Predicting Microsoft Stock Price 'Inefficiencies'

Zachary Mor

Brown University
DATA 1030: Hands on Data-Science
https://github.com/zachmor/data1030-proj

I. Introduction

If it was possible to predict if MSFT is undervalued, overvalued, efficient or volatile, it would be very helpful for making financial decisions. These classes can be better defined:

Efficient – not undervalued and not overvalued
Undervalued – before the end of the following week it sustains a price 2% higher for 2 days
Overvalued – before the end of the following week it sustains a price 2% lower for 2 days
Volatile – undervalued and overvalued

These exact parameters are the ones used because it produced a sort of balanced distribution, but any parameters could have been used, for example 5% within 1 month for 3 days. This information would be helpful for options trading strategies specifically because it can be used to predict the value of a contract. If it can be deduced that the price is going to be at a value higher or lower than the option contracts net exercise price (strike price plus premium) the contract can be bought or sold accordingly. The contract will be realized profitably when the model is right. Here, trading data from the NYSE from all of 2021 and the first half of 2022 are used to predict the inefficiency class of the price of the Microsoft stock.

Much research has attempted similar problems, some referenced here are done by Shen and Ali, but they use deep learning methods and they classify just on the direction of change rather than *inefficiencies*. Shen achieves > 90% accuracy with some models and Ali achieves similar results. I hope to expect similar results, though with the more complex classification scheme it may worsen.

The dataset contains 1-minute MSFT stock trading data for 231,171 records between 2021-1-1 and 2022-7-1 which is just 30% of 788,400, the number of minutes in a year. The other 70% of the minutes don't have trading data because the market is only open during work-hours on business days.

Each record contains 6 features: an opening timestamp, opening price, high price, low price, close price, and volume. The open/close values provide a uniformly distributed random record of last-traded price taken at the beginning and end of the one minute period respectively. The high/low values establish range-wise knowledge which speaks to the volatility of the trading period. Volume, the total number of shares traded, signals the intensity of the trading period.

II. Exploratory Data Analysis

Some interesting visualizations from the project are described here, beginning with the inefficiency class distribution. In **Figure 1** a bar plot is used to show the distribution of the target variable, one is produce for the train test and one is produced for the test. It is shown separately because the market experienced a shift in the market cycle, from bullish to bearish, which strongly affects the distribution of the inefficiency classes.

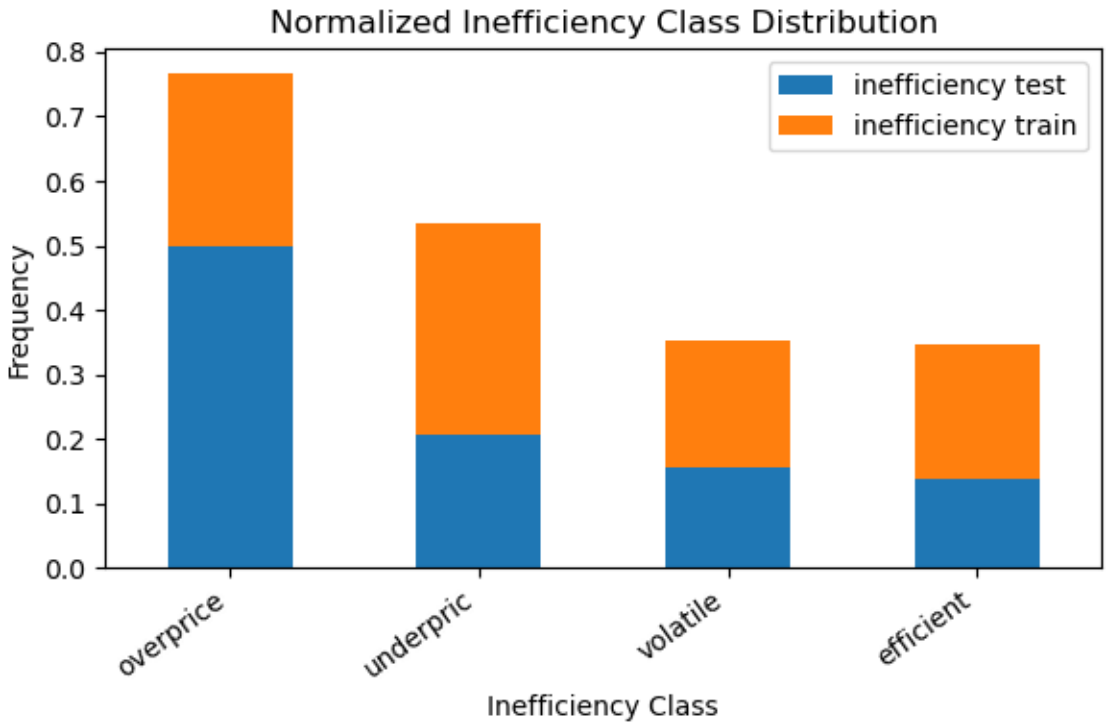

Figure 1

Visualizing multi signal time series can require feature-inclusive plots because signals recorded together are better visualized together to make sense. For example, the high/low prices are helpful because they describe the trading range of the period, and so plotting either one alone

does not fully capture the significance of the feature. Of the five features volume is most appropriately visualized alone because it is not part of a pair. **Figure 2** shows the distribution of the volume feature, grouped by inefficiency class, along with their medians and means, in black and yellow lines respectively.

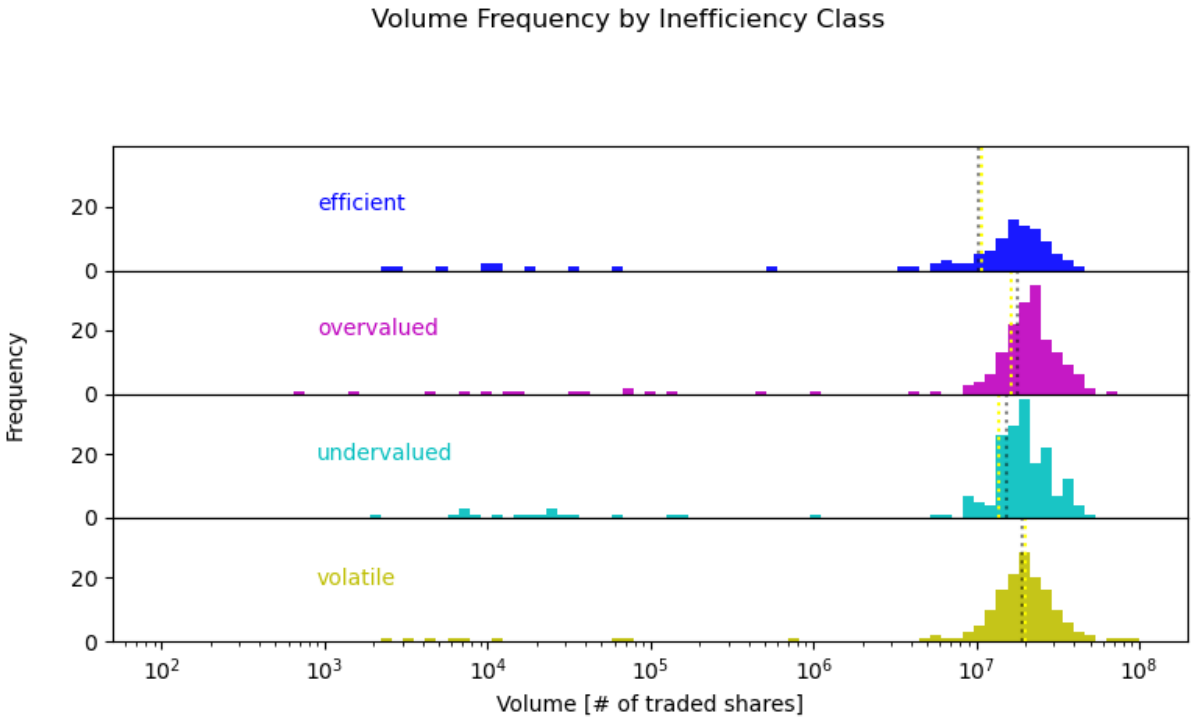

Figure 2

The records' price data can be effectively visualized using a candlestick plot, which is one unified plot that displays all of the price metrics. The thicker candle parts represent the open/close price bounds and the thin wicks represent the high/low price bounds. Traditionally, the sign of change in price for the period is represented by color, where a green candle means it closed *higher* than it opened, and a red candle means it closed *lower* than it opened. While this price-delta sign is not a feature of the original data set, it is captured in preprocess differencing and so is included in the visualization.

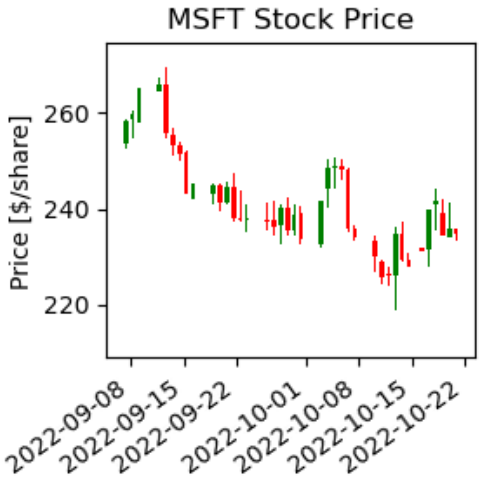

Figure 3

Instead of coloring the candles according to the price-delta sign, they can be colored according to the inefficiency class of the record. Additionally, volume can be visualized scale-less on the x-axis of the candlestick figure with the same color coding. This makes for a unified plot that includes all of the features and the target variable as well.

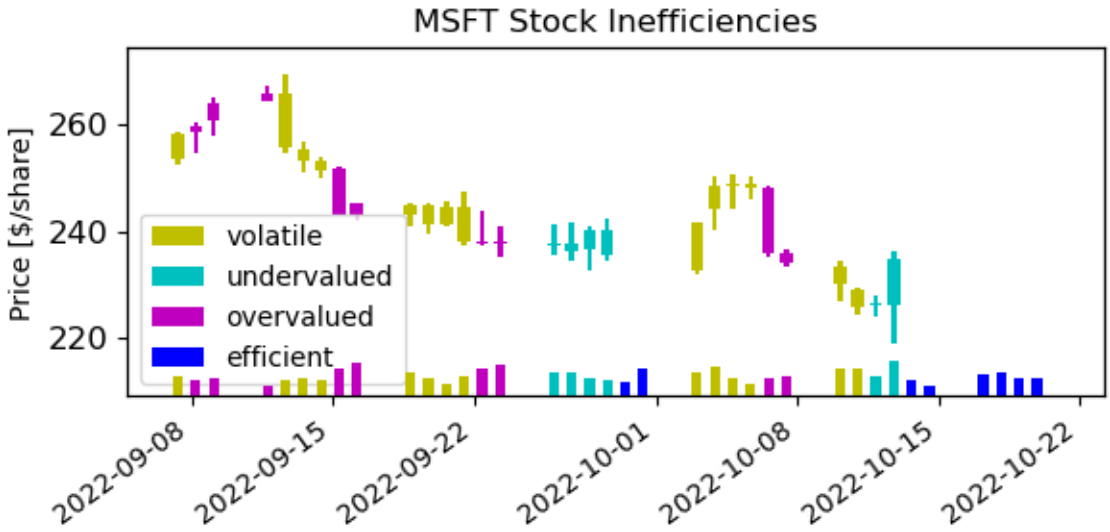

Figure 4

III. Methods

The timeseries is split chronologically with ’21 and Q1 ’22 used for train and validation, and Q2 ‘22 used as test. It is important that it is chronological and not random because the inefficiency class is not independently-identically distributed, as it highly dependent on market trends. With the test set after the train/validation chronologically, the splitting properly represents how the model would be deployed in production. It assures that the model isn’t being trained on data that would be impossible to access in practice.

The data is then down sampled to a frequency that maximizes frequency rate while minimizing the NaN rate. This is done by taking the (frequency, nan rate) pair closest to the ‘perfect signal’, which would be (0, 0) on the frequency-nan rate axis, representing infinitely fine resolution with no missing values. This value turns out to be 1276 minutes, which is approximately 21.27 hours and so the number of records in the relevant range is reduced to 616 records.

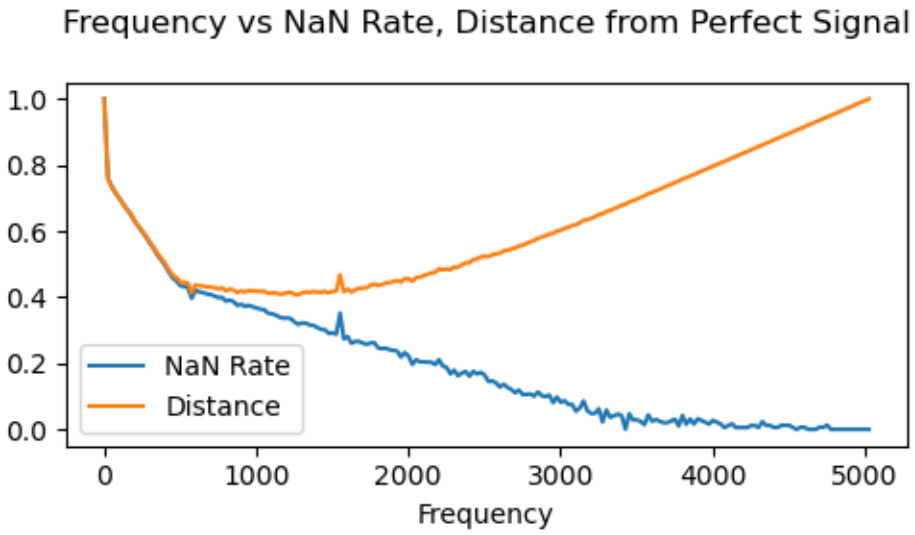

Figure 5

In the beginning of the pipeline, the timeseries gets some engineered features. Two ranges are calculated as differences of the open/close and high/low pairs. The open/close range represents intra-periodic change in price, and the high/low range represents the intra-periodic extrema. These ranges are added to each record, resulting in a net 7 columns.

Then, autoregression and differencing features are added to make the records more stationary. They are both calculated here using a lag times of 2 days. This number is arbitrary and could be optimized for in the hyperparamater grid, but is fixed in this project due to scope and scale. Each one scales the number of features linearly and so, with a down sample frequency of 1276 and a lag of 2 for both autoregression and differencing, the shape of the dataset goes from

Original (231171, 5) ->
Preprocessed (231,171/downsample_frequency, 7 * (auto_lag + 1) * diff_lag)
= (616, 42),

All the features are continuous and so finally, a standard scaler is applied ubiquitously to standardize them.

At this point, all of the described preprocessing is consistently applied to all models, but this is where it diverges depending on the model used. This is because the XGBoost classifier handles NaN natively and so it doesn't require imputation to be imputed. For the other models, LogisticRegression, KNearestNeighbors, and RandomForestClassifier, multivariate imputation is applied.

The table below summarizes the models that were trained. Each row shows: the model name, the parameter-values pairs for which it was optimized and what the optimal parameters are here.

name	parameters	optimal
LogisticRegression (elasticnet)	L1 ratio, C	0.1, 0.001
SVC	C, gamma	100, 0.1
KNearestNeighbors	N_neighbors, p, weights	10, 1, 'distance'
RandomForestClassifier	N_estimators, max depth, max features	300, 9, 0.3

IV. Results

The baseline accuracy of the model is about 40% *overvalued* in the test set. This is interesting in the case of alternatively distributed test sets because the train set's baseline is about 40% *undervalued*. The train set's baseline model is only around 15% accurate on the test set, because the train's baseline model predicts a different class than the test set's baseline.

name	test score
kneighborsclassifier	0.303922
logisticregression	0.137255
randomforestclassifier	0.225490
svc	0.215686
xgboostclassifier	0.196078

Figure 7

The table summarizes each model's performance. The best performing model here is KNearestNeighborsClassifier. It achieves the highest accuracy score compared to the other models. Unfortunately, none of the models achieve a better score than baseline. While this is not very exciting it is perhaps expected, because the referenced papers have larger datasets with more features. Shen uses technical indicators like Stochastic %K, %D, Price rate of change, William %R, and ten more, all calculated as a function of the (open, close, high, low) and Ali uses 60 indicators.

While none of the models seemed to do better than baseline, the most important feature to determine the inefficiency class is volume. This is the case for all of the models across all of the different global importance metrics used, permutation importance, linear model weights, and mean SHAP values. This is interesting because it is not even included as a feature in the other papers. Perhaps the relationship between volume and inefficiency class is easier to learn and so the model exploits this feature which prevents it from learning the relationship with the other features.

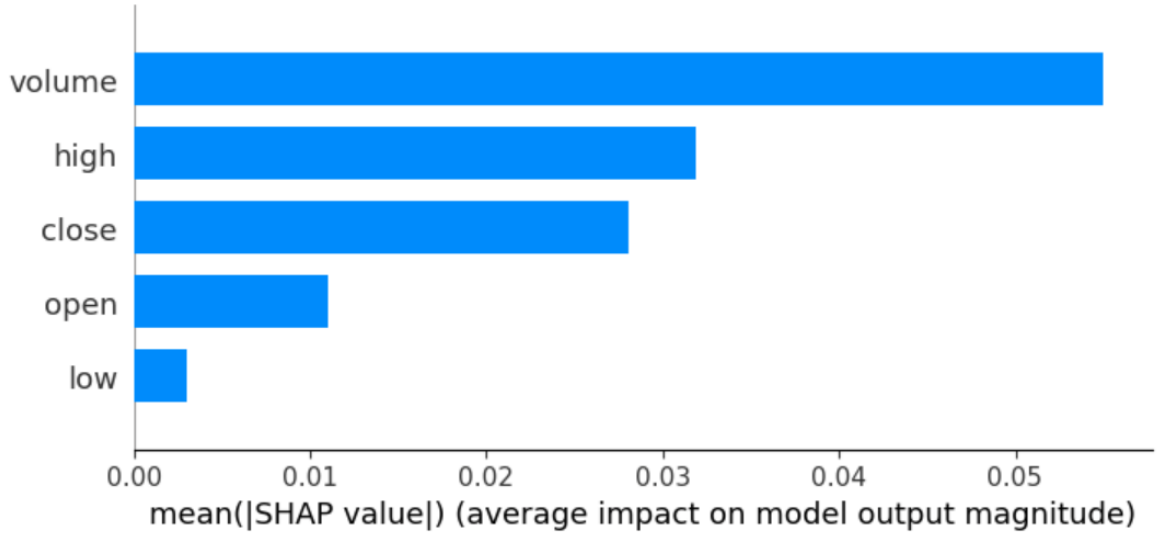

Figure 8

The SHAP package was used to inspect the local feature importance of different specific predictions, the graph below shows how influential the features was for a specific prediction.

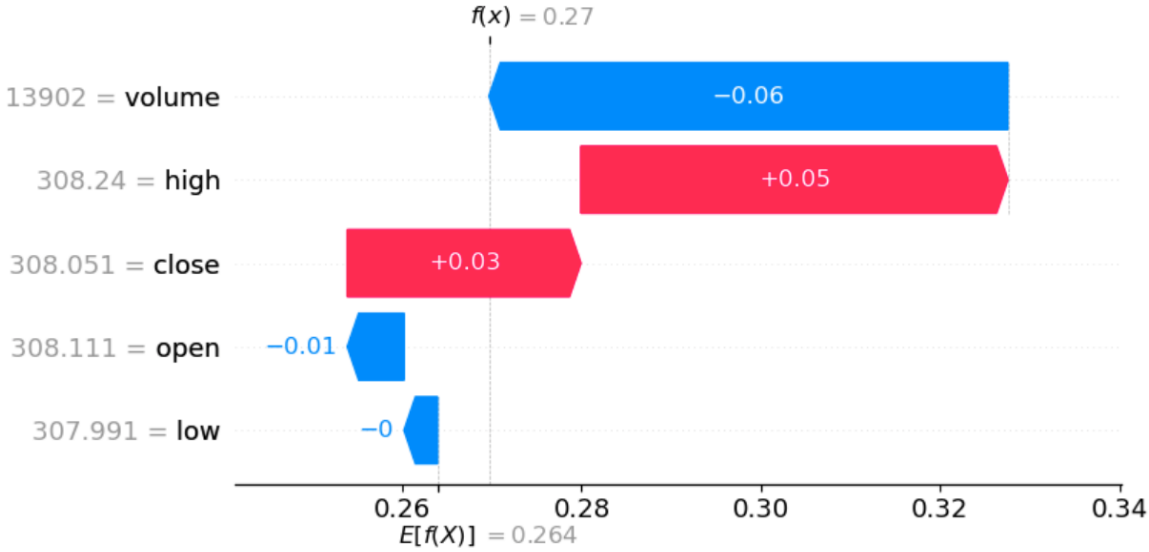

Figure 9

V. Outlook

The performance of the model can certainly be improved with the addition of more features, as shown by Shen and Ali. They add many different financial indicators to make the features more diverse than just two pairs of prices and volume. They also include indicators relating to the general markets movements like popular indexes gains (e.g. S&P 500) This is a good way to give the model a frame of reference for the rest of the market, which apparently could improve scores.

A motivation for a previous version of this project was to use *diverse* data and so it would be interesting to see if non-financial indicators can be used to diversify the features and similarly improve the accuracy of the model, like social media popularity, congress trading, or insider trading to name a few. This sort of information can be scraped as it is public, but it is also readily available as an API thanks to companies like Quiver Quantitative.

Other avenues which can be explored include those relating to sample size and frequency. These are positively correlated because for a constant temporal range, as the frequency increases the period decreases and so the number of samples increase. It would be interesting to compare the accuracy of models on a constant time range across different frequencies, as well as on a constant sample size and different frequencies. Additionally, a model can be trained on data engineered with information from multiple frequency signals at once.

Though more data would slightly improve these models used in this paper, they are not the most fit algorithms for large datasets with many features and highly complex relationships. A better algorithm for larger feature order application would be a deep learning method that is more capable of learning from more features, with more parameters.

VI. References

Muhammad Ali, Dost Muhammad Khan, Muhammad Aamir, Amjad Ali, Zubair Ahmad "Predicting the Direction Movement of Financial Time Series Using Artificial Neural Network and Support Vector Machine",
Complexity, vol. 2021, Article ID 2906463, 13 pages, 2021.
https://doi.org/10.1155/2021/2906463

Zhong, X., Enke, D. Predicting the daily return direction of the stock market using hybrid machine learning algorithms.
Financ Innov **5**, 24 (2019).
https://doi.org/10.1186/s40854-019-0138-0

Data
alphavantage.com