Final Report
ISyE 6740 – Spring 2021
Project Title: Find the stock winners!

Team Member Names:
Klaus Smit – GTID# 902852825
Emilio Flores Braeckow – GTID# 903468849
Andres Urrutia – GTID# 903399102

1. Problem Statement

In today's world given the recent surge of high-frequent trading and the media's influence in the stock market (such as the recent spike of Gamestop (GME) stock caused by a group of amateur traders in reddit), it would seem as if fundamental data is no longer relevant for evaluating the long-term performance of a company's stock price. The hypothesis we are interested in testing in this project is that the fundamental data of a company, such as Revenue, Operational Cost and Cash Flow, are still very relevant to predict the valuation of a company in the long term despite the increasing market noise described earlier.

The goal of our project is to build a bagging model that combines the prediction power of multiple individual models to classify a list of stocks as "winners" or "losers". A winner stock is one where the price of that symbol increased at the end of a one-year timeframe while a loser stock would be the complete opposite – the price decreased at the end of the same period. We will build that bagged model using only the fundamental data of each of those companies being studied.

2. Introduction

There are various established techniques investors have traditionally used to evaluate stocks and predict future price movements. The most commonly used methods fall under either of these two categories: technical analysis and fundamental analysis.

Technical analysis is the "study of the market itself as opposed to the study of the goods in which markets deal." This type of analysis focuses on changes in price, volume and related statistics, with a forward-looking nature through the inferences gathered with technical indicators, developed through heuristics or mathematical calculations. (Čelebić, 2020)

Fundamental analysis, on the other hand, evaluates a stock's intrinsic values using publicly available information. It uses a broad number of factors from the overall economy in relation to industry performance and a company's financial factors such as earnings, profit margin, assets. These financial factors will become the variables from which our models to classify the stocks will be built.

The mathematical relationship that exists between the different variables that will be explored, which are going to be built in the following experiments, are all machine learning models. "Machine learning refers to creating and using models that are learned from data." (Grus, 2019). Within the Machine Learning field there are two major types of algorithms: supervise learning and unsupervised learning. Since we are using historical data, we know the output of each of our observations (also known as the labels) and therefore a supervised learning approach would make more sense to solve the problem and, thus, it is what we decided to implement in this project.

Making predictions on the market data is a complex problem, due to the noise found and the number of possible factors. This is better characterized in the Efficient Market Hypothesis (EMH). Which states that the market is extremely efficient in reflecting individual information about individual stocks and about the stock market as a whole. EMH states then that stock performance is therefore impossible to predict as future price changes represent random departures from previous prices as information arrives randomly and prices adjust quickly.

There are three versions of Efficient Market Hypothesis:

1. Weak Form: Future prices cannot be predicted by analyzing historical prices. Therefore, technical analysis cannot predict future stock performance.
2. Semi-strong Form: Prices adjust rapidly to new public information. Therefore, fundamental analysis cannot predict future stock performance.
3. Strong Form: Prices reflect all information, public and private.

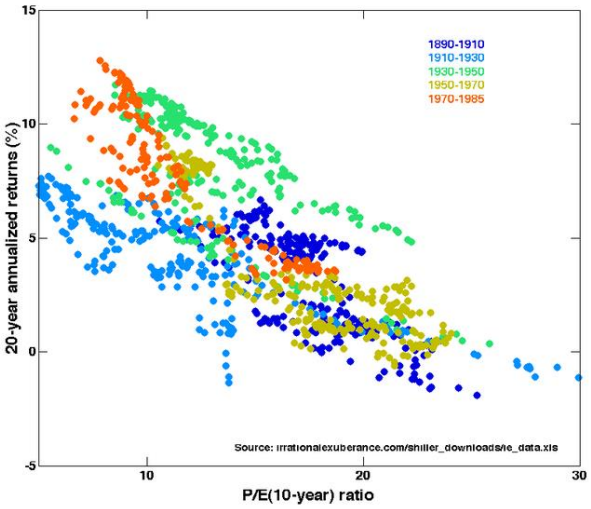

Figure 1: Price/Earnings Ratios as a Predictor of 20-Years Returns (Shiller, 2005)

The basis of classifying particular companies' stocks in terms of "winners" and "losers" with fundamental information, requires that at least the semi-strong form of the EMH does not hold. Figure 1 refutes the semi-strong form, as each dot is the Price/Earnings ratio on a date and lower price and higher earnings is better and

should result in higher value for a stock. This shows that P/E ratios, a stock fundamental, are very predictive across many decades of future returns.

3. Methodology

The flowchart below summarizes the methodology that we followed to collect/pre-process the source data and to build the required models:

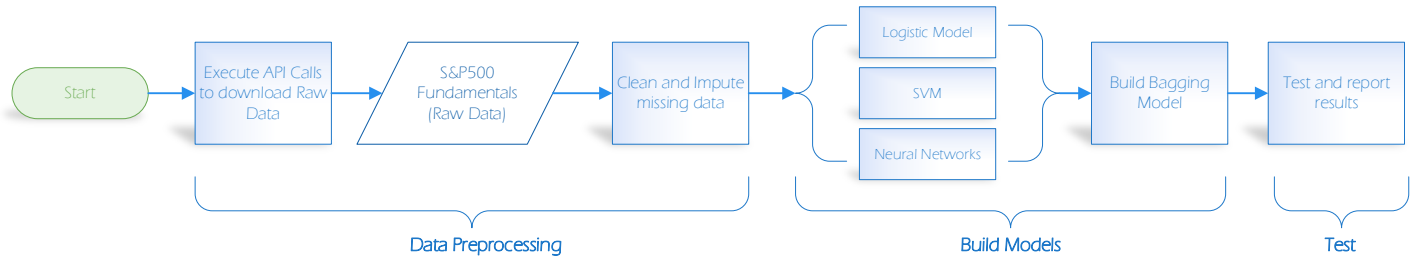

Figure 2: Project methodology

Expanding on that graph, here is the detailed description of each of the steps we followed to find the stock winners:

1. **Collect Raw Data**
    a. Defined universe of stocks based on the S&P 500 list of companies since those are the companies that would most likely be influenced by their fundamental data.
    b. Determined the list of features to be collected based on the most common fundamental values that are used to valuate public companies.
    c. Created a Python script that used the Alpaca.Markets SDK to generate the necessary API calls and obtain the list of features determined on the previous step for each of the selected companies at timeframe $X^{(1)}$.
    d. Obtained the adjusted stock price of the same list of stocks at future time $X^{(2)}$ with new API calls in Python.

2. **Pre-Process and Clean Data**
    a. Identified and determined the percentage of missing values for each of the features.
    b. Replaced the stocks that have more than 50% of the data missing.
    c. For each feature, imputed the missing values using methods such as:
        i. Mean, mode, median.
        ii. Random values.
        iii. Regression using another feature of the dataset.
    d. Split the data and used 400 of those observations as training data (about 80%) and the other 100 (20%) to test and validate the models.

3. **Classify Data**
    a. Generated a label with options {*Winner*, *Loser}* for each randomly sampled stock. A *Winner* label was given to the stocks that had a higher adjusted stock price at time $X^{(2)}$ when compared to time $X^{(1)}$, else the observation was labeled as a *Loser*.

4. **Build and Validate Models**
    a. Used the cleaned dataset to build the following 3 supervised classification models: SVM, Logistic Regression and Neural Networks.
    b. Applied PCA to identify the principal components and re-trained models to see if there was any improvement in accuracy.
    c. Compared the models and selected the model that best classified the stock data into the labels *{Winner, Loser}* using metrics such as F1 Score, Precision, Recall and Accuracy.
    d. Cross-validated the model to quantify the overfitting in each of the models.
    e. Performed hyperparameter tuning on the best performing model.

5. **Create Bagging Model**
    a. Assembled the prediction results of different models with the goal of improving classification accuracy.

4. Data

4.1 Data Source

The input data for our model would consist of a matrix where each row corresponds to a company and columns representing the features that correspond to the fundamental data of that company. Our training data will have a "label" column with a 1 for a company classified as a "winner" (the stock price went up in the respective time period) or a 0 if the company is classified as a "loser" (the stock price went down in that same time period).

One of the challenges of the project was to collect, clean/process and organize the data in the matrix format described above. One potential way to gather this data is to use websites such as Yahoo Finance or Investing.com to manually search each of the companies being evaluated and capture the data into the required format. Another way to obtain this data is by leveraging API services offered by companies such as Alpaca (https://alpaca.markets) where we can automatically collect those fundamental values directly from our algorithm using HTTP requests. Our decision was to go with the latter option of building a Python script to automatically collect that data for us.

4.2 Data Features

The features to be collected per company are (Definitions from Investopedia.com):

Feature Name	Description
Book Value Per Share (BVPS):	Takes the ratio of a firm's common equity divided by its number of shares outstanding.
Cash and Cash Equivalents (CCE)	Reports the value of a company's assets that are cash or can be converted to cash immediately
Debt to Equity Ratio (D/E)	Used to evaluate a company's financial leverage
Earnings per Basic Share (EPS)	How much of a firm's income was allotted to each share of common stock
Free Cash Flow per Share (FCF)	Is a measure of a company's financial flexibility, determined by dividing free cash flow by the total number of shares outstanding
Gross Profit	Profit a company makes after deducting the costs associated with making and selling their products
Net Cash Flow	Calculated by subtracting a company's total liabilities from its total cash
Net Income	How much profit a company made after paying all expenses
Operating Expenses	Operating cost of a business
Research and Development Expense (R&D)	Expenses associated with research and development of a company's goods or services
Revenue	is the income generated from normal business operations and includes discounts and deductions for returned merchandise
Sales per Share	Total sales earned per share
Total Liabilities	Combined debts and obligations that an individual company owes
Profit Margin	Percentage of sales that has turned into profit

4.3 Missing Data

It was expected that the collected data was going to have missing entries and we had to address those issues to avoid any issues that could occur when training each of the classification models. We used the following strategies to properly pre-process and clean the data:

1. Delete and Replace: Any stock that had 50% of its features missing was replaced by another randomly selected stock.
2. Mean Substitution: Replaced missing values with the Mean value for the column/feature.
3. Regression estimation: Used available data from other features to compute a regression model in order to provide a prediction for the missing feature.

4.4 Data Example

All of the data features where collected for two years, being year 1 (2018) the timeframe $X^{(1)}$ and year 2 (2019) the timeframe $X^{(2)}$. Then, the delta of these data points was calculated and the resulted dataset is what was used to build our models.

Here is an example of the datasets showing only 3 features for 3 of the observations/stocks:

Year 2 (2019)

Stock	bookValuePerShare	cashAndEquivalents	debtToEquityRatio	earningPerBasicShare	salesPerShare
BK	44.149	114683000000	8.191	4.53	17.52
LIN	90.694	2700000000	0.713	4.22	52.168
IT	10.45	280836000	6.619	2.6	47.266

Year 1 (2018)

Stock	bookValuePerShare	cashAndEquivalents	debtToEquityRatio	earningPerBasicShare	salesPerShare
BK	40.52	88000000000	7.924	4.06	16.344
LIN	156.162	4466000000	0.703	13.26	45.097
IT	9.367	156368000	6.289	1.35	43.77

Delta: Year 2 (2019) – Year 1 (2018)

Stock	bookValuePerShare	cashAndEquivalents	debtToEquityRatio	earningsPerBasicShare	salesPerShare	Label
BK	3.629	26683000000	0.267	0.47	1.176	0
LIN	-65.468	-1766000000	0.010	-9.04	7.071	1
IT	1.083	124468000	0.330	1.25	3.496	1

This final dataset also includes the label column that determines whether the stock price increased or decreased during the 1-year period. This is the format of the final dataset and what will be used for the rest of the project.

4.5 Data Preparation

As seen from data samples above, the scales of the selected features can vary highly and therefore different data transformation options were explored. Data standardization can be in the form of normalization or standardization. To choose the correct data preparation technique to use, prediction accuracy was calculated using the original data and then that value was compared to the accuracy of the models using normalized data and standardized data. Normalization was performed through Max-Min normalization. Figure 2 below shows that standardized data provided the best performing models. This may be because the standardization method is more robust to outliers.

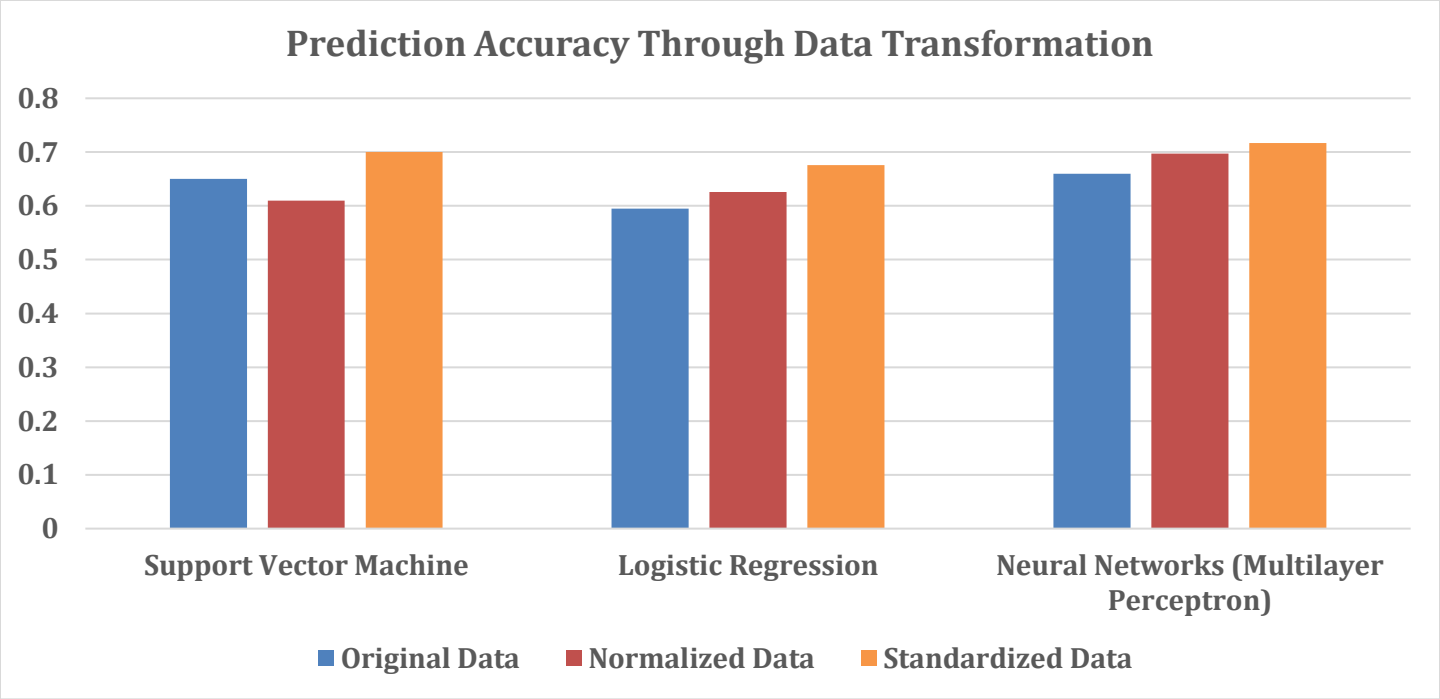

Figure 3: Prediction Accuracy Through Data Transformation

Standardization was applied to all features not only to improve the performance of the classification models but also to avoid violating the assumptions made by some of the models we built. For example, Support Vector Machine and Logistic Regression models both assume that all the features in the dataset are centered around zero and have a variance in the same order.

This is the following formula that was used to standardize our data:

$$x^i = \frac{x^i - \bar{x}}{\sigma}$$

Where x^i is the original feature value, $\bar{x}$ is the mean of the feature vector and σ is the standard deviation.

4.6 Data Splitting: Training vs Testing

Data splitting is necessary to prevent a model from seeing all available data causing overfitting. To build more generalized models we split the dataset into training data and testing data. Our training data corresponds to 80% randomly selected observations from the data and the test data corresponds to the remaining 20%.

4.7 Principal Component Analysis

Principal component will also be explored per machine learning model and will be used to compare model prediction as well as converge time benefits that could result from transforming the features into the principal components that explain most of variability of the variable we are looking to predict. Another reason why we decided to use PCA was to eliminate any potential correlation that might exist between the features since many

of them are ratios that might come from the same fundamental data for each of the companies. Applying PCA would, therefore, generate the principal components that explain the data without any correlation between them.

Figure 4 below shows the variability of the prediction variable as explained by each principal component. It is clear that the first couple of components are capable of explaining most of the variability found. Therefore, there is potential for PCA to provide an increase in model performance since the reduction of the dimensions of the data could improve the convergence speed of each of the algorithms. In addition, the reduction of noise can increase the prediction power of each of the models.

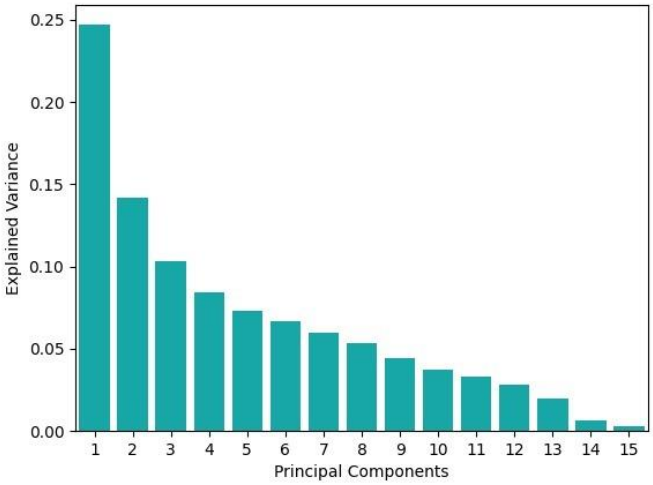

Figure 4: Variability Explained by Each Principal Component

5. Model Evaluation

5.1 Logistic Regression

Data: $\{(x_{11}, \dots, x_{1p}), Y_1\}, \dots, \{(x_{n1}, \dots, x_{np}), Y_n\}$ where $Y_1, \dots, Y_n$ binary responses
Model: Probability of success given predictors

$$p = p(x_1, \dots, x_p) = P\{Y = 1|1x_1, \dots, x_p\}$$

Link p to the predicting variables through a nonlinear link function:

$$g(p) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p\, x_p$$

We furthermore link the probability of success to the predicting variables using the g link function, in a way that this g function of the probability of success is a linear model of the predicting variables. The model has a linear relationship to the predicted variables, plus an error term.

Hyperparameter tuning for best model performance can be performed on:

- **Solver:** Algorithm to be used in the optimization problem.
- **Penalty:** Regularization to be performed on the predictor variables.
    - Ridge Regression: Applies a penalty to the vector of regression coefficients, which is equal to the sum of squared regression coefficients to be penalized, does not perform variable selection.
    - Lasso Regression: The penalty applied is the sum of the absolute values of the coefficients, performs variable selection.
    - Elastic-Net: Applies both Ridge and Lasso penalties.

Without any hyperparameter tuning, the accuracy of the logistic model is at 66.66%. Given the high number of features and the suspicion of correlation between them, there is potential for hyperparameter tuning to result in a more robust model with a higher prediction power.

Hyperparameters where then explored with Ridge and Lasso regression. For Ridge, 3 solvers were explored and 8 C values therefore 24 models were generated. For Lasso, 1 solver with 15 C values therefore 15 models were generated. Each combination of parameters was cross validated with 5 data splits. Where the best results are:

Accuracy	Penalization	C value	Solver
67.16%	Ridge Regression	0.001	Liblinear
66.65%	Lasso Regression	0.2	Liblinear
66.14%	Lasso Regression	0.3	Liblinear
66.14%	Ridge Regression	0.01	Lbfgs
66.14%	Ridge Regression	0.01	Newton-cg

From table below we can see that the Logistic Regression model with both Lasso and Ridge penalization and low C values provide similar high accuracy values. Lasso penalization was chosen to continue exploring hyperparameter tuning in order to help simplify the model by reducing the number of features used.

After selecting Lasso as the penalization to be applied to the logistic regression model. A more robust list of lambdas was generated in order to further explore the optimization of the model parameters. This new list was cross-validated with 5 data splits. The results can be seen on *Figure 4*, which shows the solution path of each feature through Lasso penalization different lambda values which shows that some features were forced to zero. Along with how the optimal lambda value for Lasso penalization was reached, where a lambda value of 0.4 resulted in model prediction accuracy of **73.73%.**

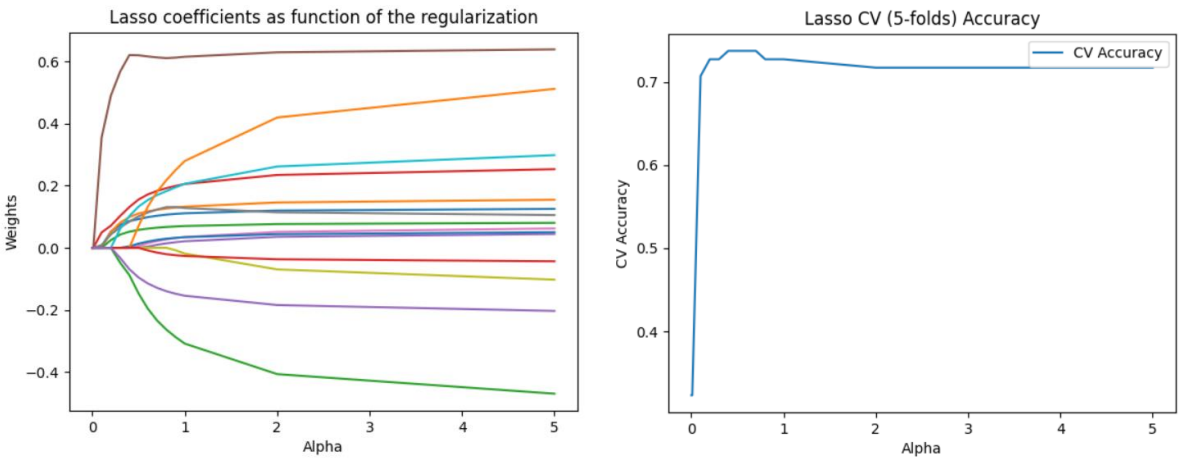

Figure 5: Solution Path & Accuracy Through Alphas

The following tables show the value of each predictor variables. Which states that Gross Profit and higher investment in R&D resulted in the variables with the most influence of which companies will have a higher stock value in a future. Indicating that companies with higher gross earnings and higher investment will result in "Winner" companies.

Feature Name	Value
Intercept	0.6662
Book Value Per Share (BVPS):	0.0859
Cash and Cash Equivalents (CCE)	0.099
Debt to Equity Ratio (D/E)	0.0518
Earnings per Basic Share (EPS)	0.1310
Free Cash Flow per Share (FCF)	0
Gross Profit	0.6200
Net Cash Flow	0

Feature Name	Value
Net Income	0.0841
Operating Expenses	0
Research and Development Expense (R&D)	0.1019
Boolean for R&D Present	0.00453
Revenue	0
Sales per Share	-0.0877
Total Liabilities	0
Profit Margin	-0.0688

Lastly the classification report and confusion matrix are displayed below showing that all "Losers" were correctly classified. However, as the data is not exactly distributed between "Losers" and "Winners", then the misclassification of "Winners" has a higher weight on the overall accuracy of the model.

	Precision	Recall	F1-Score
0	1	0.19	0.32
1	0.72	1	0.84
Accuracy			0.74
Macro avg.	0.86	0.59	0.58
Weighted avg.	0.81	0.74	0.67

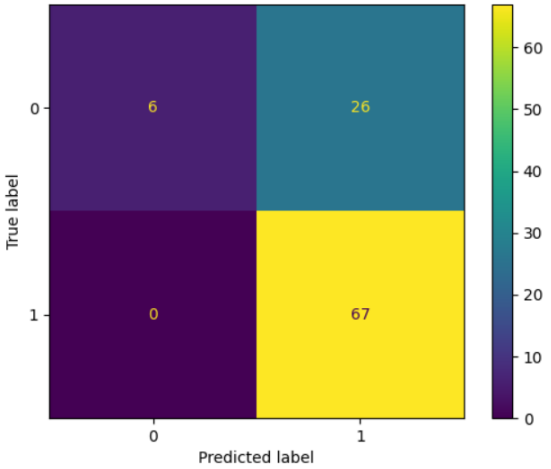

Figure 6: Logistic Regression - Confusion Matrix

5.2 Support Vector Machine

Support Vector Machine construct a classifier or hyperplane that separates the data points; the classifier is defined by the support vectors or respective data points where the Lagrange multipliers are non-zero in the process of minimizing the Lagrange function, which in turn maximizes the margin between the decision boundaries. The objective is to minimize the sum of the squares of the coefficients. Expressed as:

$$Minimize \sum_{j=1}^{n} \max\{0,1 - (\sum_{i=1}^{m} a_i x_{ij} + a_0) y_i\} + \lambda \sum_{j=1}^{m} (a_i)^2$$

Where a_i are the coefficients of the predictor factors and λ is the regularization parameter which serves as a degree of importance that is given to misclassifications.

Hyperparameter tuning for best model performance can be performed on:

- **Kernel:** Kernel to be used in the algorithm in order to transform the hyperparameter form a linear to a non-linear classifier.
- **λ Parameter (0.1 to 100):** Penalty for each misclassified data point. If small the model focuses too much on the training set which can lead to overfitting. As lambda increases more misclassification are allowed which improves the model performance with test data at the expense of making the model too general, increasing bias.
- **Gamma (0.0001 to 10):** Parameter that influences the kernel construction, for example for radial basis function it controls the distance of influence of a single training point.

To hypertune the model the following parameters were considered:

- C: 0.01, 0.1, 1, 10, 100, 1000
- Gamma: scale, auto
- Kernel: rbf, poly, sigmoid

The best estimators for the model were C = 0.1, gamma = auto, kernel = sigmoid

	Precision	Recall	F1-Score	Support
0	1.00	0.06	0.12	32
1	0.69	1.00	0.82	67
Accuracy			0.70	99
Macro Avg	0.85	0.53	0.47	99
Weighted Avg	0.79	0.70	0.59	99

Accuracy as function of C = 1/ λ

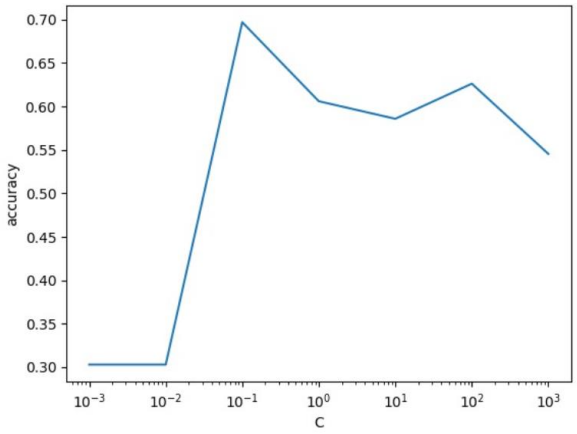

Figure 6: Accuracy vs C

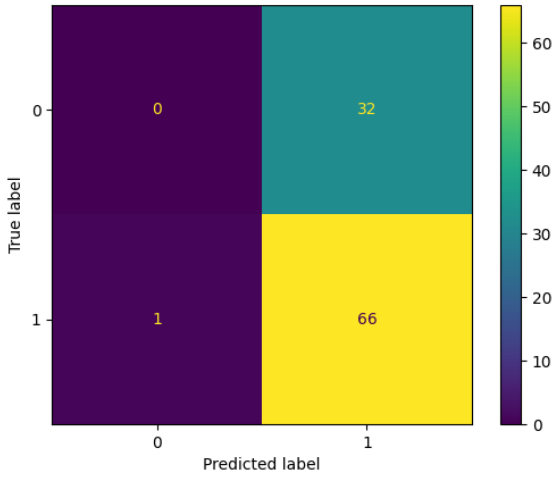

Figure 7: Confusion Matrix

5.3 Neural Networks (Multilayer Perceptron)

A Multilayer Perceptron is a form of neural networks architecture which consist of at least three layers: an input layer, a hidden layer and an output layer. A weight (W_l) is assigned to each of the connections between a neuron in one of the layers and all neurons of the previous layer. Then, take all activations (a_l) from previous layers and their weighted sum is calculated $a_{l+1} = (W_l a_l)$ to calculate the activation of this node. To restrict the activation values into {0,1} values, then a sigmoid function $\sigma()$ is applied. Lastly, a bias (b_l) is applied to make sure that a neuron is active only after a previously specified threshold.

That summarizes to:

$$a_{l+1} = \sigma(W_l a_l + b_l)$$

Gradient descent is used for backpropagation, where the weight of each connection throughout all layers is updated. The goal is to compute the partial derivates $\frac{\partial C}{\partial \varepsilon}$ and $\frac{\partial C}{\partial W}$ of the cost function with respect to any weight. Resulting in:

$$\Delta W_l = -\gamma \frac{\frac{\partial C}{\partial \varepsilon}}{\frac{\partial C}{\partial W}}$$

Where γ is the learning rate, ε final error and W the weights of each of the nodes.

During the optimization of our model, the following hyperparameters were tuned to obtain the best possible model that accurately classified our data:

- **Number of Hidden Layers:** The number of hidden layers that will make up the neural network. The more hidden layers, the more the model adjusts to the training data. Therefore, there is potential for overfitting if not selected appropriately.
- **Number of Neurons on Each Hidden Layer:** The number of neurons on each hidden layer that will make up the neural network. The more neurons on each hidden layer, the more the model adjusts to the training data. Therefore, there is potential for overfitting if not selected appropriately.
- **Activation Function:** The activation function to calculate the probability at each neuron found in the hidden layers.
- **Alpha:** This value corresponds to the L2 penalty (regularization term) parameter
- **Solver:** There are many different solvers available to solve Neural Network algorithms including stochastic gradient descent and quasi-Newton methods.
- **Learning Rate:** Controls the rate of adjustment at which weights are updated when performing backpropagation.

Multiple "versions" of the dataset were tested to see which one provided the best accuracy using the default hyperparameters of the Neural Network model. Without any standardization or normalization, an accuracy

of **0.58** was obtained. After that, the data was normalized and the tested again using the Neural Network model and a higher accuracy of **0.61** was obtained this time. However, the best accuracy we were able to obtain was when the data was standardized resulting in a total value of **0.64**.

After determining the dataset that provided the best accuracy, we manually tested multiple combinations of hyperparameters. The chart below shows, for example, the different accuracies obtained with different values of the Alpha hyperparameter:

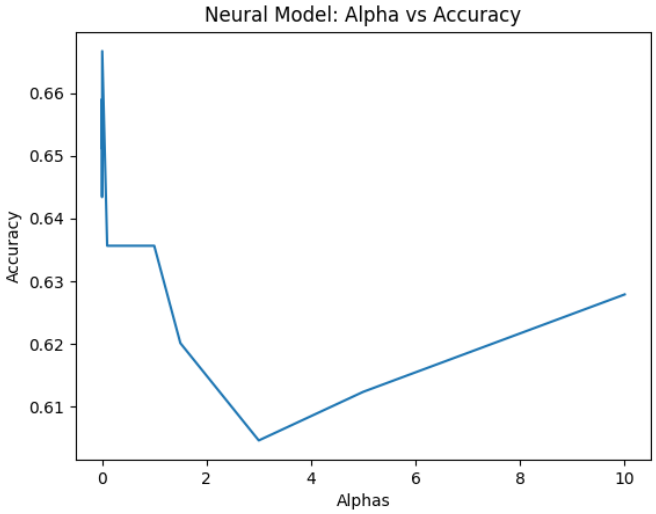

Figure 9: Neural Networks – Alphas vs Accuracy

Since there are many different hyperparameters that can be tuned and optimized, we decided to use the Sklearn's GridSearch function to find the optimal hyperparameters. Here is the list of the best hyperparameters as determined by the GridSearch function:

- **Activation:** Hyperbolic Tan Function
- **Alpha:** 0.05
- **Learning Rate:** Adaptive
- **Solver:** Stochastic Gradient Descent

After building the Neural Network model with the best parameters, we obtained a total accuracy of **0.68** and the following Confusion Matrix:

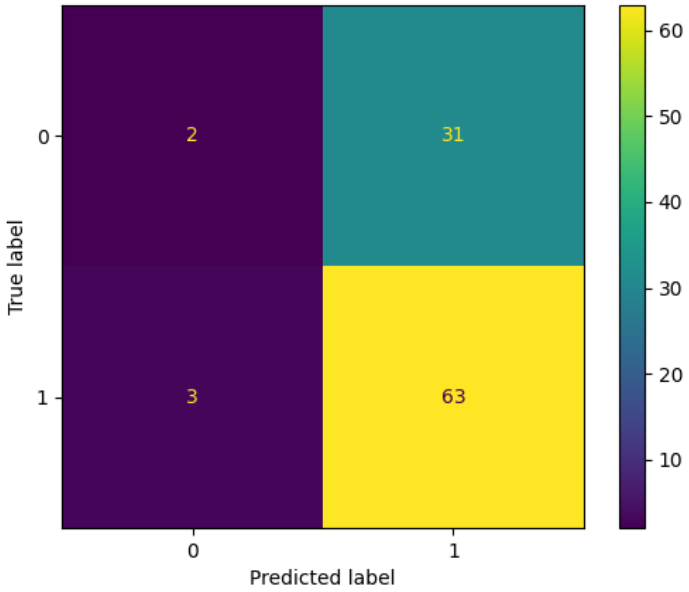

Figure 10: Neural Networks - Confusion Matrix

Here is the final classification report for the Neural Network model:

	Precision	Recall	F1-Score
0	0.40	0.06	0.11
1	0.67	0.95	0.79
Accuracy			0.68
Macro avg.	0.54	0.51	0.45
Weighted avg.	0.58	0.66	0.56

5.4 Bagging Model

After each individual machine learning model was tuned and tested, we built a bagging model that assembled the prediction results of each of the different models with the goal of improving the overall classification accuracy of our algorithm. Bagging will allow for the weakness of each model to be minimized out by leveraging the prediction power of the other models.

The class that was more commonly predicted by each of the models (either 0 or 1) is chosen for each of the stocks and the results are returned by the bagging algorithm as shown in *Figure 11* below. The diagram depicts the relationship of the models built with bagging. It is then clear how the predictive power of each model is optimized through hyperparameter tuning and aggregated through bagging.

The overall accuracy of our bagging model was **0.69.**

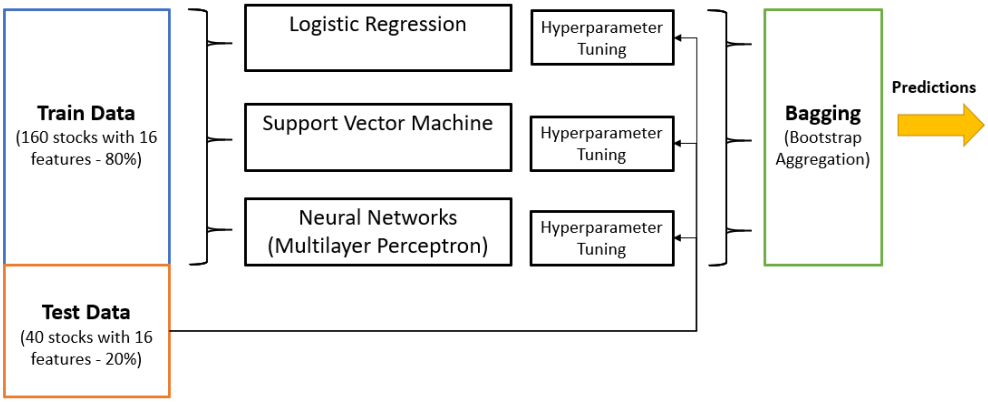

Figure 11: Description of the Bagging Algorithm implemented

6. Conclusion and Final Results

The accuracy from our classification model was 0.69. Two thirds of our data was made up of *Winners* and the remaining one third were all *Losers*. The accuracy we obtained is significantly higher than 50% (a coin toss) indicating that there is a relationship between the fundamental data of a company and the price of the stock therefore validating and confirming our initial hypotheses.

Since we have an unbalanced dataset (more *Winners* than *Losers*), our accuracy is only better than a model that predicts for all data points to be *Winners*. A good next step for this project would be to find different methods to balance the data so we can improve the confidence of a results and confirm that, in fact, our bagging model does a good job of accurately classifying winner stocks vs loser stocks.

7. Work Breakdown

All three members contributed equally with their ideas and their effort to the completion of this project. More specifically, here are activities that each of the member of our team was responsible for:

Team Member	Tasks/Work Performed
Klaus Smit	1. Data Cleanup a. Convert JSON from API call responses to a Pandas Dataframe object b. Impute Missing Data 2. Build, tune and optimize the Support Vector Machine (SVM) model 3. Compare and select the best performing model 4. Create charts for SVM and write results specific for their model
Emilio Flores Braeckow	1. Write Introduction, Methodology and Data sections of the report 2. Build/tune/optimize the Logistic Regression model 3. Compare and select the best performing model 4. Create charts for Logistic Regression and write results specific for their model
Andres Urrutia	1. Data Collection a. Research Alpaca.Markets API documentation b. Create Python code to execute API calls to download stock data. 2. Build/tune/optimize the Neural Networks model 3. Compare and select the best performing model 4. Create charts for Neural Networks and write results specific for their model

8. References

1. Grus, Joel. (2019). Data Science from Scratch, 2nd Edition.

2. Čelebić, Nedim. Halilbegovic, Sanel. (2020). Study of technical analysis indicators: relationship between profitability and signal strengths of MACD and RSI.

3. Shiller, Robert. (2005). (Figure 10.1 from Shiller, Robert (2005) Irrational Exuberance (2d ed.), Princeton University Press ISBN 0-691-12335-7) using data from irrationalexuberance.com/shiller_downloads/ie_data.xls