PREDICTIVE STOCK PRICE MODELING AND ANALYSIS

Ethan Lewis

Wake Forest University | 3 December 2021

Table of Contents

ABSTRACT

Predictive stock analysis is an invaluable skill with obvious financial applications. In the report below, we detail the process of cleaning 5 years' worth of United States financial data and fitting and testing multiple models and predictive algorithms including classification forests and logistic regression. A summary of our results can be found below in Table 0.1; a fully detailed report of our findings, conclusions, and recommendations for future work follow Table 0.1.

Table 0.1

Model	Accuracy Rate
Bagged Classification Forest	63.39%
Gradient Boosted Classification Tree	63.74%
Logistic Regression	46.32%

SECTION 1: Introduction

Financial engineering is a massive and ever-growing field; it's importance in an economically competitive and driven society cannot be understated. With this in mind, our research group has sought to construct predictive United States stock performance models via bagged and boosted decision tree algorithms as well as logistic regression.

SECTION 2: Data

As our data frame's name implies, *200+ Financial Indicators of US Stocks (2014-2018)*[1] spans five years; however, each year is allotted an individual data frame. Each of these year-specific subsets contain 226 variables worth of information (including "Class" and "PriceVar" which will serve as our binary and perfectly correlated numeric response variables, respectively) on a varying collection of stocks.

In other words, the five subsets share the same variables; however, they contain a varying number and collection of stocks (observations). Ultimately, we would like to trim each subset to contain identical stocks and stack them to create one, homogeneous data frame to be used for model building and analysis.

SECTION 3: Data Cleaning

Each subset is riddled with missing data; several variables lack the majority of their information and we also have concerns regarding the overwhelming number of 0-valued entries. Fortunately, the author of our data frame provided Python cleaning steps/code[2] which allowed us to expedite the data preparation phase.

First and foremost, we need to remove "Class" and "PriceVar" from each subset until the conclusion of data cleaning. We do not want either variable involved in the manipulation process outlined below.

Each subset will face an identical set of cleaning steps and we can begin by identifying the percentage of 'NA' and 0-valued entries within each variable. Based on these results, we can determine a reasonable percentage threshold any given variable may not cross without being dropped from the subset entirely. An 'NA' threshold of 5-7% is recommended, while a 0-valued threshold should be 5-10%[2]. This step simultaneously reduces missing data as well as the overwhelming number of original variables in each subset.

Based on the initial cleaning step, we know several variables still contain 'NA' entries and we need to find a way to fill them. We would like to consider imputing these missing entries based on the variable's mean value, particularly by "Sector". The process of averaging out variables across the entire subset is too simplistic and homogenous; instead, it is more reasonable to assume stocks in identical sectors behavior similarly, and they should be separated and distinguished as such[2].

First, however, we can get a jump on improving the accuracy of our imputation (and future models) by removing the extreme values from our remaining variables. Excluding the top and bottom 3% of values within each variable vastly improves their individual variances[2].

From here we can execute our imputation strategy, return "Class" and "PriceVar" to the newly cleaned data frames, and begin to devise our own cleaning strategy.

Despite each subset undergoing an identical set of cleaning steps, our newly cleaned subsets do not contain the same collection, nor number, of variables. We can quickly identify the variables shared across all five subsets and trim the individual subsets to only contain these specific 40 variables. In terms of model simplicity and computational expense, 40 variables is a massive improvement upon our original amount of 226.

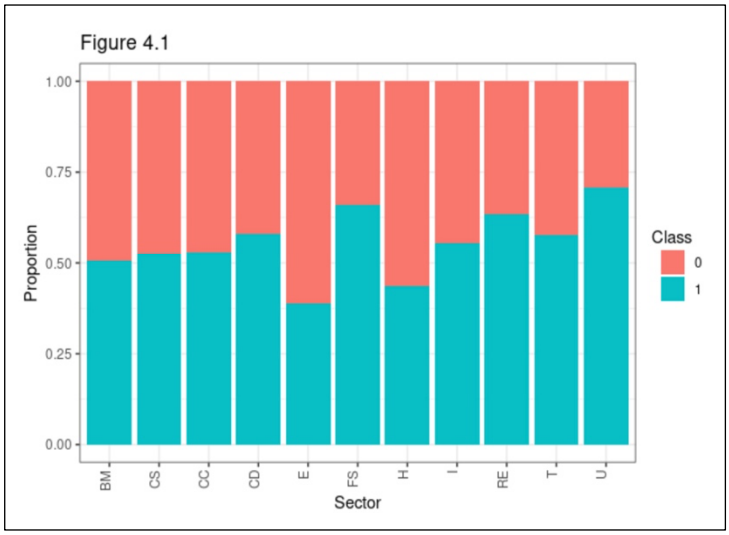
Now, we are ready to combine our annual subsets. To do so, we identified the "Stock" values shared across all five subsets and placed these observations in a list. This list was then used to cross reference the subsets' "Stock" values; if the stock did not appear in both the list and the

annual subset, its entire row was removed. In total, 3,726 stocks were found to be shared amongst all of the data frames.

Finally, we can combine the individual subsets to produce a completely filled, homogenous data frame with 18,630 observations and 40 variables (including "Class" and "PriceVar"). A **Variable Index** is attached to the end of this report.

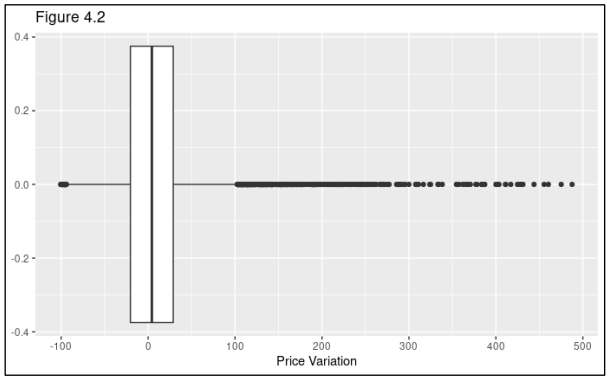
SECTION 4: Exploratory Data Analysis

First, we would like to validate the assumption made during our cleaning process by visualizing the distribution of the proportion of "Class" within each "Sector" using Figure 4.1.



In doing so, we can clearly note a difference among the various levels, thus validating our assumption to impute the missing data by "Sector". We will also keep this variable in mind as we move towards model building.

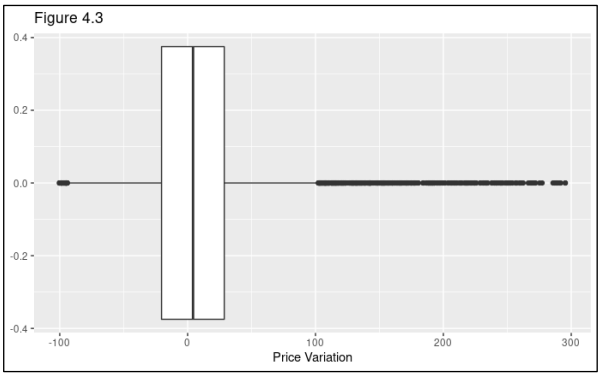
Although we preemptively removed the most extreme (top and bottom) 3% of values within each feature, we would like to investigate the possibility of outliers in the data frame further. To do so, instead of directly observing our binary response variable, "Class", we will identify outliers by plotting the distribution of the perfectly correlated, numeric variable "PriceVar". A negative "PriceVar" value equates to a "Class" of 0, while a positive value equates to a 1.



To begin this process, we will remove all observations exceeding a "PriceVar" value of 500%: a threshold to weed out "inorganic growth"[2]. While this initial sweep removed 44 outliers, Figure 4.2 indicates the existence of several more extreme "PriceVar" values.

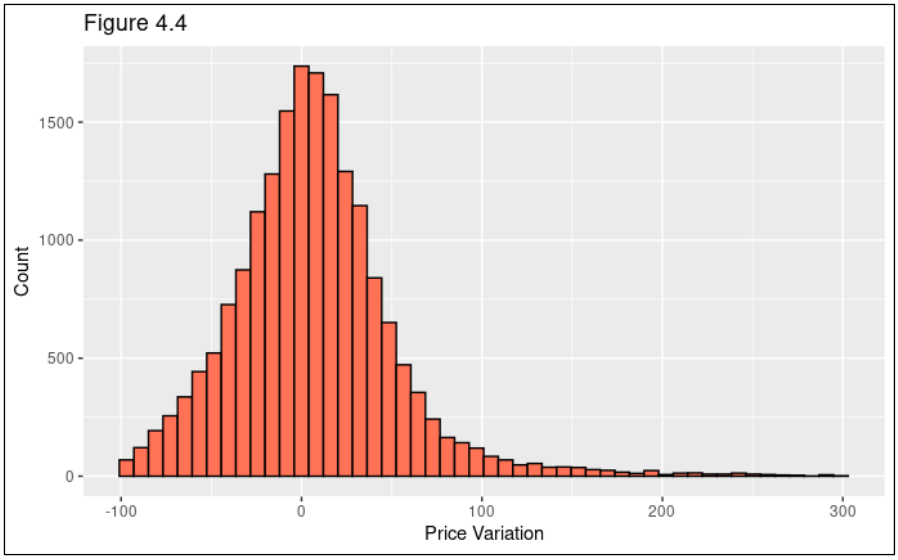
These remaining outliers are fairly bunched together; but, as we approach a "PriceVar" value of 300%, they slowly begin to spread apart. With

this in mind, we will constrain our threshold to remove all observations exceeding a “Price Var” value of 300%. Figure 4.3 displays the effects of applying this data transformation and we can notice an obvious improvement upon Figure 4.2.



While outliers certainly remain in this trimmed data, these constrained “PriceVar” values now seem more intuitive. In other words, it is more reasonable to assume a stock has the potential to triple in price throughout a year as opposed to quintuple. Additionally, we still want to afford our models to the opportunity to contain and consider some extreme performers.

In all, the outlier removal process has altered our data frame to contain 18,545 observations. A finalized, right skewed distribution of “PriceVar” can be found below in Figure 4.4.



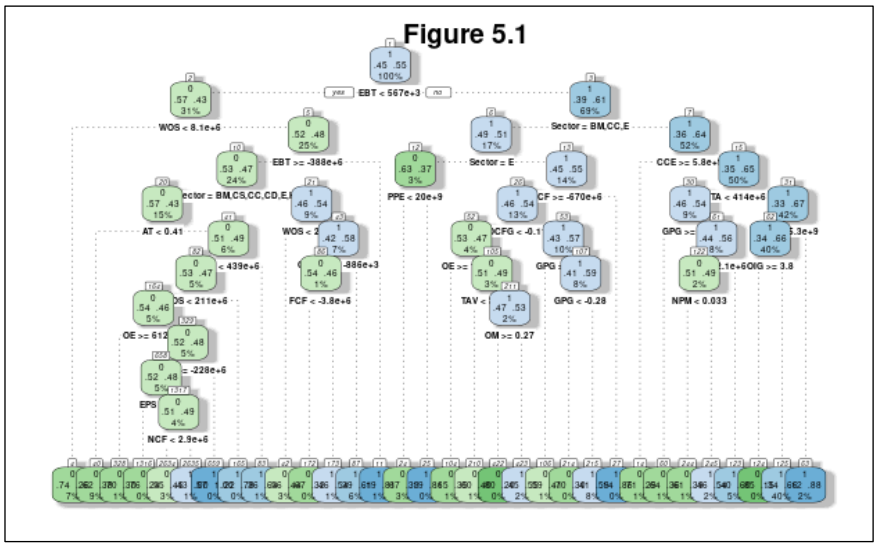
SECTION 5: Methods

After considering the binary nature of our response variable, “Class”, we have decided to explore three model options: a bagged classification forest, a gradient boosted classification tree, and a logistic regression model. The latter will be fit to predict the probability a stock experiences a “Class” value of 1, while the bagged forest and gradient boosted tree will directly predict the “Class” value of a given stock.

Bagged Classification Forest

A bagged classification forest is the aggregation of a predetermined number of classification trees. To better understand the mechanics of a forest we need to define a tree as well as the process of bootstrapping.

In its most simple form, a tree is a visual classification model. We begin at the root of our tree where all possible predictors are considered; the variable (and splitting value) that results in the lowest classification error rate (explained at length in **Section 6**) is officially selected as a leaf. From here, the tree expands to create more leaves; we repeat the process of minimizing the overall classification error rate through variable and splitting value selection until the aforementioned metric no longer decreases or plateaus. Below, Figure 5.1 provides a visual understanding of a typical tree fit to our data.



Our bagged classification forest is nothing more than a collection of trees who have been fit using varying subsets of our original data frame; to create a forest comprised of n trees, we must take n bootstrap samples. To create a bootstrap sample, we randomly sample our original data frame with replacement; a bootstrap sample is the same size as the original data frame but contains a variation of the original observations.

From here, we can fit n trees using n bootstrap samples to construct the bagged classification forest. The prediction produced by the forest is determined by calculating the appropriate binary value within each tree and choosing the most common result. For example, if 60 of our 100 trees predicted a value of 1, our forest would also predict a 1 for the observation.

Advantages of Bagged Forests[3]

- Reduce variance, thus avoiding overfitting the model
- Bootstrap sampling allows multiple weak learners to combine and outperform a singular strong learner

Disadvantages of Bagged Forests[3]

- Forests are less interpretable when compared to trees
- The reduction of variance results in an increase in bias
- Increased accuracy comes at the cost of increased computational expense

Gradient Boosted Classification Tree

Gradient boosted classification trees are similar to bagged forest in their status as an ensemble learner where "each predictor tries to improve on its predecessor by reducing the errors"[4]. Boosting differs from bagging, however, by fitting "a new predictor to the residual errors made by the previous predictor"[4] instead of fitting a predictor on the data at each iteration.

To obtain the first iteration of predictions on the data, the gradient boosting algorithm "will get the log odds of ['Class' = 1 and] … convert that value to a probability"[4]. For every observation in the training data, the algorithm "calculates the residual for that [observation and] … builds a new decision tree that tries to predict the residuals that were previously calculated"[4].

From here, we need to transform every leaf in the decision tree using Equation 1 since our base estimator is in terms of log odds, but the tree itself was built on probabilities.

$$\frac{\Sigma(Residual)}{\Sigma[PreviousProb\ *(1-PreviousProb)]}$$

Equation 1

We can then use Equation 2 to make a new iteration of predictions. "$LearningRate$ is a hyperparameter that is used to scale each trees contribution, sacrificing bias for better variance … so that we do not overfit the data"[4].

$$PreviousLog(Odds)\ +(LearningRate\ *PreviousPredictedResidual)$$

Equation 2

After obtaining new log odds predictions, we revisit Equation 1 to convert these values to probabilities and calculate new residual values. "The process is repeated until a predefined threshold is reached, or the residuals become negligible"[4].

Advantages of Gradient Boosted Trees[3]

- High predictive accuracy due to self-optimization
- Reduces bias, thus avoiding underfitting the model
- Net error is evaluated at every step and considering in descending trees

Disadvantages of Gradient Boosted Trees[3]

- Increases the complexity of classification making interpretation difficult
- Learning rate may not fully counteract high variance, thus leading to overfitting
- Increased accuracy comes at the cost of increased computational expense

Logistic Regression

Logistic regression models are much more simplistic when compared bagged forests and gradient boosted trees. Instead of actively predicting a "Class" value for each observation, logistic regression allows us to predict the log odds (and odds) of a stock experiencing a "Class" value of 1; this is important to keep in mind when predicting on the fitted model and computing the classification error rate.

The population logistic regression model for "Class" will take the following form since the variable is binary.

$$Y \sim Bernoulli(\pi_i)$$

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1(X_i)$$

*$\beta_1(X_i)$ *represents all potential parameters and variables that will be appropriately featured*

Since our logistic regression only predicts the probability of a "Class" value of 1, we need to set a threshold value to convert our prediction to either a 1 or 0. While a threshold of 0.5 is usually the intuitive choice for a binary variable, we would like to avoid our model predicting a true "Class" value of 0 as a 1. Afterall, it is more favorable to miss out on earning profit than to take a loss; therefore, we will bump the threshold value to 0.75. In other words, any observation with a predicted log odds greater than 0.75 will be assigned a predicted "Class" value of 1 and those below 0.75 will be predicted as "Class" = 0.

Advantages of Logistic Regression[5]

- Easy implementation and simplistic interpretations
- Predicted parameter values provide inference about the respective feature's significance
- Low computational expense

Disadvantages of Logistic Regression[5]

- Prone to overfitting in high dimensional data frames
- Time consuming EDA process makes it difficult to capture complex relationships

Section 6: Results

Prior to beginning model fitting, we need to ensure we have an appropriate method and metric to assess and compare the predictive accuracy of each proposed model. Our metric of interest will be the Classification Error Rate (CER): the total number of incorrect predictions divided by the sample size. Ideally, our models will produce CERs lower than 50% as this would indicate they are improvements upon simply guessing at random.

To produce these CERs we will divide our data frame into two subsets; the training data (random 80% of our original data frame) will be used to fit the models while the test data (remaining 20% of original data frame) will be used to predict.

Each model is capable of producing both a training and test CER. The training CER is computed by predicting the "Class" of every stock contained in the data frame the model itself was built on (training data). The test CER is computed by predicting the "Class" of every stock contained in a foreign data frame (test data), thus making it a more accurate overall measure of the model's predictive ability.

Bagged Classification Forest

Fitting the data to a bagged forest comprised of 1000 trees resulted in a test CER value of 36.61% and a training CER value of 37.28%. These are extremely promising values as they indicate our bagged classification forest is predicting the "Class" outcome correctly for a stock 63.39% of the time (based on test CER). We can gather deeper information from the model and further our understanding of "Class" by investigating variable importance.

Prior to any interpretation, it is important to understand what we mean by "importance". As we know, the process of creating a bagged classification forest will result in the calculation of a Test CER value. In order to determine a feature's importance, we isolate all values of that feature in our data set and assign them to new observations. Following this shuffle, we reconstruct our forest and obtain a new Test CER value and compare it to the original value in terms of percentage change. In other words, the larger the importance value, the more important and relevant the feature.

With this in mind, Table 6.1 indicates the 20 most important features in our model along with their respective importance values. Comparing this variable list with the splitting features used in Figure 5.1 yields a certain level of consistency.

Table 6.1

Variable	Importance	Variable (contd.)	Importance (contd.)
Sector	21.274	EBT	6.284
CCE	19.738	ICF	4.603
DEPS	13.828	EBIT	3.850
EPS	13.668	SE	3.274
FinCF	12.522	TAV	3.253
SGA	9.358	NI	2.901
QIR	7.583	GP	2.530
NCF	7.225	DDA	1.432
RE	7.217	CI	1.287
WOS	7.212	OE	1.146

Gradient Boosted Classification Tree

Fitting the data to a gradient boosted classification tree using 10-fold cross validation produced a test CER value of 36.26% and a training CER value of 32.86%. Additionally, our gradient boosting algorithm used a learning rate of 0.3 and iterated 100 times. Once again, these are extremely promising CER values as they indicate our model is predicting the "Class" outcome correctly for a stock 63.74% of the time (based on test CER). Additionally, Table 6.2 displays the 20 most important features involved in our gradient boosted decision tree, as well as their importance values.

Table 6.2

Variable	Importance	Variable (contd.)	Importance (contd.)
EPS	0.1017	QIR	0.03577
NPM	0.06667	TA	0.03458
WOS	0.05559	FCF	0.02966
CCE	0.04883	OCFG	0.02573
Sector	0.04780	GM	0.02421
DEPS	0.04644	RE	0.02374
GPG	0.04453	PPE	0.02321
NCF	0.04208	OIG	0.02307
FinCF	0.03960	TL	0.02083
OCF	0.03655	SGA	0.01991

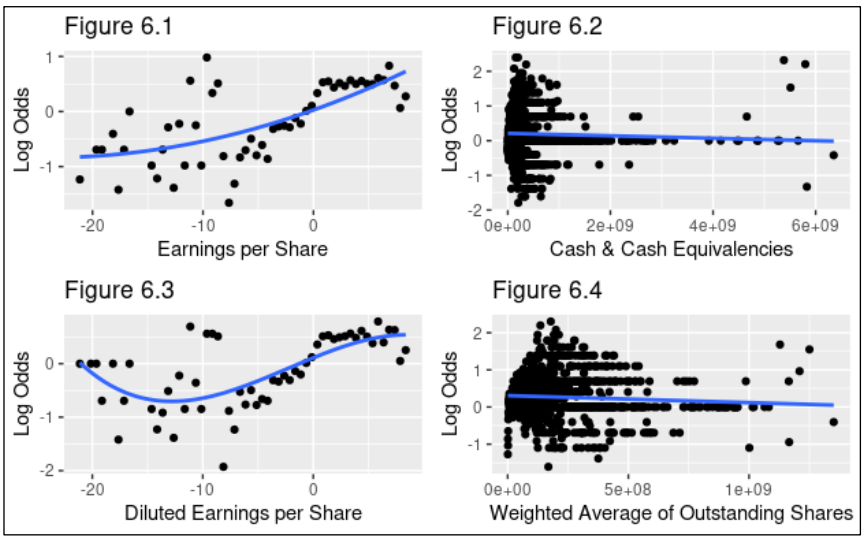
In terms of predictive accuracy (test CER), the bagged forest and gradient boosted decision tree are extremely similar with a slight lean towards the latter. Interestingly, despite their equivalent

predictive ability, the models only share 9 (highlighted) variables in common among their 20 most important listed in Tables 6.1 and 6.2, respectively.

Logistic Regression

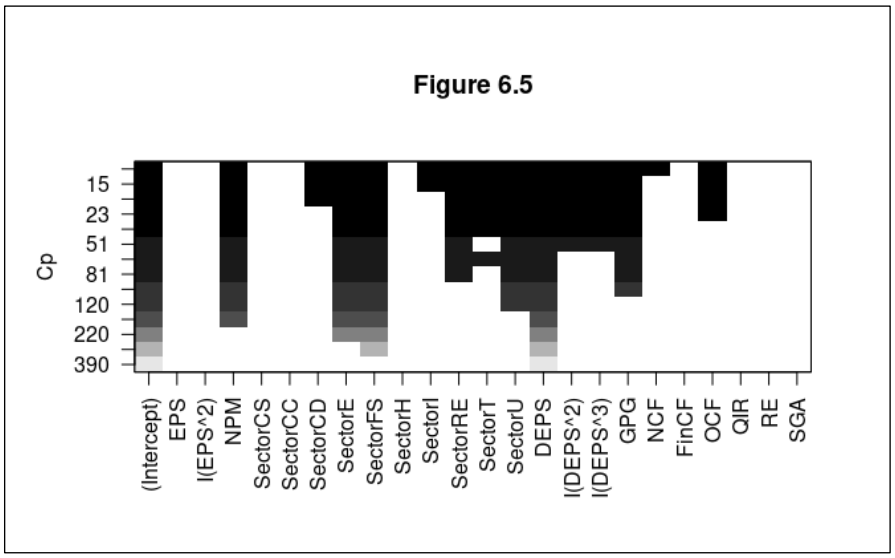
Prior to fitting a model, we need to revisit EDA as we need to take a slightly different approach when it comes to logistic regression. Particularly, we need to investigate the relationships between the log odds of a "Class" value of 1 and our explanatory variables. As highlighted in **Section 5**, this process can be extremely time consuming; we will only focus on EPS, CCE, DEPS, WOS, and Sector (5 of the 9 reoccurring variables with high importance values from our previous models).

To investigate "Sector", we can revisit Figure 4.1 and note an obvious difference in the distribution of "Class" among the levels of the explanatory variable; "Sector" should be included when fitting the logistic regression model. The remaining relationships can be visualized using Figures 6.1 – 6.4, below.



In short, we can note "EPS" and "DEPS" take on polynomial relationships with "Class" (2nd and 3rd degree, respectively), while "CCE" and "WOS" seem to entirely lack a relationship with the response variable.

Now, we will fit a logistic regression model using the top 10 most important variables in both the bagged classification forest and gradient boosted tree being sure to factor in the relationships discovered in EDA. This initial fit resulted in an AIC value of 19,612 and a test CER of 53.46%. Next, we can employ a best subset selection (BSS) algorithm to identify a better fitting model in terms of the AIC using Figure 6.5, below.



After making adjustments based on Figure 6.5, the final logistic regression model incorporates the variables and slope values listed in Table 6.3, below.

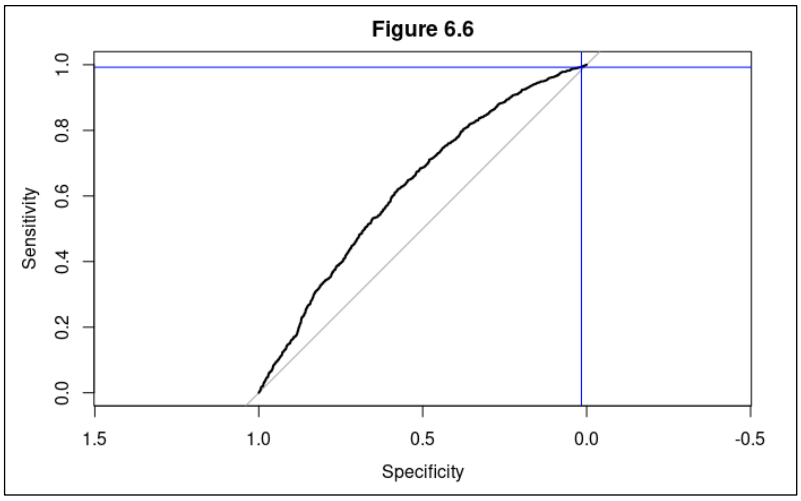
Table 6.3

Variable	Slope	Variable (contd.)	Slope (contd.)
Intercept	0.05225	SectorCD	0.2165
NPM	0.05276	SectorE	-0.5006
DEPS	0.09462	SectorFS	0.5008
$DEPS^2$	-0.005568	SectorI	0.1314
$DEPS^3$	-0.0003819	SectorRE	0.4103
GPG	-0.1631	SectorT	0.2571
NCF	$-1.525 \cdot 10^{-10}$	SectorU	0.7958
OCF	$-1.525 \cdot 10^{-11}$		

This finalized version of our logistic regression model possesses an AIC value of 19,604 and a test CER of 53.68%. In the end, both models are extremely similar; after BSS the AIC value slightly dropped, but the test CER slightly increased. These similarities lead us to support the reduced model since it features less variables but achieves similar results.

We can investigate the effects of our threshold value of 0.75 by noting the model's sensitivity and specificity values of 0.99235 and 0.01592, respectively. Sensitivity refers to the model's ability to predict true "Class" 0s as 0s, while specificity focuses on its ability to correctly predict true "Class" values of 1^6. These values align with our intention to help the model avoid incorrectly predicting true "Class" values of 0. Below, Figure 6.6 plots the trade off

between sensitivity and specificity: a threshold value of 0.75 is pinpointed in the blue cross hairs.



Section 7: Conclusion and Future Work

Table 7.1 summarizes the findings discussed in **Section 6**.

Table 7.1

Model	Accuracy Rate
Bagged Classification Forest	**63.39%**
Gradient Boosted Classification Tree	**63.74%**
Logistic Regression	**46.32%**

Based on the accuracy rates our fitted models were able to achieve we can rank the gradient boosted tree as the strongest, followed by the bagged forest, and finally the logistic regression model. As **Section 5** discussed, each of these models have unique advantages and disadvantages to consider; however, we are extremely confident in recommending our Gradient Boosted Classification Tree and Bagged Classification Forest as accurate tools to assess a stock's future price performance.

Though we are confident in our models, plenty of additional or future work could be done to enhance our results. While we logically assigned a threshold value of 0.75 to our logistic regression model, further research could explore a more optimized value that both boosts the model's CER value and keeps our concern of incorrectly predicting true "Class" values of 0 in

mind. Additionally, our logistic regression EDA is not as thorough as it could be; provided more time and resources, we could create empirical logit plots for every explanatory variable to help capture more complex relationships within the data. Finally, transitioning our bagged forest and gradient boosted tree algorithms to a more powerful computer would allow us to generate more trees and iterations: a process that would undoubtedly lead to more accurate models.

In summary, the above report is a solid foundation to continue to explore the predictive abilities of modeling stock price performance. We were able to generate complex and highly accurate models, better understand variable relationships to “Class”, and truly feel confident in extending these model applications to the real world. Further research and future work would likely serve to bolster any conclusions and findings we have made.

Variable Index[7]

Response

- **Class:** indicates positive or negative growth on the year
    - Positive (1)
    - Negative (0)
- **PriceVar:** numeric representation of positive or negative growth on the year

Explanatory

- **Stock:** official stock ticker symbol
- **Revenue (Rev):** money generated from normal business operations, calculated as the average sales price times the number of units sold
- **Gross Profit (GP):** the profit a company makes after deducting the costs associated with making and selling its products, or the costs associated with providing its services
- **Selling, General, and Administrative Expense (SGA):** includes all general and administrative expenses as well as the direct and indirect selling expenses of the business
- **Operating Expense (OE):** an expense a business incurs through its normal business operations
- **Operating Income (OI):** an accounting figure that measures the amount of profit realized from a business's operations, after deducting operating expenses
- **Earnings Before Tax (EBT):** a calculation of a firm's earnings before taxes are considered.
- **Net Income (NI):** calculated as sales minus cost of goods sold, selling, general and administrative expenses, operating expenses, depreciation, interest, taxes, and other expenses
- **Earnings per Share (EPS):** calculated as a company's profit divided by the outstanding shares of its common stock
- **Diluted EPS (DEPS):** a calculation used to gauge the quality of a company's EPS if all convertible securities were exercised. Convertible securities are all outstanding convertible preferred shares, convertible debentures, stock options, and warrants.
- **Weighted Average of Outstanding Shares (WOS):** a calculation that a company uses to reflect any changes in the number of the company's outstanding shares over a reporting period
- **Gross Margin (GM):** equates to net sales minus the cost of goods sold
- **Operating Margin (OM):** measures how much profit a company makes on a dollar of sales after paying for variable costs of production, such as wages and raw materials, but before paying interest or tax

- **Earnings Before Interest, Taxes, Depreciation, and Amortization (EBITDA):** a measure of a company's overall financial performance
- **Earnings Before Interest and Tax (EBIT):** a company's net income before income tax expense and interest expenses are deducted
- **Consolidated Net Income (CI):** the sum of net income of the parent company excluding any income from subsidiaries recognized in its individual financial statements plus net income of its subsidiaries determined after excluding unrealized gain in inventories
- **Pretax Profit Margin (PPM):** measures how much EBT is generated as a percentage of revenues received
- **Net Profit Margin (NPM):** measures how much net income is generated as a percentage of revenues received.
- **Cash and Cash Equivalents (CCE):** reports the value of a company's assets that are cash or can be converted into cash immediately
- **Property, Plant, and Equipment Net (PPE):** net value of the long-term assets vital to business operations and the long-term financial health of a company
- **Total Assets (TA):** combined value of assets with economic value owned by an entity
- **Total Liabilities (TL):** the combined debts and obligations that an individual or company owes to outside parties
- **Retained Earnings (RE):** the amount of net income left over for the business after it has paid out dividends to its shareholders
- **Shareholder Equity (SE):** the corporation's owners' residual claim on assets after debts have been paid. Equal to a firm's total assets minus total liabilities
- **Depreciation, Depletion, and Amortization (DDA):** depreciation spreads out the cost of a tangible asset over their lifetime, depletion allocates the cost of extracting natural resources, and amortization is the deduction of intangible assets over a specified time period
- **Operating Cash Flow (OCF):** a measure of the amount of cash generated by a company's normal business operation
- **Capital Expenditure (CapEx):** funds used by a company to acquire, upgrade, and maintain physical assets such as property, plants, buildings, technology, or equipment
- **Investing Cash Flow (ICF):** any inflows or outflows of cash from a company's long-term investments
- **Financing Cash Flow (FinCF):** the net flows of cash that are used to fund the company. Financing activities include transactions involving debt, equity, and dividends
- **Net Cash Flow (NCF):** either the gain or loss of funds over a period after all debts have been paid
- **Free Cash Clow (FCF):** represents the cash a company generates after accounting for cash outflows to support operations and maintain its capital assets
- **Asset Turnover Ratio (AT):** measures the value of a company's sales or revenues relative to the value of its assets
- **Quality of Income Ratio (QIR):** the proportion of cash flow from operations to net income. It refers to the amount of earnings that come from the business operations themselves

- **Tangible Asset Value (TAV):** total value of all tangible assets. Tangible assets are assets that have a finite monetary value and usually a physical form
- **Gross Profit Growth (GPG):** the change in gross profit on a year-on-year basis expressed as decimal percentage
- **Operating Income Growth (OIG):** the change in operating income on a year-on-year basis expressed as decimal percentage
- **Operating Cash Flow Growth (OCFG):** the change in operating cash flow on a year-on-year basis expressed as decimal percentage
- **Sector:** an area of the economy in which businesses share the same or related business activity, product, or service
    - Basic Materials (BM)
    - Communication Services (CS)
    - Consumer Cyclical (CC)
    - Consumer Defensive (CD)
    - Energy (E)
    - Financial Services (FS)
    - Healthcare (H)
    - Industrials (I)
    - Real Estate (RE)
    - Technology (T)
    - Utilities (U)

Works Cited

1. Carbone, N. (2020, January). 200+ Financial Indicators of US Stocks (2014-2018). Retrieved August 2021 from https://www.kaggle.com/cnic92/200-financial-indicators-of-us-stocks-20142018

2. Carbone, N. (2020, January). Explore and Clean Financial Indicator Data Set [Code Book]. https://www.kaggle.com/cnic92/explore-and-clean-financial-indicators-dataset

3. Ragini, Ruhi. "Bagging and Boosting Method". *Medium*, 6 June 2019, https://medium.com/@ruhi3929/bagging-and-boosting-method-c036236376eb

4. Aliyev, Vagif. "Gradient Boosting Classification Explained Through Python". *Towards Data Science*, 5 September 2020, https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d

5. Ranjan Rout, Amiya. "Advantages and Disadvantages of Logistic Regression". *Geeks for Geeks*, 2 September 2020, https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/?ref=lbp

6. Dvorak, Tomas. "Lab 16: Logistic Regression and ROC Curves". *RPubs by RStudio*, 2016, https://rpubs.com/dvorakt/255527

7. Investopedia. "Financial Term Dictionary". *Investopedia*, https://www.investopedia.com/financial-term-dictionary-4769738

R Packages

- Sam Firke (2021). janitor: Simple Tools for Examining and Cleaning Dirty Data. R package version 2.1.0. https://CRAN.R-project.org/package=janitor

- Terry Therneau and Beth Atkinson (2019). rpart: Recursive Partitioning and Regression Trees. R package version 4.1-15. https://CRAN.R-project.org/package=rpart

- Max Kuhn (2021). caret: Classification and Regression Training. R package version 6.0-88. https://CRAN.R-project.org/package=caret

- Williams, G. J. (2011), Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery, Use R!, Springer.

- A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18--22.

- Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

- Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng and Yutian Li (2021). xgboost: Extreme Gradient Boosting. R package version 1.4.1.1. https://CRAN.R-project.org/package=xgboost

- H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

- Thomas Lumley based on Fortran code by Alan Miller (2020). leaps: Regression Subset Selection. R package version 3.1. https://CRAN.R-project.org/package=leaps

- Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez and Markus Müller (2011). pROC: an open-source package for R and S+ to analyze and compare ROC curves. BMC Bioinformatics, 12, p. 77. DOI: 10.1186/1471-2105-12-77 http://www.biomedcentral.com/1471-2105/12/77/