minute-by-minute percentage movements rather than the actual prices. Finally, I prefer to avoid random splitting when allocating the training and test sets. Instead, I intend to investigate the scenario of training my model on the data from '14 to '22 and then evaluating its performance over the entire year of 2023, simulating a real-world scenario where the model is trained on historical data and tested on a subsequent period.

METHODS

For this analysis, the machine learning algorithm that I have selected is Extreme Gradient Boosting. Moreover, I use Simple Linear regression as a benchmark model. The decision to utilize tree-based model is grounded in their capability to handle non-linear relationships effectively, a crucial aspect given that price movements in this context do not follow a linear pattern.

XGBOOSTING

Extreme Gradient Boosting is a supervised algorithm. It belongs to the gradient boosting family, which builds an ensemble of weak learners sequentially, improving model performance at each step.

In simpler terms, XGBoost constructs a decision tree to make predictions, then examines the errors made by the tree. It learns from these mistakes, identifies where the initial model went wrong, corrects those errors, and constructs another tree. This iterative process repeats, with each cycle enhancing its performance by incorporating lessons learned from the previous models.

I include MAE, RMSE, and R-squared as standard evaluation metrics. Additionally, for the research aspect, I introduce a new self-made metric called "Profit," tailored to assess the effectiveness of the targeted strategy. As previously mentioned, this metric involves testing model predictions throughout the year 2023, initiating hypothetical buy positions with a value of $1000 every day. At the end of each day, the resulting amount from these predictions is calculated, leading to the inclusion of a fourth metric named "Profit" for each model. I consider the "Profit" metric crucial because while the conventional metrics assess models from a statistical viewpoint, the essence of this research lies in the financial aspect. The primary focus is on determining the potential financial yield, aligning with the research question posed.

To obtain the best RMSE from the learning iteration I do a grid search to tune the hyperparameters. I tuned maximum depth (max_depth), this parameter in XGBoost controls the maximum depth of a tree in the ensemble. Deeper trees can memorize the training data, potentially resulting in overfitting. By adjusting this, the model can identify an optimal depth that mitigates the risk of the model becoming excessively complex and overfitting the training data. Further, I also tuned the learning rate (eta), this parameter determines the step size at each iteration of the gradient descent optimization process. A smaller learning rate requires smaller steps, which can help the algorithm converge more smoothly. Setting a too high learning rate may cause the model to overshoot the minimum of the loss function, leading to instability and slower convergence. Tuning eta helps in finding an appropriate balance between convergence speed and stability. Next is gamma, it is a regularization parameter that controls whether a node in the tree will split based on the expected reduction in loss after the split. By tuning gamma, you can control the complexity of the trees, preventing them from growing too deep and overfitting the training data. It continued with subsample; proper tuning of subsample contributes to better generalization performance. Using a subset of the data for each tree promotes robustness and helps the model generalize well to unseen data, especially in the presence of noise or outliers. The last hyperparameter is the number of boosting rounds, (nrounds), this determines how many decision trees are sequentially added to the ensemble. Tuning this parameter allows to find the right balance in model complexity. Too few rounds may result in underfitting, while too many rounds may lead to overfitting. I decided not to prune colsample_bytree, and max_weight_child because the domain knowledge suggests that variations in feature sampling or node weights are may not useful for this problem. Because in my opinion, randomly selecting minutes (features) is not ideal, nor determine the minimum number of instances needed to be in each node.

LINEAR REGRESSION

In benchmarking a tree-based model, linear regression can serve as a baseline. It provides a straightforward comparison for evaluating the performance of more complex models. It is computationally less intensive compared to decision trees or XGBoosting, making it an efficient choice for establishing a baseline model.

ANALYSES & RESULTS

During my analyses I have built 3 models:

1. XGBoosting with default parameters
   - For this model, I choose not to define specific parameters, only indicating the number of rounds of iteration, which is commonly set at 100.

2. XGBoosting with tuned parameters after grid search:
   - The grid search that I have computed:

     ```
     objective = "reg:squarederror",
     booster = "gbtree",
     eval_metric = "rmse",
     max_depth = c(6,9,12,15),
     eta = c(0.01,0.05,0.09,0.13,0.16),
     gamma = c(1,2),
     subsample = c(0.8,0.9,1),
     colsample_bytree = c(1),
     nrounds = c(100,200,300)
     ```

   - The results of the grid search:

     ```
     objective = "reg:squarederror",
     booster = "gbtree",
     eval_metric = "rmse",
     max_depth = 12,
     eta = 0.01,
     gamma = 1,
     subsample = 0.8,
     colsample_bytree = 1
     ```

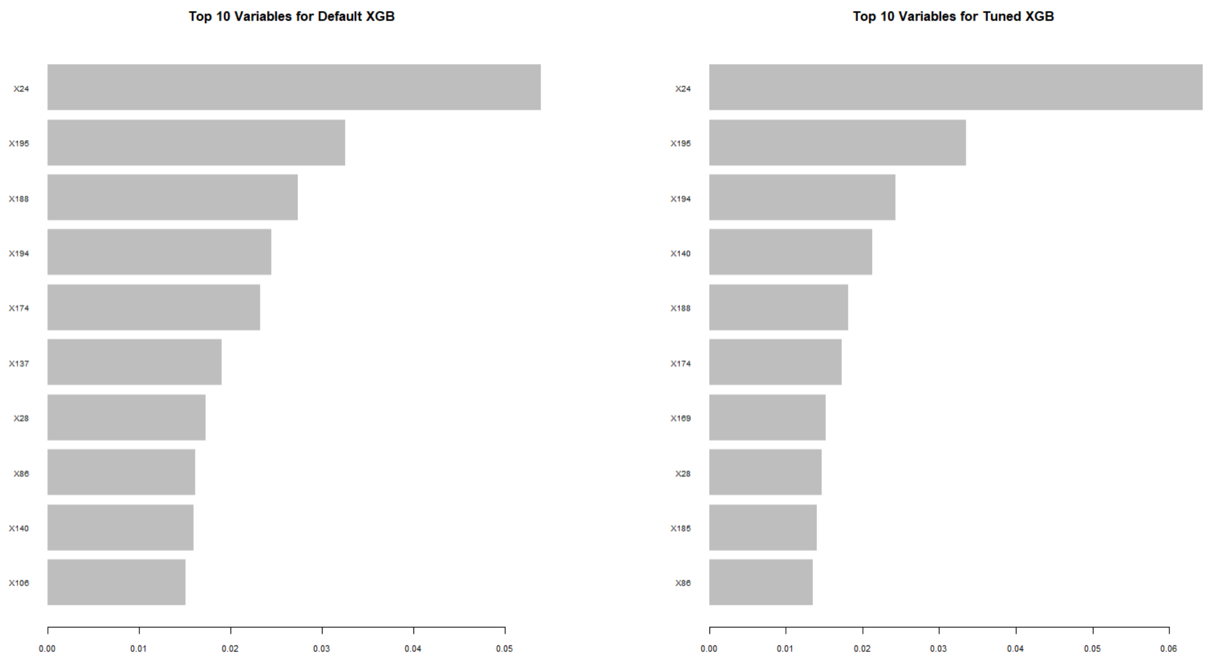
3. Simple Linear regression

Table 3: Results of the three models.

	MAE	RMSE	R2
Linear	0.72	0.96	-0.26
Default Boosting	0.693	0.92	-0.15
Tuned Boosting	0.630	0.83	0.05

Looking at Table 3. statistically, we can see improvements when moving from linear regression to boosting and then fine-tuning the hyperparameters. However, my research question focuses on the actual financial gains we could get from using predictions of XGBoosting regression models. Before I introduce the so-called metric "Profit", let me continue to investigate the results of the models.

Figure 1: Variable importance for the default and the tuned model (top 10).



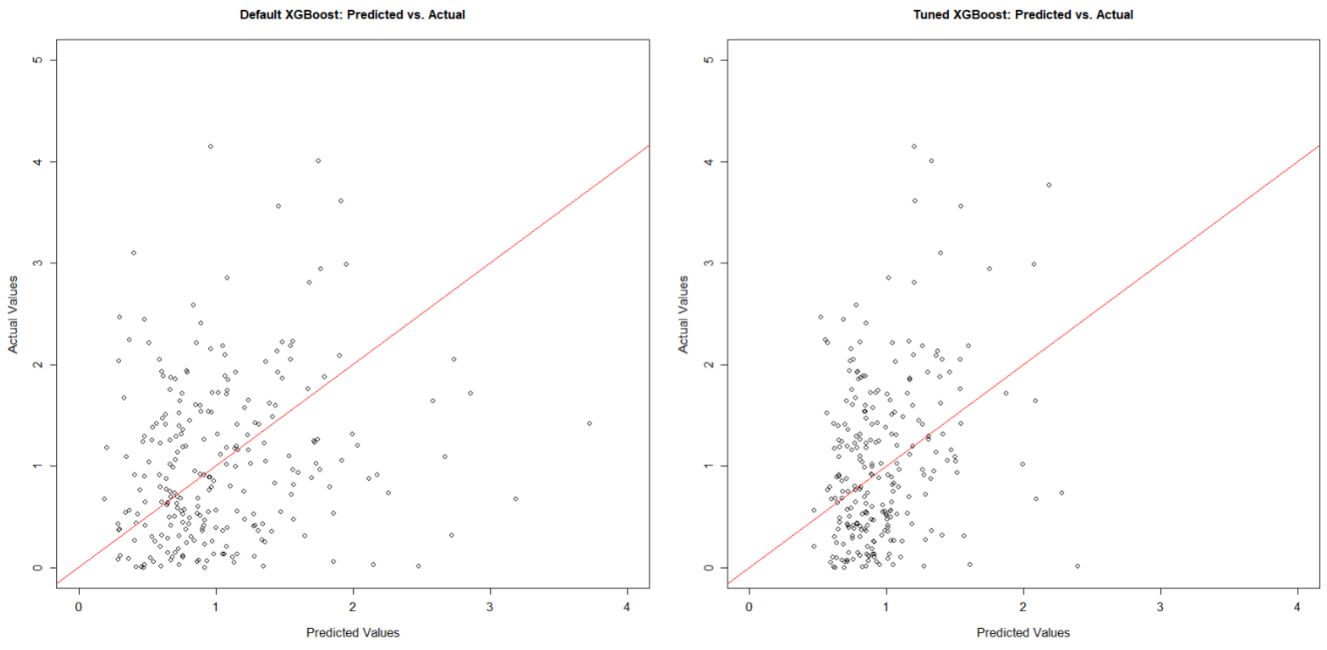
Examining the variable importance plots reveals that, for both the default and tuned models, the 24th and 195th minutes stand out as the most crucial, corresponding to 9:54 AM and 12:45 PM, respectively. Interestingly, most variables align between the two plots, except for X137, X106, X169, and X185, which differ. The remaining variables remain consistent but in a varied order. In my opinion this observation could hold significance for financial analysts, indicating that both models recognize the importance of specific minutes, suggesting noteworthy events during those time points.

Table 4: Same variables in different order

Default model	Tuned model
X24	X24
X195	X195
X188	X194
X194	X140
X174	X188
X28	X174
X86	X28
X140	X86

Continue with Figure 2. which displays the comparison between predicted and actual values for both the Default XGBoost model and the Tuned XGBoost model. The common red line represents the ideal scenario where predicted values perfectly match actual values.

Figure 2: Predicted versus Actual values.



To analyze these plots, I would like to summarize my strategy again. Strategy in a nutshell:

1. Every trading day, open a buy position with TESLA stock for a value of $1000, precisely at 12:45 PM, if the predicted value for that day is positive.
2. Set Take Profit at buying price multiplied by the prediction.
3. If the price reaches that price, it closes the position automatically, but if the price does not reach that price during the day, it closes the position at 15:59PM.

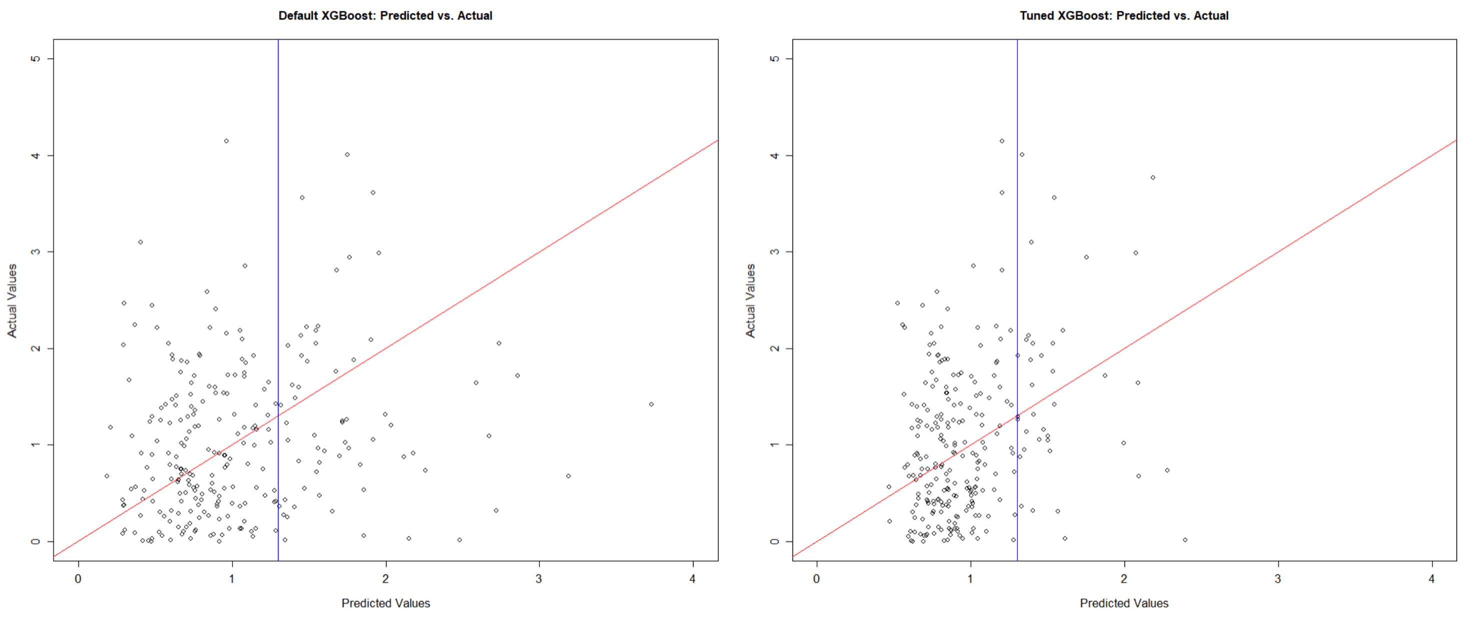
From this we can understand that strategy could only reach the target price if the black dots are above the red line. For the default model 47.2% of the dots are above the red line, thus 47.2% of the trades successfully reached the target price. For the Tuned model 48% of trades are successful. However, Table 5. surprised me.

Table 5: The results of relying on the predictions and following this strategy for 250 trading days.

	MAE	RMSE	R2	Profit	Yield
Linear	0.72	0.96	-0.26	112$	11.2%
Default Boosting	0.693	0.92	-0.15	263$	26.3%
Tuned Boosting	0.630	0.83	0.05	183$	18.3%

Using linear regression alone, I achieved an 11.2% yield in 2023 by following this strategy. Surprisingly, the default model, despite having less favorable statistical metrics compared to the tuned model, resulted in a higher yield of 26.3%. What is the reason behind it? Taking a closer look at Figure 3. could reveal the answer for this question.

Figure 3: Predicted versus Actual values with a blue line at value 1.3.



In the left plot, there are 22 black dots in the right upper corner (above the red line and to the right of the blue line), while on the right plot, there are 18 black dots in the right upper corner. This indicates that the default model successfully predicted on 6 more days where the predicted yield exceeded 1.3% (as indicated by the positions of the blue lines). Furthermore, as indicated in Table 5, there is an $80 difference in profits. Notably, the default model did 6 more successful trades, potentially resulting in an additional $80 in gains. Nevertheless, it is important to recognize that this is likely not the only factor contributing to the observed outcome.

CONCLUSION

To draw a conclusion, it is noteworthy that within the specific time range from December 22, 2022, to December 22, 2023, all three regression models demonstrated positive yields. Specifically, linear regression achieved a 11.2% yield, while the two boosting models achieved 18.3% and 26.3%, with the non-tuned model outperforming. It is important to emphasize the disclaimer: "The strategies discussed in this paper are not guaranteed to yield positive results, and past performance is not indicative of future outcomes."

Throughout this research, it became apparent that relying solely on statistical metrics does not always align with the objectives of the research question. Moreover, the influence of external factors on predictions cannot be fully understated. Also, the task of defining a trading strategy turned out to be more challenging than anticipated, underscoring the necessity for continuous refinement.

The process highlighted the importance of having a backup plan (Plan B) for closing positions, especially in situations where the target price seems unattainable during a given day. These considerations offer valuable insights for refining both the models and trading strategies, paving the way for enhanced control and effectiveness in future aspirations.

Furthermore, the results shed light on the significance of 9:54 AM as a pivotal time, encouraging reflections on whether shorter time periods could yield more insightful outcomes. This discovery opens avenues for exploration, allowing for a more nuanced understanding of the models' behavior.

REFRENCES

Chapman, C., & Feit, E. M. (2015). R for Marketing Research and Analytics (pp. 327-336).

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning (pp. 303-321).

GeeksforGeeks. (2023, 6 February). XGBoost. https://www.geeksforgeeks.org/xgboost/

Jason Brownlee. (2016, August 16). A Gentle Introduction to XGBoost for Applied Machine Learning.

Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-appliedmachine-learning/

Machine Learning with R: A Complete Guide to Gradient Boosting and XGBoost | R-bloggers. (2021,

February 18). R-Bloggers. https://www.r-bloggers.com/2021/02/machine-learning-with-r-a-completeguide-to-gradient-boosting-and-xgboost/#google_vignette

Molnar, C. (2019). Interpretable Machine Learning: A Guide for Making Black Box Models Explainable(pp. 79-124).