LaTeX Template for STAT451 Project Report (replace with your project title)

Marlin Lee
Mrlee6@wisc.edu

Jialuo Li
jli2249@wisc.edu

Zhijiang Chen
zchen826@wisc.edu@wisc.edu

Abstract

Stock prices are one of the most unpredictable signals researched. Despite a large amount of stock data being available from daily to hourly to minute by minute, It remains hard to predict any underlying trend. We fit four common models using grid search on Stocks. We then ran this analysis on more than 2000 stocks to get an understanding of the models and the stocks. Using this meta analysis we found that the models are statistically significantly better than a baseline random prediction model. However these models do not have very high accuracy meaning they are not great for actual use with mean accuracy around 37 %. Additionally the meta analysis on these models revealed interesting patterns between stock market cap and country and models predictive accuracy. It showed that some countries like Israel had a lower variance in accuracy and that China had a higher mean of around 39 % for some of its models. While inconclusive the shape and concentration of the market cap feature hints at a couple interesting relationships between it and accuracy.

1. Introduction

Nowadays, in the United States, people rely on the stock market. People's retirement funds and savings are heavily invested in the stock market. The good performance of the stock market has become the most important determinant that people rate their government. Thus, stock analysis is critically needed to get control of the system.

Most trading is completed by some algorithm, which is called algorithmic trading. Usually, people use technical indicators and time series or other algorithms to set up their trading strategies. With the development of technology, more and more people are trying to develop machine learning for trading. Here, we use classification techniques to create our project. In trading, classification techniques can be used to provide a class or a category to security under observation. For instance, Classification techniques can predict the type of order action to be taken for a given stock that is expected to go up, down, or stay the same. In our project, we use some stock indicators and classification algorithms to do stock trend prediction. Furthermore to guarantee the effectiveness of these models we look at their performance over many stocks.

Figure 1. Example stock graph 1

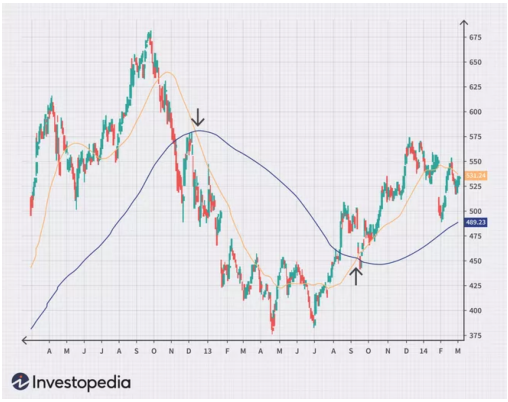

Figure 2. Example stock graph 2

2. Related Work

There are plenty of people who use machine learning to do market trend prediction. Most of them are financial institutions, and therefore don't like to publish their updated strategy because timeliness is very important for trading strategy. However, we found a class called Trading with Machine Learning: Classification and SVM from the Quantra platform[3]. We got some of the generated features

from this source and some baseline ideas about how to use classification algorithms for stock trend prediction. We also found the article [5] that uses deep learning strategies to get good results. We could not get many ideas from this paper due to the difference in available resources between this project and there's. As we do some model limit testing we found this paper [4]. It explores the validity of models but in a very different way due to the different constraints.

3. Proposed Method

This project aims to have a highly accurate algorithm and know how it behaves on different stocks. To reduce the amount of options we are limiting the problem down to a non time series classification of the wider problem space. Generally the accuracy of models analysing stock prices is pretty low. So our goal is to have accuracy above 40%. If that fails we will at least try to have a model that is statistically better than random chance.

3.1. Model fitting

We applied Gradient boost, randomforest, k nearest neighbors(KNN), and a stacking model. Due to the large amount of uncontrolled variance financial models tend to over or under fit depending on the use of a validation set.

3.2. Tunning hyperparameter

For each model we used Gridsearch to search some of the relevant parameters. Because we did this tunning on over 7000 models we picked a smaller number of hyperparameters per model. Due to issues with underfitting we had the Gridsearch optimize for average f1 score instead of the average which is normal. This forces the model to not ignore categories.

3.3. Naive prediction

We never actually train this model but it is important statistically for the analysis and will be referenced many more times in this paper so we will quickly go over it. This model guesses a category at random ignoring the training set completely. If the independent features had no relationship to the actually produced value we expect all models to be equivalent to this one. This means that proving that a model is actually working and giving results is a question of whether the model is predicting similarly to the naive model. Because the categories are balanced this model will have an accuracy of 33%.

3.4. k Nearest Neighbors

KNN is a logical starting point as it is simple to implement and can work for a wide number of models. It predicts by looking at what the closest K Neighbors are classified as. In the grid search we looked at 1NN,5NN,20NN to sample a range of possible boundary limits.

3.5. random forest

Random Forest is a good model for chaotic systems like the stock market. It fits many decision tree models on subsections of the data and takes the mode of the group of trees predicted. The only hyper parameter explored was using 100, 200, 400 trees in the model.

3.6. Gradient boost

This model is similar to the random forest but that the trees are not picked at random. This model iterates making the next set of models compensate for the previous tree's errors. This we tested max depth of 1 2 or 4 and number of trees fitted between 50 and 100. We also set the model learning rate at .9 and the minimum fraction of a leaf weight to .1 to help avoid under-fitting to guessing the category with the most occurrences.

3.7. stacking

This classifier looks at multiple models to try to get a better prediction than either. This version used a stack of Xgboost and KNN, with logistic regression for tiebreakers. We explored the hyperparameters of both models selected above.

3.8. Cross stock analysis

The most important part of having a model is knowing how well it functions and what affects its performance. To get a good understanding of how well these models generalize to predicting stocks in general and what factors affect the fit we used the above models on 1910 stocks and stored the results along with information on the stocks for a second step of analysis.

With this data we looked at the mean and standard deviation of the models. We can then compare these distributions to the naive models true mean and standard error and get a P-value using unpaired difference of means t-test. The formula for the T value is below

$$T = \frac{Mean_1 - Mean_2}{\sqrt{\frac{(N_1-1)Variance_1+(N_2-1)Variance_2}{N_1+N_2-2}}\sqrt{\frac{1}{N_1}+\frac{1}{N_2}}}$$

And because the number of samples will be the same in each group we can reduce the formula down to:

$$T = \frac{Mean_1 - Mean_2}{\sqrt{\frac{(N-1)(Variance_1+Variance_2)}{2N-2}}\sqrt{\frac{2}{N}}}$$

But given the number of stocks we can presuppose the distribution of the naive model because it will be a normalized binomial distribution. This means that the standard deviation will be $\sqrt{\frac{1*2}{3*3*1910}} = \sqrt{\frac{1}{8595}}$. This

means the above formula becomes

$$T = \frac{Mean_1 - .333}{\sqrt{\frac{(1909)(Variance_1 + \sqrt{\frac{1}{8595}})}{3818}}\sqrt{\frac{1}{955}}}$$

This T score can be plugged in with the number of stocks analysis to get the corresponding P-Value. This value is the probability that the data collected would have the mean and standard deviation it does given that the true generation is a naive prediction. This will let us find significance that other methods might not have the power to do.

4. Experiments

This project had an extensive data generation process. The main steps contain Scraping Data from the Yahoo finance API and making features.

4.1. Dataset

We used to resources to make the data set we used in the analysis. We used the Yahoo finance API that has an easy interface with python. The API lets you download stock data of many resolutions. Resources to quarry the API are bellow [1]. The API needs the symbol of the stock to work so we downloaded a dataset of stock info from nasdaq [2] to use the list of stock symbol to get from the Yahoo finance server. We used this dataset again to label the stocks with information about the country, market cap and industry. This Gave us a dataset of 8,000 Stocks with 8,620,792 rows. We ended up using only a fraction of this due to computational limitations.

4.2. Making feature

The Yahoo finance API datasets of stocks only contains the open price which is the price at which a security first trades upon the opening of an exchange on a trading day, close price which is the last price at which a security traded during the regular trading day, highest price during a certain time period, and the lowest price during a time period. Those features are bad for Machine learning models because they are very dependent on the volume of the stock and dependent on the previous days. We need to make more functional feature so as to make our model function.

Usually, stock analyzers adopt such feature values:

RSI: Relative strength Index, the average gain of up time(When the price rises) divided by average loss during the down time(when the price plunges). In our model, we set the period as ten minutes. That measures the strength of a stock.

SMA: Simple moving average average of price shift between two security check. Calculated by shift between two close price divided by the time period between. That measures how fast the price might change

Corr-Correlation between two security checked prices. Positive correlation means price at the two check point have the same trend, both increase and both decrease. Negative correlation is just the opposite, one increase while the other decrease

ADX: Average directional index. Measures the value of the strength of a trend, range from 1 to 100.

SAR-Stop and Reverse, is calculated by the turning point number between a certain period. Is used for measuring is used for measuring the prices trend and where trend will reverse.

Ret: return is calculated by the price shift between two open prices A and B(suppose A is head of B) divided by the price of B. Used for calculating the percentage of earning one might gain from if buying at a certain point.

Open shift price: the shift between two nearest prices. Close shift price, High shift price, Low shift price, are defined the same way as opend shift price.

lowest price of the minute/hour/day. Those are all classical stock analysing parameters.

All features were normalized so they have a mean of zero and a standard deviation of 1. This is done so models that use distance like KNN do not favor models that span a wider set of numbers then models that don't. Categorical data That we use in the meta analysis is one-hot encoded to allow it to be used.

4.3. Making label

We mainly categorize stock into three classes, class 1: The return is relatively high which is good buying point, class 0: Plain or medium return which is plain buying point, class -1 Low or negative return, which is bad buying point. We used the 66.th and 33.th percentile of the train set Ret as a cut-off to decide the label 1,0, or -1. If the Ret at a point is greater than the 66 percentile value, then we define it as class 1. If it is between 33.th and 66.th percentile, we define it as class 0. Class -1 refer to those point that Ret is less than 33.th percent.

33.th percent and 66.th percent cut off is chosen to have balanced category's and reduce risk of our models fitting to predict the largest category instead of actual patterns in the data. This has some issues as some stocks have very concentrated ret distributions meaning that the size of each ret bin varies widely as seen in 3.

4.4. Test-train set split

Test-train set split in financial data is different from normal data because it has auto-correlation. This means that certain indicators are correlated to rows that took place at a similar time. This means that if select the test set at random we could leak some information about the test set to the training set using the near by times. Therefore selecting points at random to make our test set is not appropriate.

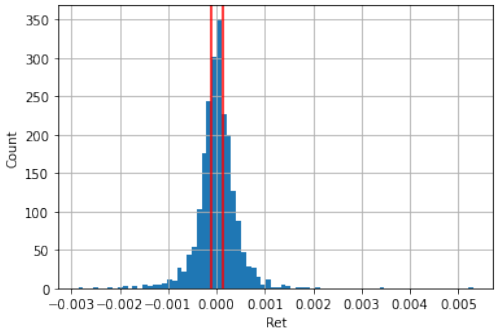

Figure 3. histogram of Return values. with 33,66 percentile in Red

Thus we cut the 20 percent of the whole data at the tail to form our test set, so as to keep both the train set and test continuous in a certain time period.

4.5. Software and Hardware

The analysis was done with just our computers and python. Further analysis could be done with more computational resources because we were unable to analysis all 8000 stocks due to the time it takes to run the models.

5. Results and Discussion

This section will have three sections. First we will look at the results on one stock. Then we will look at the results on all 1,910 stocks trained. Then a discussion section talking about the limitations of the project.

5.1. MSFT Stock analysis

Microsoft is a good model for the general pattern that the stocks follow. First note the shape of the return histogram in Figure 3. Due to the strong clumping the predictive ability might be hampered. This will be displayed in difficulty finding the true relationship between the other data and the ret variable.

The results of the four models are displayed in table 5.1. The models accuracy show a large difference between the models with the Random Forest model having an accuracy of 38% which is enough for a very successful algorithm and the Stacking accuracy lower then a naive prediction model. The next section will show these results in a wider context. The other metric that we are reporting is mean recall and precision. because the models are prone to over and under fit the training data these metrics are a very important indicator of what issues there are in the training process. Due to the number of stocks we are not going to be able to display a confusion matrix for each of them but for this example stock you can see the corresponding matrix's below in

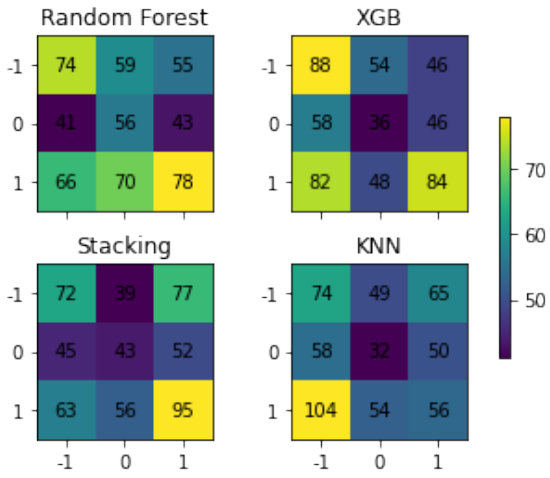

Figure 4. histogram of model accuracy.

Method	Accuracy	Mean recall	Mean precision
XGB	38.37 %	32.74 %	30.25%
Stacking	30.44 %	29.66%	26.83%
RandomForest	39.29 %	32.41%	36.06%
KNN	38.37%	39.78%	33.80%

Table 1. Model results on the Microsoft stock

graphic 4. Most information in these four plots is captured in 5.1 which will be key to our analysis in the next section.

5.2. Meta Stock analysis

There are two main goals we want to get out of the analysis of 1910 stocks. the first to to get a very accurate measure of how good each model is. The second is to know what stocks are better suited to be modeled.

5.3. model performance

We get reasonably normal shaped histograms of the modeled accuracy in Graphic 5. Unpacking this we get an analytic description of this distribution in table 5.3. None of these distributions are centered at 40 % accuracy like we were aiming for. Furthermore the standard deviation is very wide meaning that none of the models behaved consistently on the data set.

We can create a P-value using the method described in section 3. You can find the reported P-values in table 5.3. The results are surprisingly good. All four methods are statistical significant which means that we have good reason to believe that they are better than chance. This modeling is not continued for the within category modeling due to the lower number of stocks in group and the unclear nature of

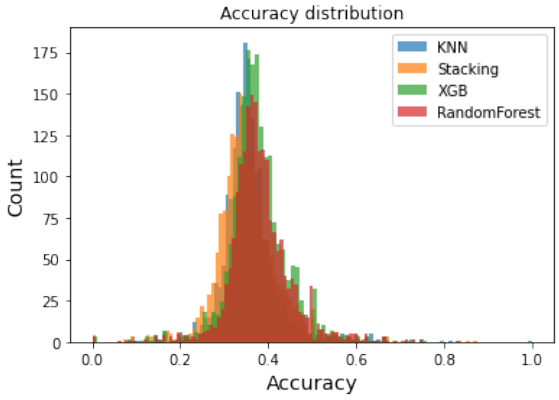

Figure 5. histogram of model accuracy.

Method	Mean	Standard deviation	P-Value
XGB	37.79%	7.05%	$< .0001$
Stacking	35.46%	7.60%	$< .0001$
RandomForest	38.19%	7.22%	$< .0001$
KNN	35.61%	6.83%	$< .0001$

Table 2. Mean and standard deviation of model's accuracy

the validity of applying this method to sub samples.

5.4. model performance within stock characteristics

The largest surprise in for this categorization is that sector was very bad at categorising models. In 6 we see that the standard deviation of the plot is massive meaning that the binned data did a bad job at collecting models of similar ability's. The finance Stacking category was particularly bad where around 66% of the data ranged from 26% to 46%. This massive bound represents the fact that there was bad separation of the data into groups. The worse sign is when the other group, which is made of stocks that have Industry labels with less then 150 stocks analysed, have a lower standard deviation.

Country's comparatively has some more interesting relationships. In 7 we see that the standard deviations tend to be smaller. Despite that the Other category,which is made from stocks that have Country labels with less then 30 Stocks analysed, still have similar levels of deviation there is some signal that this categorization hints at meaningful relationships. In particular Israel stocks have half the standard deviation for XGB and ensemble methods that the others do. Stocks within this category are the closest to having the 1 standard deviation bounding box outside of the range of the naive prediction model mean. Finally the China XGB category has really high mean which might imply that a subsection of the category might be an effective place to look for stocks.

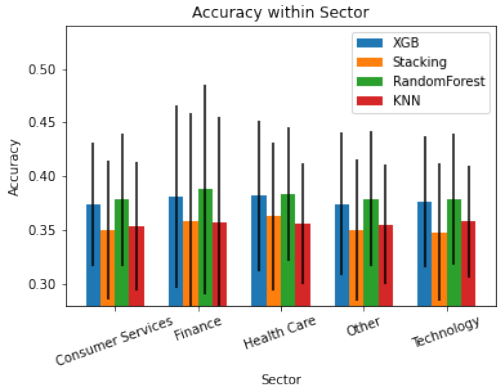

Figure 6. Model accuracy broken down by Sector. Error bars are $\pm$ standard deviation

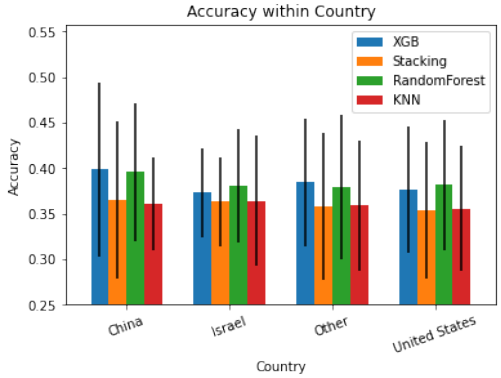

Figure 7. Model accuracy broken down by Country. Error bars are $\pm$ standard deviation

Finally we have the Stocks market cap. We can see the relationship visualized in 8 which shows that the four methods have relatively similar distributions in terms of market cap. The shape of the distribution seems interesting but is actually relatively common with near log normally distributed features. An interesting component is where the center of mass of each model centers. The highest concentration for each is all above the 33% naive model line which help support that there is a meaningful difference between them. These differences in shape might hint in some ground for further analysis but with current tools the way to use this difference is unclear.

We also wished to have an overall stock relationship between all the stock features and there relationship. We fit a random tree model and extracted the feature importance. We can see the results here FeatureImportance showing that market cap was viewed as the most important. This is an imperfect method because the One hot encoding of categorical features leads to them being values less by the importance.

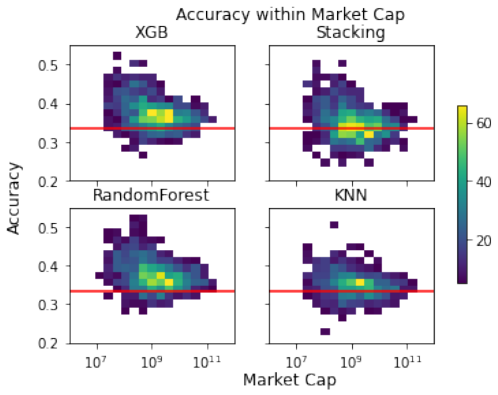

Figure 8. Heat map of relationship between Market cap and Accuracy

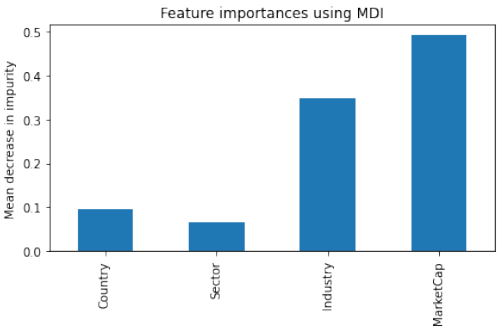

Figure 9. Feature importance for stacking accuracy

This graphic is just to give some base understanding.

5.5. Discussion

The are many potential issues with this work and many areas that it might of been better to focus on. As we had originally collected data on 8000 stocks in three resolutions only analysing minute by minute data of 2000 means that any conclusion needs to be drawn with less confidence then could be done. Also the 2000 stocks where not completely drawn at random with some alphabetical clumping occurred in the fitting process.

One issue briefly mentioned in the result section is that the ret feature is badly created. Because of the concentration of data in the mean the dividing lines do not sufficiently separate the data into strongly positive, negative return. This means that there might be strong feature relationships that we fail to capture given the discretization we used.

Additionally this method of meta-analysis is less commonly done so we are less confident on the procedure and results then for normal modeling reports. Well some research of similar effect was done [4]. It is difficult to tell if the method done to find significance is valid. It is unclear if squishing the model down to its accuracy maintains the meaning of a model sufficiently that it makes sense to compare. Additionally it is unclear if the mean of categorization schemes conveys inherently meaningful information.

Because Stock market is much more variant than other kind of dataset. It's too hard for machine learning to provide reliable prediction of high accuracy. Most models have the accuracy at around 30 percent to 40 percent which is close to the accuracy of naive predictions. A model with only slightly better then 33% accuracy can accuracy can be an effective model as over a long enough time period they will slowly improve there position. If the expected return is one percent at a unit of time period(might be days or months), that means if they invest a million dollars on stock their expected return is ten thousand dollars, and that profit will accumulate with time. their final return will be 1.01^n(n is the calculation of time period). The profit accumulate pretty fast with time passes. So these slightly better then naive models are actually reasonably effective.

6. Conclusions

The main goal was to get a good model for stock prices. Despite significant analysis we must conclude this project failed to create a good enough model. This is not too surprising given the known difficulty of the task. However we have sufficiently analyzed the limits of these weaker models and found that they are better then nothing. This is a good result because it means that the features looked at have some predictive relationship to the predicted data. Furthermore we have made meaningful inferences about the stock market using the performance of these models

This paper can serve as a good basis for many different future analyses because of the massive amount of data collected and the pipeline created. Due to the data collected but not used analysis could be done on all 8000 stocks with multiple resolutions. This could offer insight in the relation between resolution and accuracy and might improve the model. Additionally is would be easy to add new models or hyperparameters to the current pipeline.

Additionally employing other models that use the time series nature of the data like ARIRMA or VARMA could Show a diff rent view of the data achieving higher accuracy or gaining an understanding of how time plays a role in the values.

7. Acknowledgements

We must thank the Yahoo finance API which was essential to getting the data for this project and Quantra which were very useful for the feature creation and baseline of the project.

8. Contributions

This work of this paper was split into three parts. Marlin Lee was the principle assembler of the data set. This means he was the one who downloaded the stocks and saved them in a nice to use format. He was also in charge of running the pipeline on all the stocks and analysing the results.

Frank Chen was in charge in creating the feature needed for the analysis. He was also in charge of planning and high level planning because he knows the most domain knowledge on the topic.

Jialuo Li worked on creating the pipeline on individual stocks. This consisted of fitting a ton of models and selecting the ones that seemed to work the best. It also meant finding what hyper parameters made the most sense to tune.

References

[1] Download market data from yahoo! finance's api.

[2] Stock screener. *https://www.nasdaq.com/market-activity/stocks/screener*.

[3] Trading with machine learning: Classification and svm.

[4] T. G. Dietterich. Mapproximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1896–1923, 1998.

[5] M. O. S. Jingyi Shen. Short-term stock market price trend prediction using a comprehensive deep learning system. *Journal of Big Data*, 7(66), 2020.

9. Image sources

The two images of stock fluctuations are from:
`https://www.investopedia.com/articles/active-trading/041814/four-most-commonlyused-indicators-trend-trading.asp#moving-averages`