Betting on Machine Learning: A machine-learning Reddit stock picker

Alexander De Laurentiis John Lin Kaylee Acres Yasseen Semlali

December 7, 2021

Abstract

The popularity of do-it-yourself investing has exploded in recent years due to the advent of mobile-first trading platforms and online brokerages such as Wealthsimple, Robinhood and Webull, among many others. However, compared to large institutions, non-professional (retail) investors do not have the time or the resources to perform due diligence on every stock on the market. Here, we sought to crowdsource due diligence on various stocks by gathering posts from an online aggregator, Reddit. From these posts, we preprocessed them to extract a ticker symbol, the sentiment of a post, and a performance ratio, and generated a dataset to train classification algorithms to determine whether or not a stock is worth buying. We aimed to level the playing field between the individual investor and large institutions, employing the wisdom of the crowd to generate predictions on future investment opportunities.

1 Introduction

It is no secret that do-it-yourself (DIY) investing has become more popular in the past years, especially among the younger generations. In an effort to out-perform the stock market, DIY investors laud the hands-on approach in growing their wealth. However, despite the growing popularity of retail investing, retail investors still cannot compete with the financial and corporal power of large investing entities, who employ thousands of people and use speedy algorithms to eke out even the slightest profit. Unlike these large entities, individual investors have neither the time nor the resources to perform due diligence on every stock on the market. Fortunately, what individual investors do have is access to multiple different online forums, where users can discuss investing strategies and the peculiarities of various stocks.

One of the largest discussion forums on the internet is Reddit, which contains thousands of sub-forums, or "subreddits", that cater to every user's whim and fantasy, including investing. Of these investing subreddits, two subreddits, r/wallstreetbets and r/stocks, stand out as the most popular and active. However, since Reddit by nature is an informal public forum, where users can post and comment in anonymity, investors looking to find a competitive edge may have a hard time sifting for "gold" amongst the thousands of comments posted daily. Herein, we crowdsource due diligence for the retail investor on Reddit, and discuss aggregating Reddit posts, analyzing their sentiment, and using the derived sentimental score to predict the performance of a mentioned stock on the r/wallstreetbets and r/stocks subreddits.

2 Methodology

2.1 Data Retrieval

To begin, the top 1000 posts of each day from two subreddits, r/wallstreetbets and r/stocks, were

retrieved since 2014. The official Reddit API was unable to be used, since date ranges were not supported, so we opted for the pushshift.io API. Beginning from January 1, 2014 to October 31, 2021, the day's top 1000 posts were parsed into a JSON file.

Next, the Python Reddit API Wrapper (PRAW) was used to retrieve comments from each submission. For each comment, the following were retrieved:

- Submission time
- Text of comment, parent comment, and submission
- Number of upvotes of comment, parent comment, and submission
- Number of replies
- Submission upvote vs. Parent comment upvote ratio
- Whether or not it is a top level comment

Next, stock data was retrieved. We created a custom dataset based on the historical stock market dataset by Boris Marjanovic on Kaggle [1]. Tickers that could be misconstrued as common words were also removed, such as $WSB and $HAS, by retrieving all the most frequently mentioned stocks and selecting for those that had the highest likelihood of producing false positives. We used the Yahoo Finance (yfinance) API to retrieve company and ticker names, and populate our dataset with historical prices. For example, a post mentioning Microsoft may refer to it by both its name or its ticker symbol, $MSFT. Our own custom dataset is therefore filled with a reduced list of tickers and their associated company names, and represents their historical pricing from 2014 to the present.

Finally, the Reddit comment data and stock data were combined by parsing through each comment and matching them with mentioned ticker or company names. We generated a sentimental score using the a pretrained sentiment analysis model, VADER. For true y-values, 6 metrics were generated for next-day, next-week, and 4-week stock performance ratios, each classified to `is_buy`, which represents a 2% stock gain, or to `is_strong_buy`, which represents a 5% stock gain. An example of our cleaned dataset (transposed) is seen in table 1.

Table 1: *A transposed example of one row of our cleaned dataset.*

Column header	**row 1**
date	2014-01-02
ticker	ford
score_neg	0.057
score_neu	0.874
score_pos	0.069
score_compound	0.1383
parent_score_neg	0
parent_score_neu	1
parent_score_pos	0
parent_score_compound	0
score	3
parent_score	22
submission_score	22
is_root	1
submission_ratio	0.14
parent_ratio	0.14
num_replies	1
close	1.64999999976
day_increase	.06452
tomorrow_increase	0.9697
tomorrw_is_buy	0
tomorrow_is_strong_buy	0
week_increase	1.01212
week_is_buy	0
week_is_strong_buy	0
4week_close	1.00606
4week_is_buy	0
4week_is_strong_buy	0
volume	11939.39411
today_close	1.6499999976
tomorrow_close	1.60000000024

2.2 Preprocessing

2.2.1 Obtaining a sentimental score

Although humans are relatively adept at understanding the sentiment of written text, it is unfeasible to have a human rater classify thousands of Reddit posts on sentiment. It was therefore necessary to automate our sentimental analysis. One option that we considered was to hard code dictionaries of words with positive or negative emotion (Table 2), summing up each instance of a positively or negatively classified word to

Table 2: *An example of categorized words that would be hard coded to generate a sentimental score.*

Category(Emotion)	Examples	Score
Positive	love, like, excited, gain	+1
Negative	bad, loss, poor, tank, worst	−1

generate a raw sentimental score, then scaling the raw score into a range between -1 and +1. This approach would be similar to that used by the Linguistic Inquiry and Word Count (LIWC) program [2]. However, one drawback with this approach includes the possibility of introducing bias into the classification, as whoever hard codes the dictionaries has say in which words are classified as positive or negative. Another drawback is that even words in the same category have varying levels of intensity. For example, the statements "The weather is nice" and "The weather is amazing" convey different levels of meaning, something that is not considered in the LIWC program.

We decided to use the Valence Aware Dictionary for Sentiment Reasoning (VADER) API to analyze our text for sentiment [3]. Some of the benefits mentioned by Hutto and Gilbert in their accompanying article include generalizability across multiple domains, speed, and no training requirement. Compared to machine learning algorithms such as Naïve-Bayes, Max Entropy and SVMs, VADER outperforms them on generating sentiment scores on a sample of 4000 tweets (Fig. 1) [3]. Additionally, VADER does not require text preprocessing to generate a sentimental score. In fact, preprocessing text by making words all lowercase and removing punctuation effectively removes meaning of a given text and alters the sentimental score output.

```
Text: I am SO happy!,
Score: {'neg': 0.0,
        'neu': 0.335,
        'pos': 0.665,
        'compound': 0.7875}
Text: i am so happy,
Score: {'neg': 0.0,
        'neu': 0.388,
        'pos': 0.612,
        'compound': 0.6948}
```

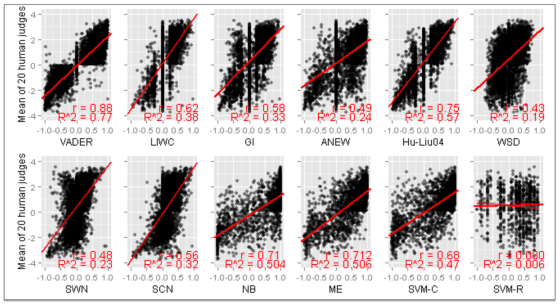

Figure 1: *Correlation of classification programs compared to 20 human judges. Note that VADER outperforms Naive-Bayes (NB), Maximum Entropy (ME) and Support Vector Machines (SVM) [3]. Other sentimental text analyzers: GI = General Inquirer, ANEW = Affective Norms for English Words, HuLiu04 opinion lexicon, WSD = Word-Sense Disambiguation, SWN = SentiWordNet, SCN = SenticNet.*

The speed at which VADER processes text and the ease-of-use in our project made it the top contender for analyzing the sentiment of Reddit posts during our preprocessing.

2.2.2 Generating Target Data

In our dataset, our target data was our `is_buy` or `is_strong_buy` classification based on the performance ratio of a stock over time. Performance ratio was calculated using `price_after_interval / current_price`. A stock was classified as `is_buy` if the performance ratio exceeded 1.02, and was classified as `is_strong_buy` if the performance ratio exceeded 1.05. We selected several different timeframes: next-day, next-week and 4-week performance. This allowed us to train six different models for both `is_buy` and `is_strong_buy` classifications across all three time frames.

2.3 Training

In our machine learning models, we used a date-restricted clean dataset from January 2, 2014 to March 29, 2019. We opted to train our models twice using scikit-learn's `model_selection` functions: once with a train_validation_test split of 60-20-20, and another with a TimeSeriesSplit. Using scikit-learn's StandardScaler, we also normalized our features, before training using a scikit-learn's DecisionTreeClassifier, RandomForestClassifier, LogisticRegression and SVM (support vector classifier). Plots were generated for each classfier's accuracy score against various hyperparameters that we explicitly initialized beforehand. This allowed us to determine the best classifier and hyperparameter setting. The objective for this training was to predict the performance of a stock the next day, the next week, and after four weeks based on the popularity and sentimental score of a Reddit post. Success was classified as a correct prediction of a stock's performance based on its popularity and sentimental score.

3 Results

3.1 Train_Test_Split

Logistic Regression was the fastest to train while Random Forest was the slowest to train. The best estimator was determined by the hyperparameter with the best validation score. Our best estimators were the RandomForestClassifier with `max_depth` of 50 for target `tomorrow_is_buy`, RandomForestClassifier with `max_depth` of 5 for target `week_is_buy`, and RandomForestClassifier with `max_depth` of 100 for target `4week_is_buy`.

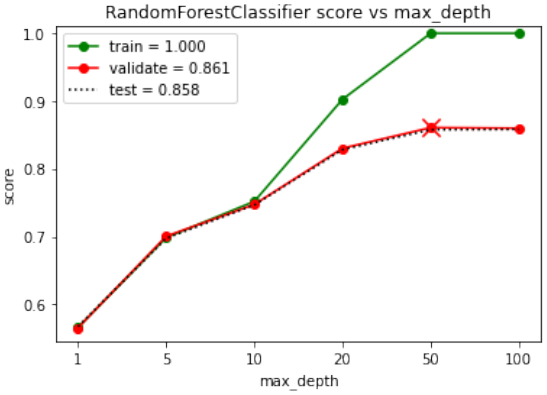

Figure 2: *Example of our best classifier for tomorrow_is_buy using train_test_split. All performance plots can be viewed in the included supplementary appendix.*

3.2 TimeSeriesSplit

Similar to `train_test_split`, we also performed a TimeSeriesSplit on our dataset, using older data to train our models, while using newer data to test our predictions. We observed that in all cases, our SVM was the worst performer and had one of the worst training times. This suggests that a SVM may not be suited for our objectives. In any case, the Decision Tree, Random Forest, and Logistic Regression classifiers performed similarly across multiple hyperparameters, with the Random Forest having better performance overall on classifying our test set compared to the other classifiers.

It is worth noting that for our decision trees, the best model for most of the y values only had a depth of 1. Upon further analysis, we found that this layer used objective values, such as the % change for that day, rather than the sentiment analysis score. This may indicate that it would be better to train a model based purely off the technical metrics of a stock, rather than sentiment analysis.

A notable trend in our results is that classifying a stock as a strong buy tended to be more accurate. This is likely because stocks with such high movement are preceded by a lot of positive

attention. The accuracy of our models also decreased as time increased. This is to be expected, since the further out a stock is projected, the harder it becomes to predict due to the inconsistency of the market.

4 Conclusion

From our results, we extrapolate that any model offering a greater than 50% accuracy of prediction may in the long run yield a profit for the user. We observed in our models that compared to TimeSeriesSplit, train-test-split may be more accurate since it does not account for evolution of certain stocks' performance over time, while in the time series, the test data comes after the training data. This may be more relevant in a real world use case and better test our models' capabilities. We are also aware that our sources, r/wallstreetbets and r/stocks, may not be optimal to generate stock recommendations, owing to their informal and temporal nature. Future iterations of this project should include other machine learning algorithms such as neural networks and Naive-Bayes. Overall, using machine learning to classify stock recommendations based on sentimental score of Reddit posts was a good learning opportunity, and highlighted some of the limitations of machine learning for our purposes. Finally, although some models were above average in correctly predicting a stock's performance, it is probably best to consider crowdsourcing Reddit posts as opinion-driven and not financial advice.

References

[1] B. Marjanovic, "Huge stock market dataset," 2017.

[2] J. Pennebaker, C. Chung, M. Ireland, A. Gonzales, and R. Booth, "The development and psychometric properties of liwc2007," *Austin, TX, LIWC. Net*, 2007.

[3] C. J. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *ICWSM*, 2014.

5 Appendix - Supplemental Figures

5.1 Figures for TrainTestSplit

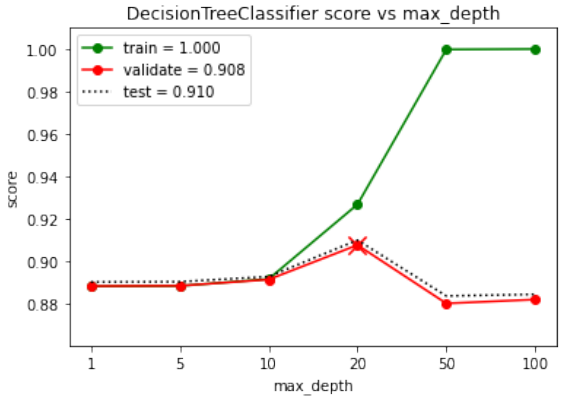

Figure 3: Decision Tree Classifier processed with target 4week_is_buy using train_test_split

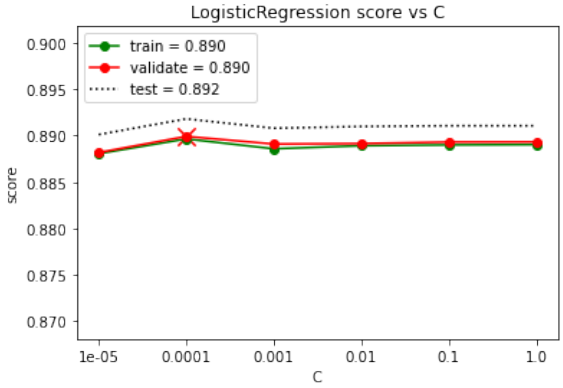

Figure 4: Logistic Regression Classifier processed with target 4week_is_buy using train_test_split

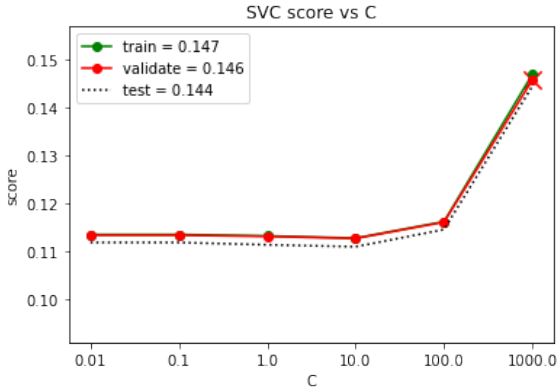

Figure 5: Support Vector Classifier processed with target 4week_is_buy using train_test_split

5.2 Figures for TimeSeriesSplit

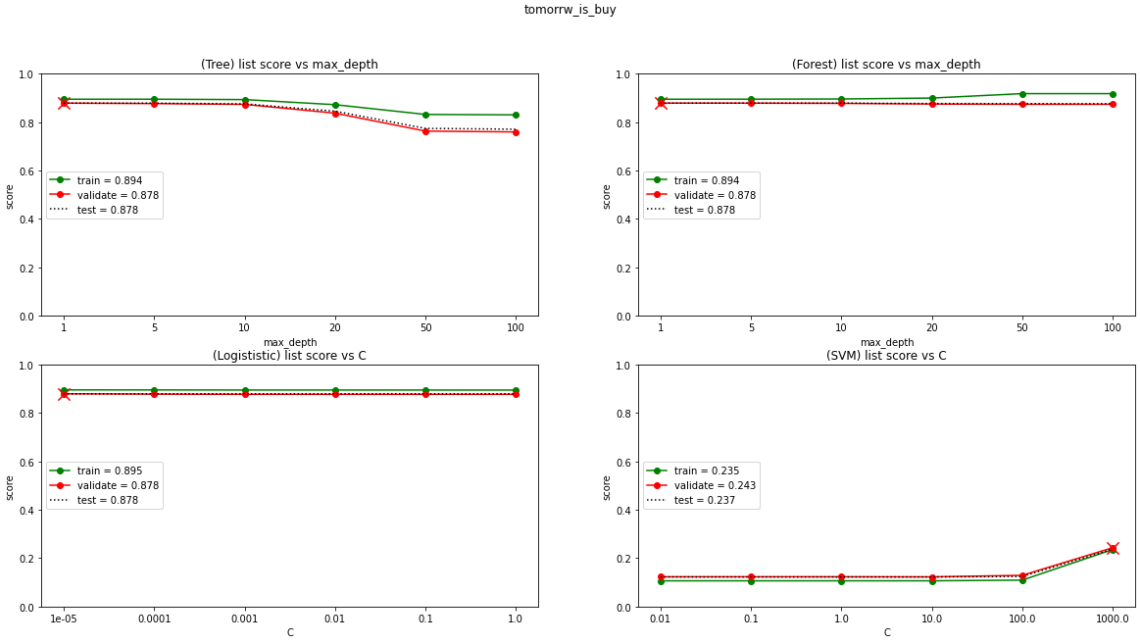

Figure 6: tomorrow_is_buy processed with dataset 1, out.csv

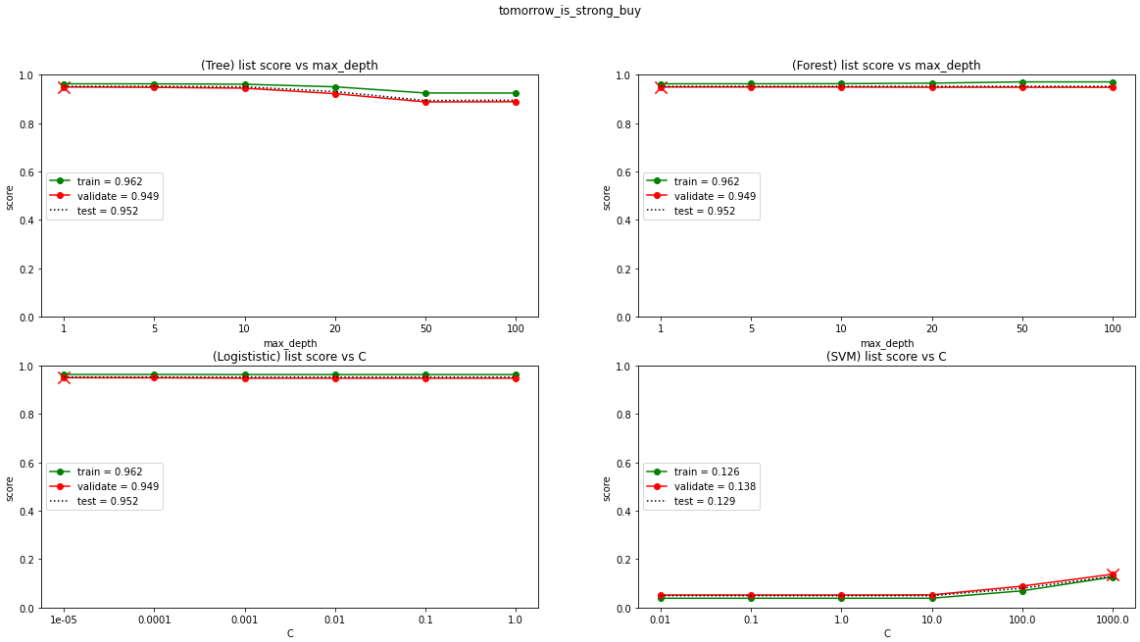

Figure 7: tomorrow_is_strong_buy processed with dataset 1, out.csv

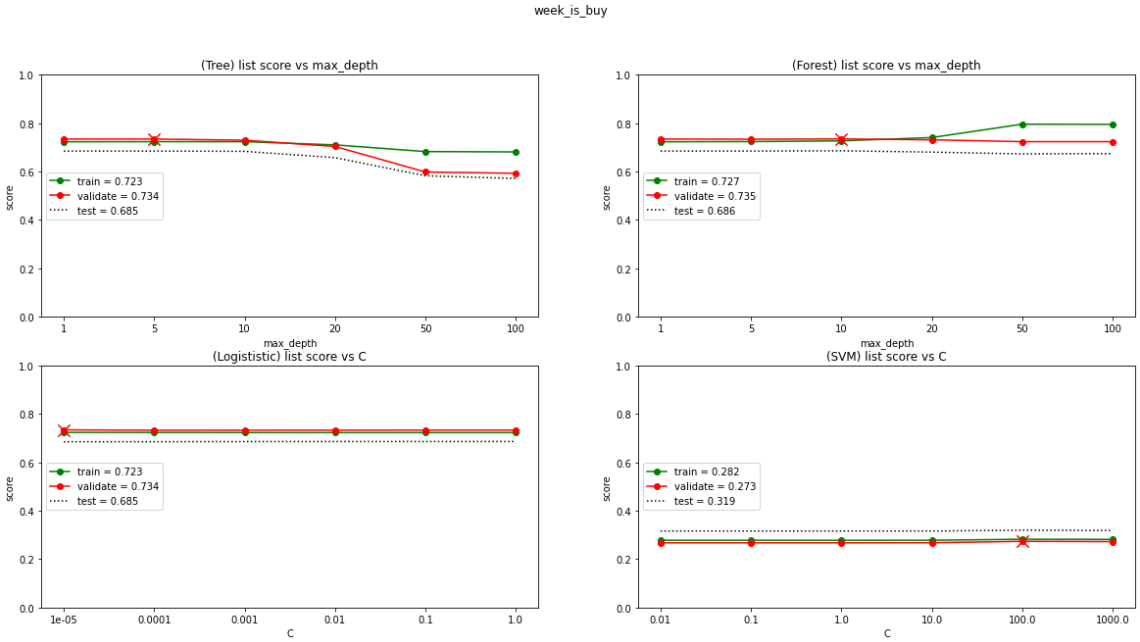

Figure 8: week_is_buy processed with dataset 1, out.csv

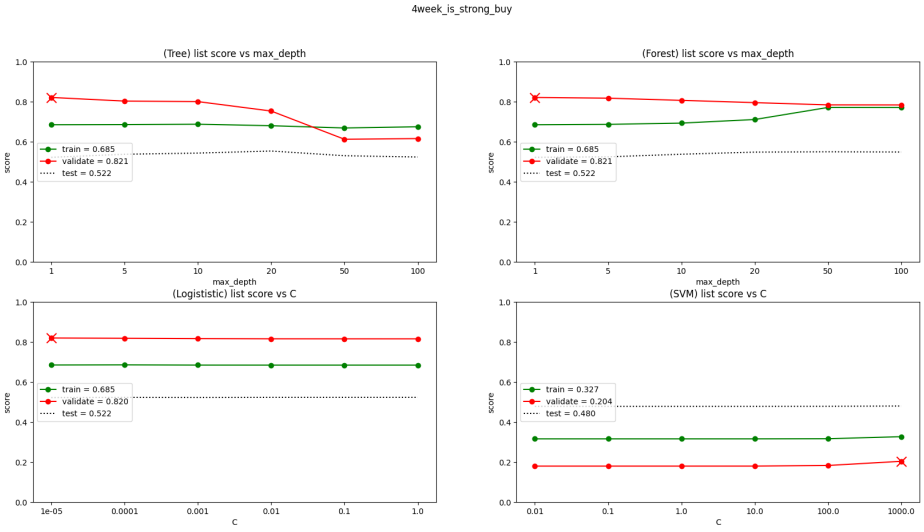

Figure 9: 4week_is_strong_buy processed with dataset 1, out.csv

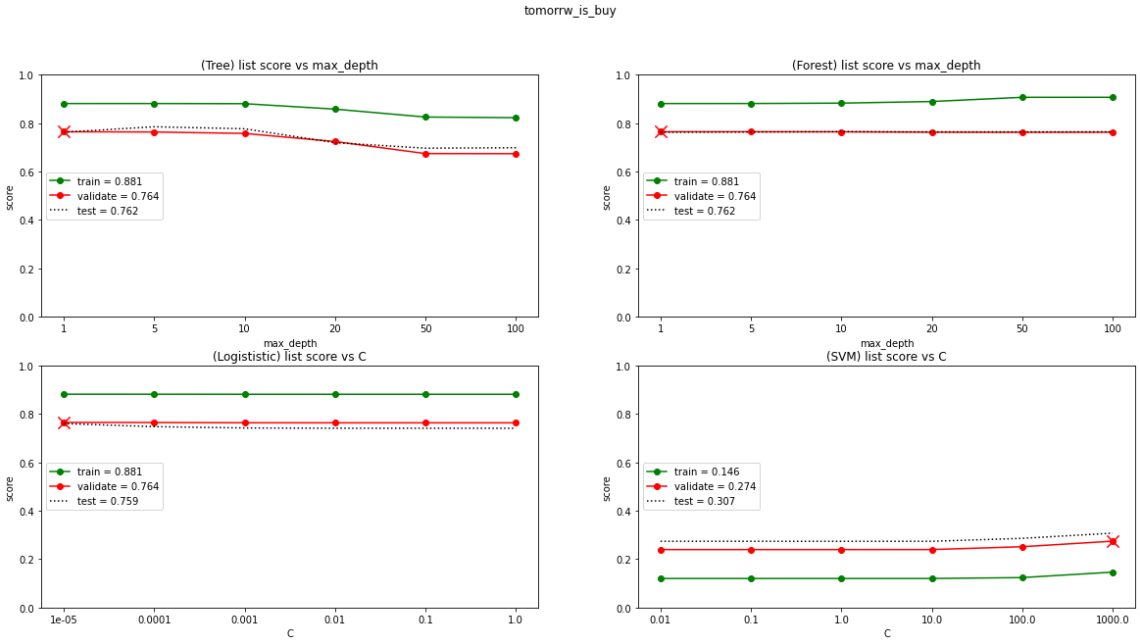

Figure 10: tomorrow_is_buy processed with dataset 2, out2.csv

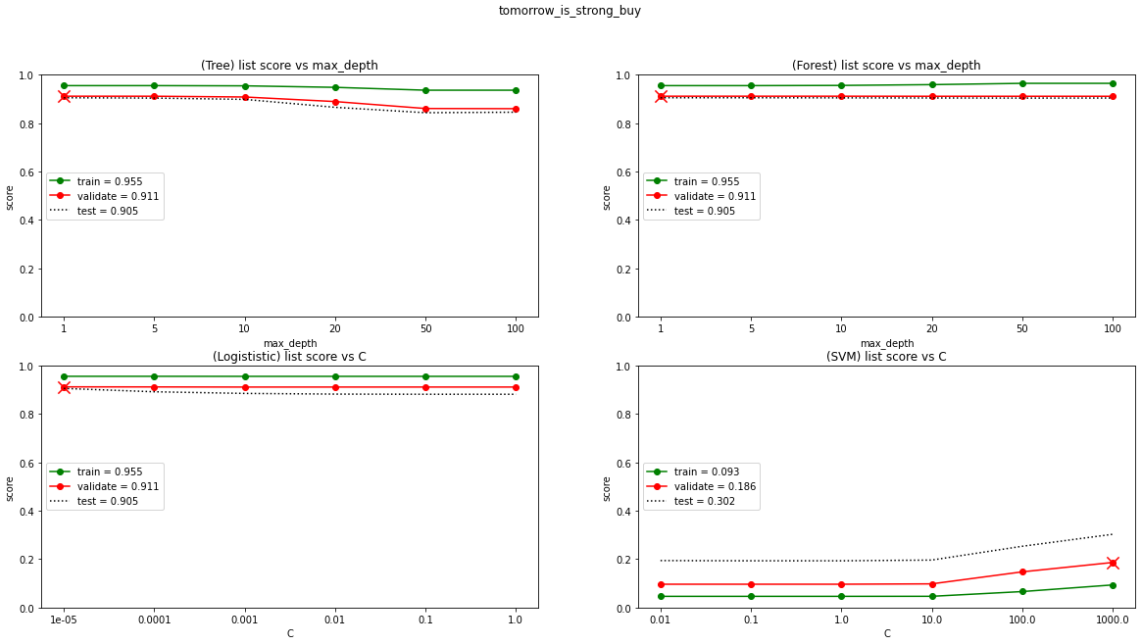

Figure 11: tomorrow_is_strong_buy processed with dataset 2, out2.csv