Predicting Future Stock Performance with Comprehensive Historical Data

Jianing Guo, Jingxuan Liu, Zhiting Zheng[1]

Abstract

In this project, we want to find the best machine learning model that uses comprehensive historical data related to a stock to predict its future price moving direction.

Specifically, we are interested in:

- Whether the price moving direction of stocks can be predicted by historical data using machine learning models? (For example: APPL for Apple stock)
- Which machine learning model is the better, and what are the most important features if we want to predict the return of a stock?
- How to measure the goodness of the model (For example: what metrics we use in validation set to finetune the hyperparameters)
- What strategy can we make based on the model predictions to outperform the benchmark?

In this report, we took Apple stock (AAPL) as an example to figure out the questions we ask above. Firstly, we gathered data from varied facets including fundamental data, price and volume data, and alternative factors. Then we cleaned the data and did feature engineering. We tried three different machine learning models to fit the data, used the predicted result to build trading strategies, and conducted the back-testing from 2019 to 2022.

1. Data Gathering

In order to include as much and comprehensive information as possible, we gathered varied facets of data as potential features of our models. Specifically, we gathered three kinds of data:

1) Price and volume data from Yahoo finance: This can be downloaded directly from https://finance.yahoo.com/quote/AAPL?p=AAPL&.tsrc=fin-srch
2) Financial data: We manually collected fundamental data from Apple's quarterly financial statements, including profit and loss, assets and liabilities, market capitalization, cash flow, asset turnover, etc.

[1] Jianing Guo (jg2353), Jingxuan Liu (jl3627), and Zhiting Zheng (zz485) are all students registering for ORIE 5741 Learning with Big Messy Data in Spring 2023. We appreciate Professor Haiyun He for her amazing lectures all over the semester and for her helpful comments and suggestions during our project. We thank our TAs for their super kind help throughout the semester. All of us gained a lot from this course and really enjoyed this wonderful journey. The authors are named in alphabetical order, and contribute equally to the project.

3) News sentiment: We got this alternative data from RavenPack News Analytics (RPNA) dataset, a unique source of explanatory and predictive inputs derived from news. For each company, RPNA collects each piece of news related to this company and evaluates a news sentiment score called “Composite Sentiment Score” (CSS). This is a sentiment score between 0 and 100 that represents the news sentiment of a given story by combining various sentiment analysis techniques. CSS scores larger than 50 indicating a positive signal, and less than 50 indicating a negative signal. For each day, we calculated the total number of news related to Apple, and the ratio of positive news to negative news.

We collected and gathered the data described above into three dataframe, from 2012 to 2022.

2. Data Cleaning and Feature Engineering

2.1 Missing Data

1) We only had data on the trading date for the price and volume data, while the news sentiment data existed almost every day. So after merging those two dataframes by date, we firstly dropped the observations without price and volume data.
2) In terms of the financial data, the most frequent data we could get is quarterly, and thus we only had four groups of data per year. After merging these dataframes, we applied the “forward-filling” method, to fill missing values in the fundamental data with the previous non-missing value in each column. In this way, we guaranteed that for each observation, the financial data we used was retrieved from the most recent financial statements.

2.2 Feature and Label Encoding:

1) Label: Since we would like to predict whether the stock price will increase or decrease in the next trading period (next month in our project), instead of focusing on the exact price of the stock, we chose to use classification models. Therefore, we encoded our label y as 0 if the return is negative in the next month, and 1 if the return is positive in the next month.
2) CSS: As we described above, CSS is a news sentiment score that indicates positive signal if it is larger than 50 and negative signal otherwise. So we encoded CSS as a dummy variable: -1 if it is less than 50 and 1 otherwise.

2.3 Data Manipulation to Generate Features:

We used linear and non-linear combinations of original data to generate features.

1) Financial features: We used some simple linear combinations of financial data to calculate fundamental indicators that could reflect the financial conditions of a company, such as Return on Equity, year-on-year growth rate of operating revenue, financial leverage, etc.
2) Technical indicators: We calculated classic technical indicators, including Moving Average Convergence Divergence, Bollinger Band, etc., with price and volume data.
3) Alpha and Beta: We applied Fama-French three-factor model to calculate alpha and betas. In particular, we run the regression
$$r_{stock} - r_f = \alpha + \beta_{market}(Stock - r_f) + \beta_{size}\,(r_{stock} - r_f) + \beta_{value}\,(r_{stock} - r_f)$$
using previous one year data to get the estimates for each day.

Then we calculated the average of those features to represent the features of a month.

2.4 Normalization

After cleaning the data and generating features, we got 32 features in total, which can be found in the appendix. Given that there were significant differences in scale among these features, in order to make our predicting more accurate, we normalized these features by

$$feature_i = \frac{feature_i - mean(feature_i)}{standard\ deviation\ (feature_i)}$$

3. Model Approach and Analysis

We tried three different classification models to predict the returns of the stock in the next month using historical data. For each of these models, we conducted the following four steps to answer our questions and making trading strategies:

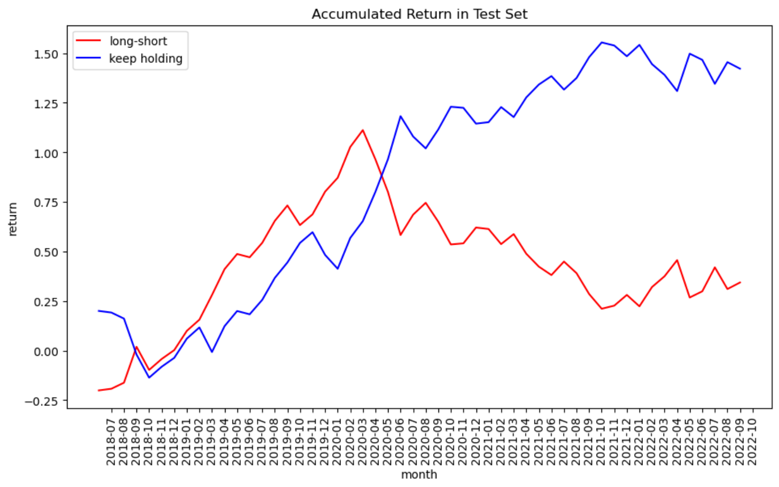
1) We separated the whole dataset into train dataset, validation dataset, and test dataset, as the proportion of 6:1:3. Notice that we did not use the classic proportion of 8:1:1, since in addition to getting the test errors to evaluate our model, we also needed to do backtesting to mimic the real trading in the financial context. So we needed more periods to see the backtesting performance.
2) Then we used the train dataset to train the model, and used the validation dataset to choose the best hyperparameters. Notice that we did not apply cross validation, since in our financial content, we must use the previous data to predict the future performance, in other words, our dataset is a time-series dataset, so we could not simply apply k-fold cross validation, or we would use future data to do prediction.
3) After choosing the best hyperparameters using specific metrics (we will discuss this later in model details), we applied the model in the test data, to get the test errors.

4) Then we made a strategy according to the prediction by each model. Specifically, if the model predicts that the return will be positive, we will long the stock by the end of this month; or we will short the stock. We plot a cumulative return curve of our strategy, comparing it with the benchmark (holding Apple's stock throughout the whole backtesting period), and evaluate the performance of our long-short strategy.

In the following sections, we will describe the details of our models and trading strategies.

3.1 Linear Classifier

To start with, we applied a simple linear SVM to do the classification. We used grid-search to find the best hyperparameter combination, including parameter C, loss function, and penalty form. The best combination is {C = 1.0, loss = hinge, penalty = l2}, using validation accuracy as a metric. The train accuracy and validation accuracy are 0.67 and 0.63, and the test accuracy is 0.56. Then we made our long-short trading strategy as we described above, and the performance can be found in graph 1. We can see that the performance of our strategy failed to beat the benchmark. In particular, the annual return of the benchmark and our strategy are 1.42 and 0.08.

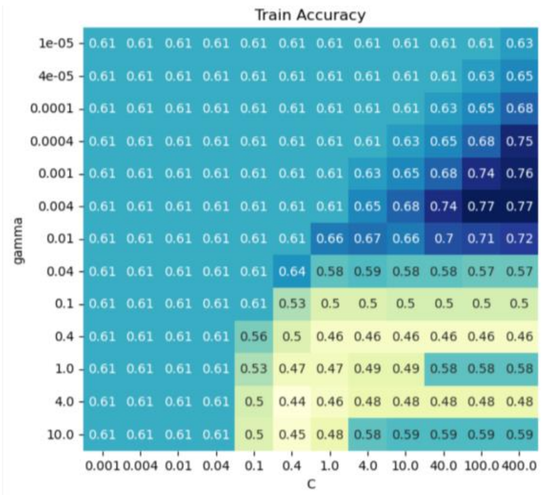


Graph 1

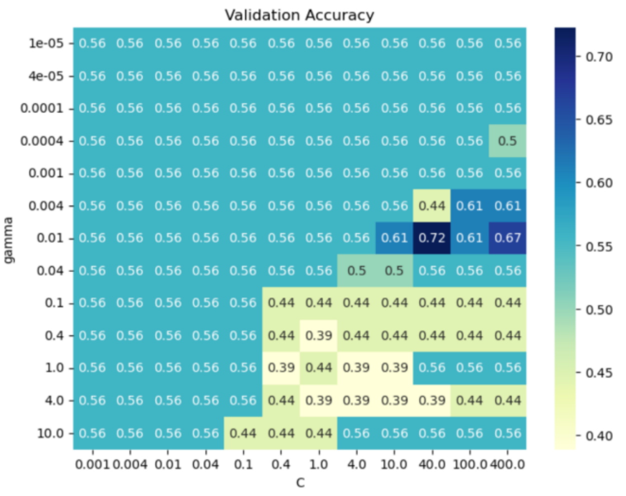
We thought it might be because that in financial context, the relationship between the features and stock cannot be simply predicted by a linear model, and it seemed that the model under-fitted the data, given that the train accuracy, validation accuracy, and test accuracy are not very high. In order to solve this problem, we tried a more complex model – SVM with a non-linear kernel.

3.2 SVM with Non-Linear Kernel

We tried different kernels such as polynomial, Gaussian, and Sigmoid. And we take Sigmoid as an example here to show how we choose hyperparameters.

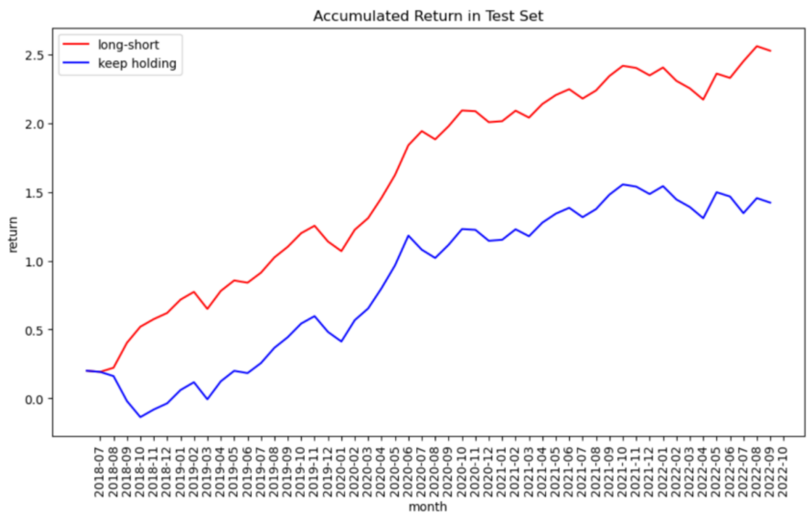


Graph 2



Graph 3

We conducted grid-search to find the best hyperparameter combination of C and gamma, which can be shown in graph 2 and graph 3. As C becomes larger, the train accuracy becomes larger. This makes sense since the C parameter determines the penalty for misclassifying training examples. However, as C goes very large, the model tends to be overfitting since the validation accuracy becomes smaller. Thus, we chose the parameter C of 40, and gamma of 0.01, which led to the highest validation accuracy. With this parameter combination, we had the train accuracy, validation accuracy, and test accuracy of 0.7, 0.72, and 0.67. As we applied the prediction to our long-short strategy, we can see from graph 4 that the performance is very good. It has the annual return of 0.58 and Calma ratio of 2.37, comparing with the benchmark of 0.32.



Graph 4

We did the similar things for other kernel, and the results are shown in table 1.

	Linear	Polynomial	RBF	Sigmoid
Validation Accuracy	0.63	0.67	0.69	0.72

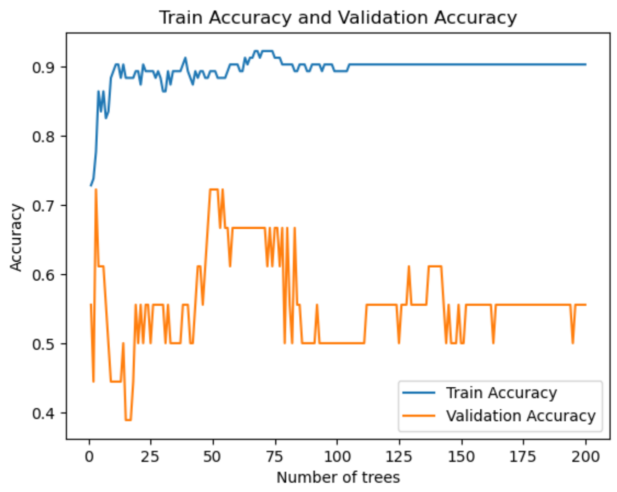
Validation AUC	0.53	0.56	0.58	0.66
Test Accuracy	0.56	0.58	0.59	0.67
Test AUC	0.51	0.52	0.57	0.64
Backtesting Annual Return	0.08	0.22	0.35	0.58
Backtesting Calma Ratio	0.09	0.36	1.21	2.37

Table 1

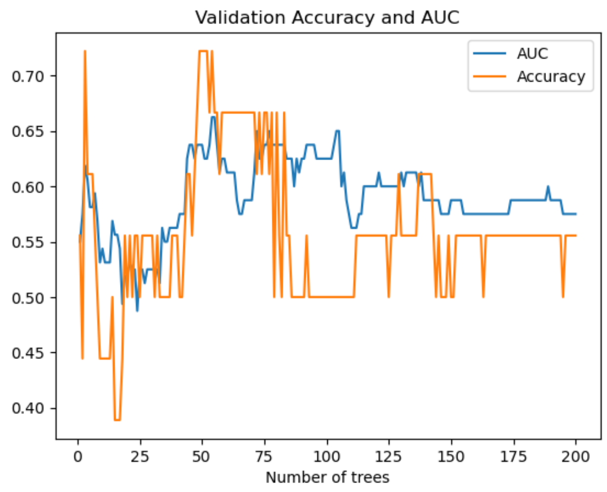
3.3 Random Forest

Since we are not only interested in the performance of the model, but also want to know what factors are important when predicting the return of the stock, we would like to use random forest to show us the important score of each feature while training the data.

One of the most important features of random forest model is the number of trees. Generally, increasing the number of trees in the model can improve its performance, but at the cost of increased computational time and memory requirements. In order to find the best number of trees, we firstly drew a graph of train accuracy and validation accuracy. From graph 5 we can observe that as the number of trees grows, the train accuracy increases, but as it grows to a certain number, the validation accuracy decreases, indicating that the model will overfit with too many trees. Therefore, we thought that the range of the best number of trees needed to be in (40, 60) as the graph shows. Given that there are several choices of parameters that have the highest validation accuracy, we decided to add a new metric - AUC (Area Under the Curve) – to find the best parameter between them. As graph 6 shows, when the number of trees is 54, the validation AUC is the highest, so we selected this hyperparameter to train and test the model. The test accuracy and AUC are 0.61 and 0.61, and backtesting performance of our long-short strategy could be found in graph 7. We can see that our strategy beat the benchmark.

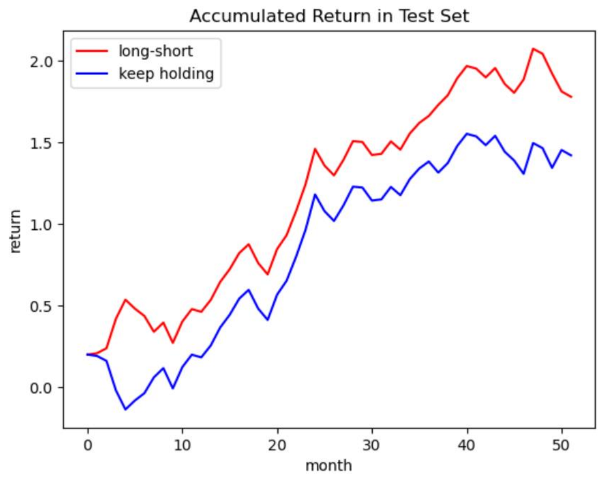


Graph 5

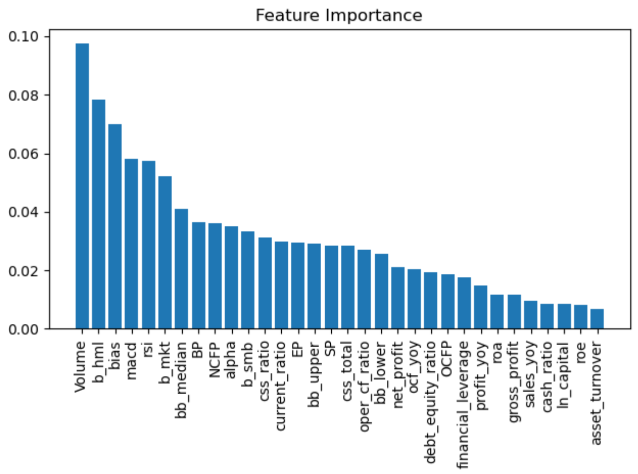


Graph 6

In addition, we also calculated the importance of each feature. As graph 8 shows, the top 5 features are Volume, Beta, BIAS, MACD, and RSI, indicating that both fundamental and technical indicators are important in our model.



Graph 7



Graph 8

3.4 Model Comparison

We compared the performance of three models. From the test accuracy, test AUC, and backtesting performance, we can see that linear SVM is the worst among our three models, and non-linear SVM is the best. Random forest is good but not as good as SVM. We thought it might be because the SVM with sigmoid kernel can capture the non-linear relationship between stock price and the market information, and we could use grid search to find the best hyperparameters. For random forest, it can reduce overfitting and increases the generalizability of the model by aggregating multiple decision trees and introducing randomness. However, there are so many hyperparameters in random forest, such as the maximum depth of each tree, the minimum number of samples required to split a node, the minimum number of samples required to be a leaf node, the maximum number of features considered for splitting at each node, the bootstrap sampling method, etc. It is hard to use a simply grid search to find out the best choice for each hyperparameter, so we thought there was still a lot of space of improvement for our random forest model.

4. Discussion

4.1 Rolling

At this stage, we used the whole previous data to predict the future returns. The lack of rolling training results in the predicted data being exclusively based on a singular training dataset. This may lead to potential negative impact on the accuracy of the model's predictions for the final test data due to the influence of early training data. Thus, in future work, we intend to employ rolling data techniques to train and test our model. For example, to use the data of the previous 72 months to predict the returns for a particular month, which captures the most recent market information.

4.2 What Metrics to Use

Our evaluation methodology primarily utilizes AUC and accuracy metrics, which may not be optimal for assessing the prediction results of price movements. This is due to the unique characteristics of Apple stock, where price increases occur predominantly in large increments, while decreases typically manifest in smaller increments. Consequently, our primary concern is with misclassifying an upward trend as a downward trend, as such errors could result in significant losses during backtesting. Conversely, errors in misclassifying a downward trend as an upward trend may not significantly impact our strategy. Therefore, we seek to develop an evaluation metric that is tailored to the specific features of Apple stock price movements to enhance the model's robustness.

4.3 More Stocks

Currently, our prediction model solely relies on Apple stock to forecast future price movements. To enhance the model's robustness and implement an effective asset allocation strategy, we intend to expand our approach to incorporate multiple stocks with diverse price movements. By analyzing the resulting predictions, we will design an asset allocation model that diversifies risk and generates superior market performance.

4.4 Fairness

The prediction of stock price movements does not raise issues of fairness. Nonetheless, investors intending to utilize our model must acknowledge the inherent risks associated with investments and exercise caution when investing. Additionally, the impact of false positives and false negatives on the strategy's returns may vary depending on the market situation and the time period, factors beyond the model's control.

Appendix – Factors Used as Features

Feature Type	Feature Name	Description
Valuation	EP	*Net Profit / Total market value*
Valuation	BP	*Net profit after deducting non-recurring gains and losses / Total market value*
Valuation	SP	*Operating income / Total market value*
Valuation	NCFP	*Net cash flow / Total market value*
Valuation	OCFP	*Operating cash flow / Total market value*
Growth	sales_yoy	*Year-on-year growth rate of operating revenue*
Growth	profit_yoy	*Year-on-year growth rate of net profit*
Growth	ocf_yoy	*Year-on-year growth rate of operating cash flow*
Financial Quality	roe	*ROE*
Financial Quality	roa	*ROA*
Financial Quality	gross_profit	*Gross profit margin*
Financial Quality	net_profit	*Net profit*
Financial Quality	asset_turnover	*Asset turnover ratio*
Financial Quality	oper_cf_ratio	*Operating cash flow / Net profit*
Leverage	financial_leverage	*Total assets / Net assets*
Leverage	debt_equity_ratio	*Non-current assets / Net assets*
Leverage	cash_ratio	*Cash Ratio*
Leverage	current_ratio	*Current Ratio*
Market Value	ln_capital	*Log of the total market value*
Technical	Volume	*Volume*
Technical	macd	*Classical technical index*
Technical	rsi	*Classical technical index*
Technical	bb_lower	*Lower rail of the Bollinger belt*
Technical	bb_mdian	*Middle rail of the Bollinger belt*
Technical	bb_upper	*Upper rail of the Bollinger belt*
Technical	bias	*Classical technical index*
Beta	b_mkt	*Beta of market factor for Fama-French three factor model*
Beta	b_smb	*Beta of SMB factor for Fama-French three factor model*
Beta	b_hml	*Beta of HML factor for Fama-French three factor model*
Alpha	alpha	*Alpha for Fama-French three factor model*
Alternative	css_ratio	*Ratio of positive news to total news*
Alternative	css_total	*Total number of news*