Predicting Stock Growth Rate

by Henrique Raposo

The stock market, a place where dreams are made and crushed; where billion-dollar companies spend hours on day trading trying to make a fortune. Trading equities is a well-known, yet challenging problem, as debate lingers on whether markets are truly efficient, or if they are predictable. According to the market efficient hypothesis, the price of stocks reflects all available existing information, therefore, prices are at fair value, making it impossible for a trader to gain the market. In other words, it should not be possible to use financial data that is already published to predict which equities will have the highest growth and outperform the overall market.

The goal of this project is to challenge the market efficient hypothesis and create a portfolio of stocks picked by a machine learning algorithm that outperforms the S&P500 in a 10-week timeline. At the end of the project, the program will then compete in a simulated investment competition against humans and a fish, in which the goal is to maximize gains from a portfolio of $100,000. The humans, fish, and program will all decide on up to 10 stocks to invest in on October 27th, and the one with the highest profits on December 31st will be the winner. If the market efficient hypothesis is true, the fish is likely to succeed as all stocks are chosen at random, however, if the machine learning algorithm wins, that is a strong indication that it is possible to gain the market. As such, the algorithm output should be the predicted growth rates of stocks in a 10-week period, to then calculate the optimal distribution of the $100,000 that maximizes profit.

A paper published in 2015 by Patel, J., Shah, S., Thakkar, P., and Kotecha, K. attempted a similar feat by predicting stock movement using Artificial Neural Network (ANN), Support Vector Machine (SVM), random forest, and naive-Bayes. The study used stock trading data (open, high, low & close prices) on 10 years of historical data from 2003 to 2012 of two stocks namely Reliance Industries and Infosys Ltd., and two stock price indices CNX Nifty and S&P Bombay Stock Exchange (BSE) Sensex. The image below summarizes their findings:

Stock/Index	Prediction Models			
	ANN Kara et al. (2011)		SVM	
	Accuracy	F-measure	Accuracy	F-measure
S&P BSE SENSEX	0.7839	0.7849	0.7979	0.8168
NIFTY 50	0.8481	0.8635	0.8242	0.8438
Reliance Industries	0.6527	0.6786	0.7275	0.7392
Infosys Ltd.	0.7130	0.7364	0.7988	0.8119
Average	0.7494	0.7659	0.7871	0.8029

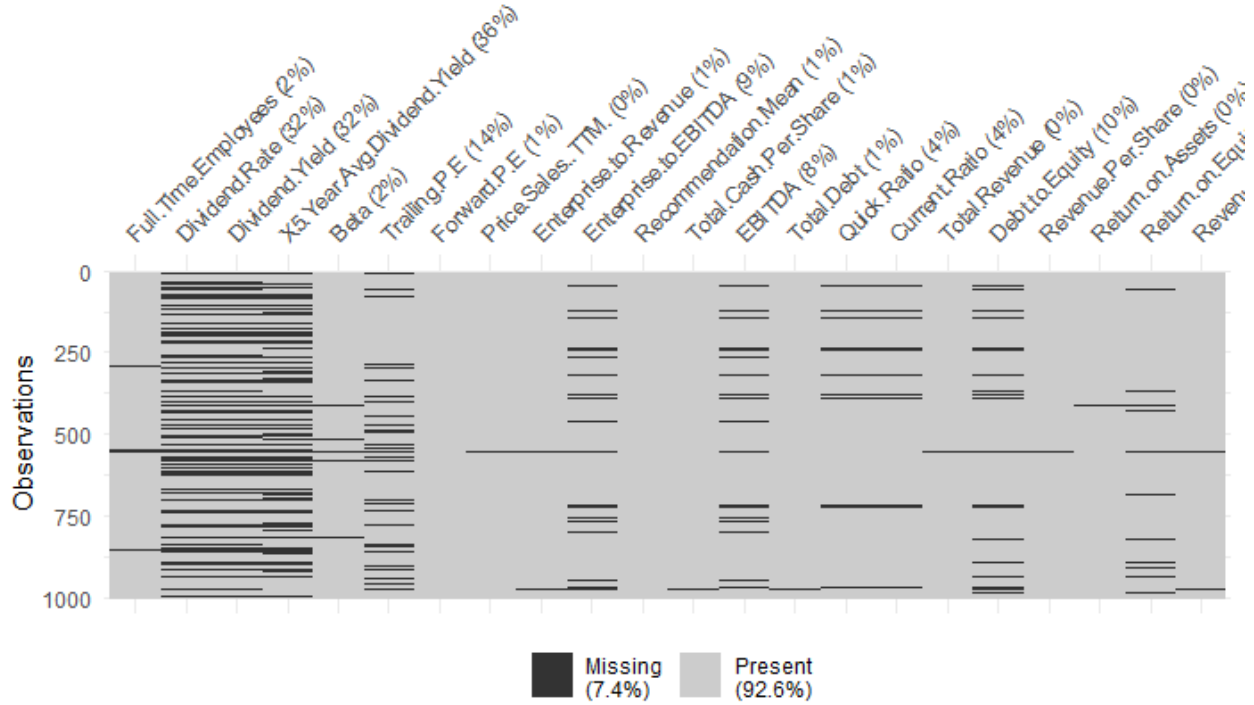
	Random forest		Naive-Bayes (Gaussian)	
	Accuracy	F-measure	Accuracy	F-measure
S&P BSE SENSEX	0.8775	0.8794	0.7354	0.7547
NIFTY 50	0.9131	0.9178	0.8097	0.8193
Reliance Industries	0.7420	0.7447	0.6565	0.6658
Infosys Ltd.	0.8110	0.8176	0.7307	0.7446
Average	0.8359	0.8399	0.7331	0.7461

As we can see, all models used had a relatively high accuracy value, with the lowest being 73.3% accuracy and the highest 83.5%. The most successful models used trend deterministic data to make the predictions such as simple 10-day moving average, weighted 10-day moving average, momentum, stochastic K%, stochastic D%, relative strength index, moving average convergence, Larry William's R%, A/D oscillator and CCI. The paper, thus, shows that predicting stock behavior is not an impossible feat. However, this project will take a different angle, as both time series data is available as well as the financial information of the company.

Regarding the data used in this project, it was obtained through a Python script that leverages Yahoo Finance API to extract the financial information and 2-year historical price of the stocks in the Russell 1000 index. This index lists the top 1000 companies by market capitalization in the United States. From this, two tables are produced, one with the financial data where each row is a ticker symbol and the columns are the metric, and a second table which contains the historical stock price values, in which each row is a day and each column a ticker symbol. The financial information is from September 24th, 2024, and the historical data ranges from September 24th, 2024 to September 26th, 2022.

The financial information table contains 1000 rows and 42 columns, while the historical prices table has 502 rows and 1001 columns. Although there are no duplicates, there are NA values in the data

that will need to be dealt with. The financial data table has a total of 1628 missing values in the columns shown below.



As we can see, dividend rate and dividend yield have most of the missing values. Upon further analysis, the NA are only for companies that do not pay dividend, therefore, we will assume that these missing values are supposed to be 0. Another column with many missing values is Trailing PE, which measures PE from the previous 12 months. To avoid losing data and keep some of the financial info we can drop the trailing PE column and instead calculate the current PE and use that as the metric. That is done by dividing the stock price by the earnings per share. In regards to all other columns, those are true NA values and as it does not make sense to apply mice as each company is significantly different from one another and have different operating strategies, the best course of action is to drop the stocks with missing values. After these changes, 17% of the data was lost, however this guarantees the quality of the models. Once the same stocks are removed from the historical price table, only two stocks have missing data. Considering it is a small portion of the data, these stocks will also be dropped.

The next stage of preparing the data was merging the financial and historical data. For that, the historical prices were transferred into time series features such as the percentage change in price, the ratio of the change in price and the standard deviation of price, all for different time windows. In this process, three more stocks were removed from the data as the percentage of price change coincidently was 0,

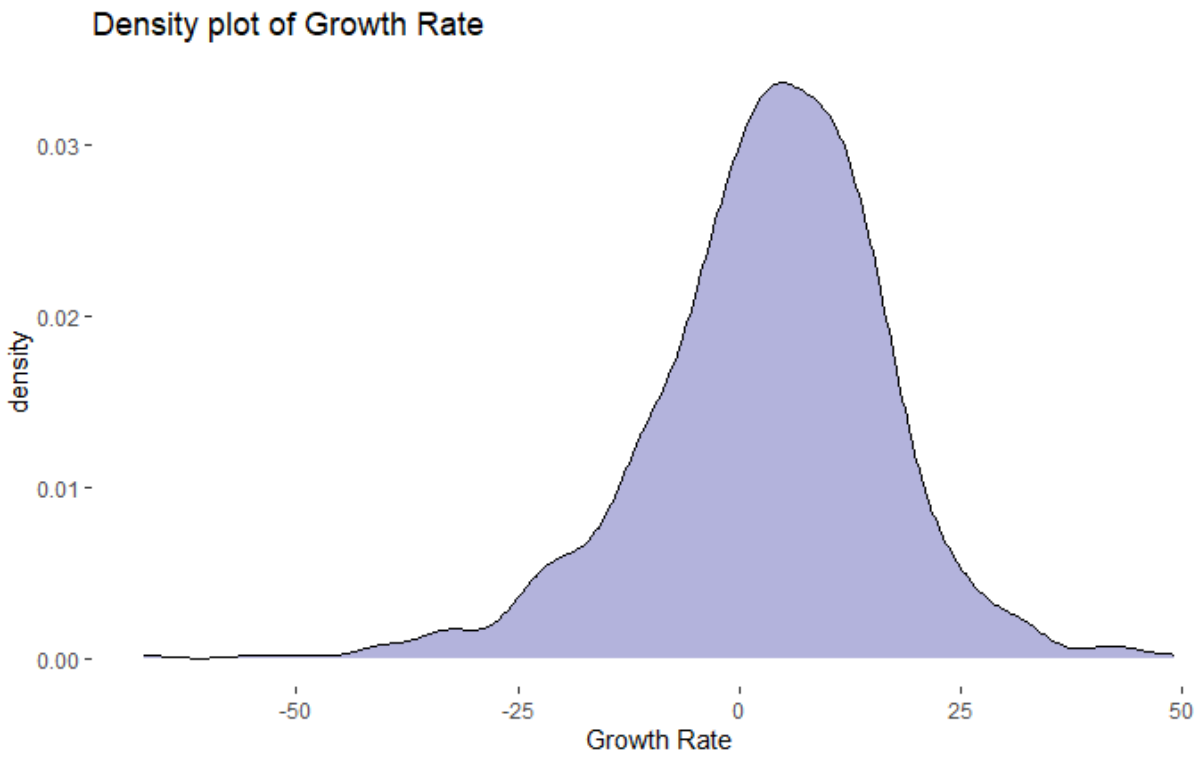
causing the ratio of change to be infinite. As it is better to simply lose three stocks rather than the entire metric, the stocks were dropped.

The last step was to calculate the response variable. The goal is to predict the growth rate of the next 10 weeks, thus we use the date from 10 weeks ago as the base date. We will use that to calculate the growth rate where the price from 10 weeks ago is the previous price and the current price is the most recent price. However, for the time series features, we will only use data from the date of 10 weeks ago and dates before that. This way, the response variable is the actual growth rate the prices had "in the future". Due to limitations from the Yahoo API, on the other hand, we only have access to the financial data from the most recent date. Thus, we assume that the financial metrics will not significantly change in 10 weeks, which is reasonable as that would mean drastic changes happened in the company.

Thus, the growth rate is calculated using the following formula:

$$Growth\ Rate\ = \frac{Current\ Price\ -\ Price\ from\ 10\ Weeks}{Price\ from\ 10\ Weeks} * 100$$

With this, it is possible to visualize a density plot of the response variable as shown below.



From the graph, we see that the distribution appears to be normal with a mean of around 10%. However, the tails are also quite long, in other words, the data has outliers. Normally, this would interfere with the quality of the model, but, for this problem, the outliers are the most important. As the goal is to

maximize the profit, the ideal scenario would be to only pick equities in the tail with a positive growth rate, meaning, accurately predicting the outliers. This makes this project challenging, but also interesting.

Moving onto the modeling process, the data was split into 80% training data and 20% test data. Moreover, three models were used, linear regression, random forest and XGBoost. To compare the results on the test data, the residual mean squared error was used, which calculates the square root of the average squared differences between predicted and actual values. This measures how well a model's predictions match the observed data, with lower values indicating better accuracy. Linear regression had RMSE of 15.79, random forest had 12.70 and XGBoost had 12.38. Given the last model had the lowest RMSE, that was the model adopted and the one I will expand on.

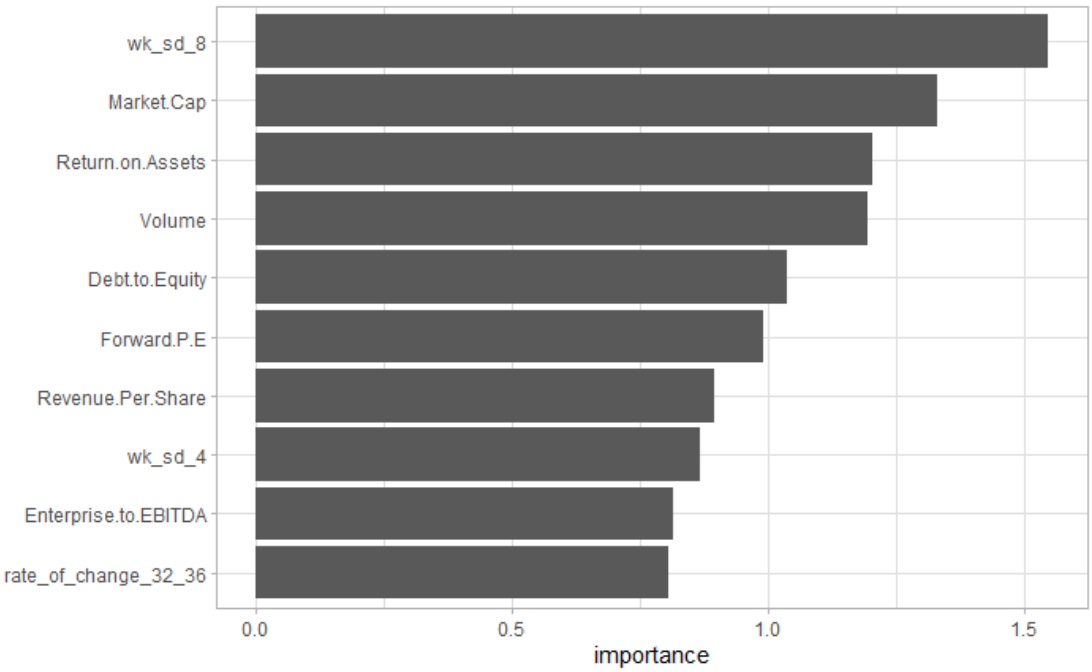
XGBoost is a powerful model that creates multiple decision trees, where each new tree tries to minimize the error made by previous trees. By multiple interactions, the model starts with a decision tree, it calculates the error in prediction, then creates a new model that attempts to predict the residuals or assign higher weights to the samples that were predicted incorrectly in the previous round. This step is repeated multiple times until the model has no further improvement to reach the maximum number of trees to create.

The model is extremely powerful, often producing high accuracy, while being flexible as it can work for both a regression or classification task and doing feature selection by calculating feature importance. However, there are also some limitations as XGBoost is especially hard to tune due to the many hyperparameters, it can easily overfit the data if not tuned correctly and it is not the most interpretable model.

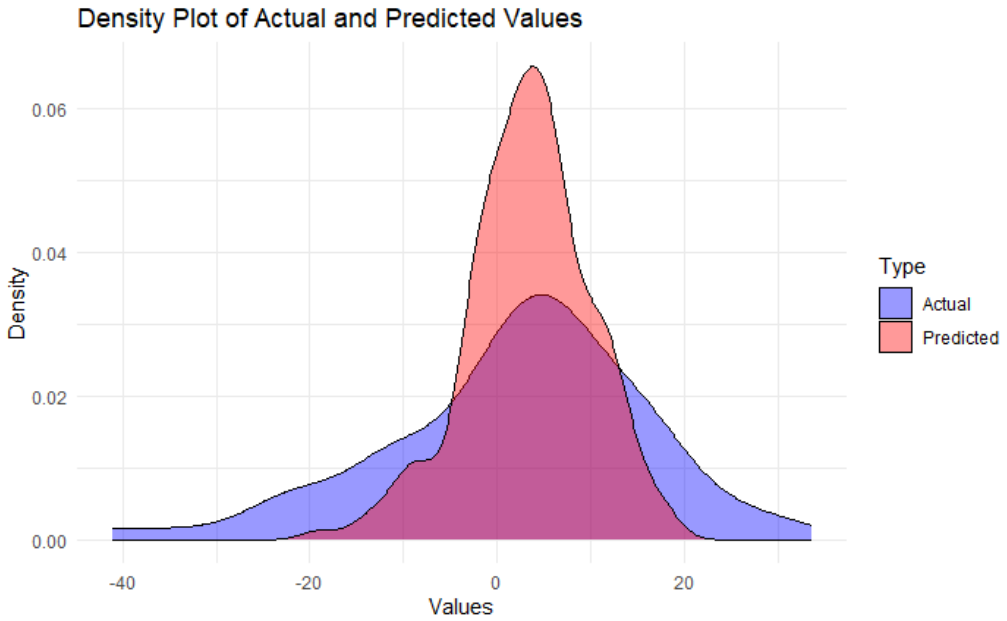
Given the importance of correcting tuning the parameters, we shall discuss what hyper parameters were chosen. By first checking the progression of RMSE and the number of rounds, we notice that the model converges quickly which can be an indicator that the model is prone to overfitting. Hence, the first parameter tuned was the eta value, which dictates how fast the model learns. The eta that minimizes RMSE is 0.1, moreover, the model stops improving around 200 trees, thus the eta is set as 0.1 and nrounds 200. Next, is the depth and minimum child weight of the model. Here it was clear that a max depth of 5 and minimum child weight of 15 minimized the RMSE. Both these values are also good as it will help our model to avoid overfitting the data by producing more shallow trees. Afterwards, gamma was tuned by choosing a gamma of 0.2 as that minimized RMSE. Finally, a subsample of 0.8 and column sample by tree of 1 was decided as to minimize the RMSE.

With all parameters decided the model is ready to run. In the train data we notice that the model does extremely well reaching a RMSE of 4.11 at round 200. Moreover, during the training it was shown that the model would lower the metric even more with more rounds. Although that might seem good, this would lead to the model overfitting the data, hence the maximum round was set at 200. With this, the

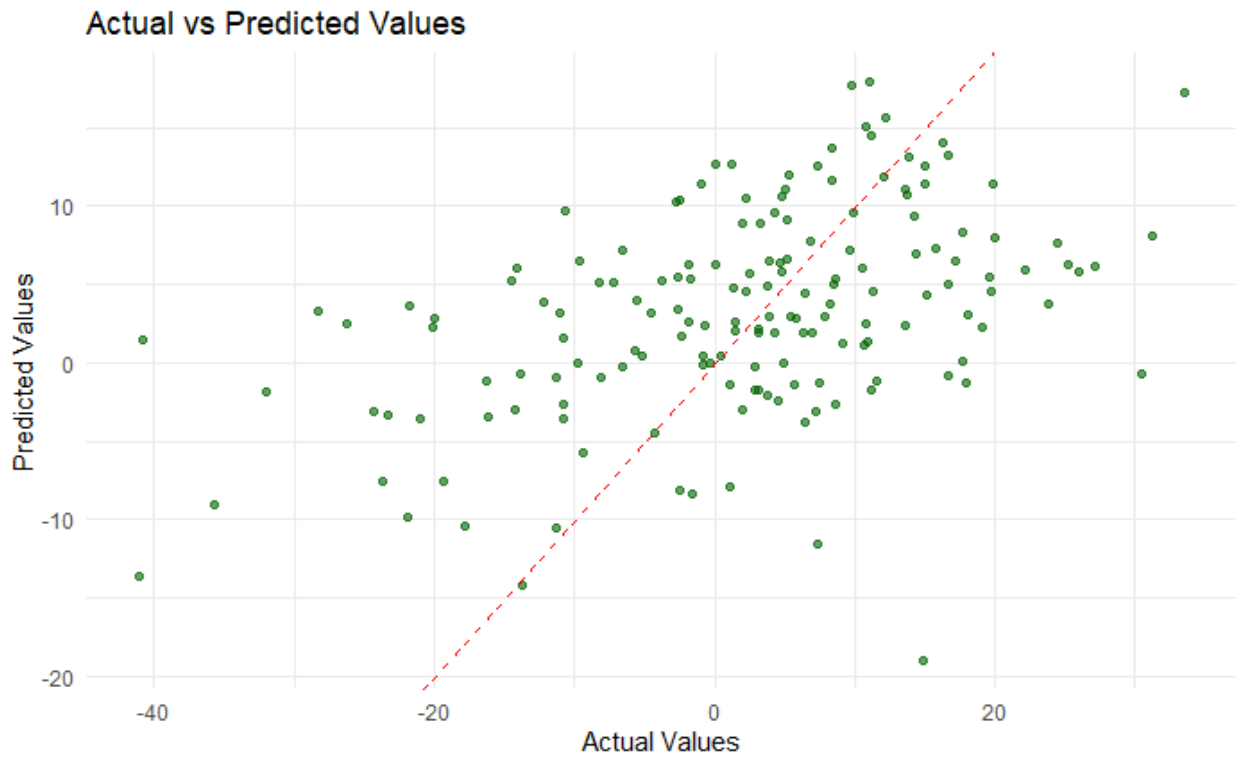
final output had a RMSE of 12.38 which was the lowest from all models. The result was the prediction of the 10-week growth rate of the stocks in the test data. The most relevant metrics in the model was a combination of the time series features and financial information, with the 8-week standard deviation, Market Capitalization, Return on Asset and Volume playing the biggest role. The 10 most important variables are shown in the graph below.



With the model selected and explained, let us look into the results produced. The following plot compares the distribution of the actual versus the predicted growth rates:



Most of the predicted values are concentrated around the mean, causing the distribution to have a high peak, but small tail. When compared to the actual values, we notice that the predictions were more conservative and fluctuated less than the actual growth rate. Moreover, the positive outliers were not captured by the model, which was the main goal of the prediction. This is further supported by the following graph:



The scatter plot dictates that the closer the green dots are from the red line, the higher the accuracy. For small values around 10%, the model does extremely well, however, for the positive outlier where the actual growth rate is higher than 20%, all dotes are very far. In other words, while the actual value was around 20% the predicted value was 10%, showing how the predicted values are more conservative.

Although this would indicate the model failed to accomplish its goal, the narrative changes when we analyze the 10 stocks with the highest predicted 10-week growth rate for the entire dataset, shown in the table below:

Ticker	Predicted Growth Rate	Actual Growth Rate
DUOL	37.76	44.04
K	36.98	42.41
INSP	33.87	42.46
APP	29.74	49.17
DOCS	29.70	40.95
AFRM	29.66	33.35
PLTR	28.27	28.71
EXAS	27.67	38.37
VNO	26.81	32.10
THC	26.17	24.69

7 out of the 10 stocks in the table are in the top 10 stocks with the highest actual growth rate, with only PLTR, VNO and THC outside the top 10. However, all stocks were still in at least the top 30 stocks with the highest actual growth rate and with actual growth rate of more than 24%. This demonstrates that although the rates are different, the model is still able to filter and identify the stocks that grew the most.

Returning to the simulated investment competition, the model, then, would be able to correctly choose the stocks to invest in. Although it may not be possible to calculate how to distribute the money based on the predicted growth due to the difference in actual growth, it can still filter the correct stocks to invest in. Considering the S&P500 10-week growth rate in the same period was 8.6% while all the stocks the model would choose had a growth of at least 24%, there is a strong indicator that the market is not efficient as it is possible to use current information to predict growth rate and beat the market.

It is worth noting, however, that the model was only trained by using the period of June to September in 2024, in other words, it is biased to a single 10-week period. Given more time, we would want to test the model for multiple 10 weeks across multiple years. This would likely enhance the performance of the model and make it even more trustworthy.

In conclusion, a XGBoost model that uses financial and historical data from a stock is able to identify which stocks have the highest growth rates in a 10-week period and surpass the market average. This is initial proof that the market is not efficient, however, it is still necessary to test in reality by conducting the simulated competition and seeing the winner.

Bibliography

Downey, L. (n.d.). *Efficient market hypothesis (EMH): Definition and critique*. Investopedia. https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, *42*(1), 259–268. https://doi.org/10.1016/j.eswa.2014.07.040

Ganti, A. (n.d.). *What is the Russell 1000 index? definition, holdings, and returns*. Investopedia. https://www.investopedia.com/terms/r/russell_1000index.asp