Final Report

ISyE 6740 – Spring 2021

Project Title: Find the stock winners!

**Team Member Names:** Klaus Smit, Emilio Flores Braeckow, Andres Urrutia

# 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 bagged 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.

# 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.

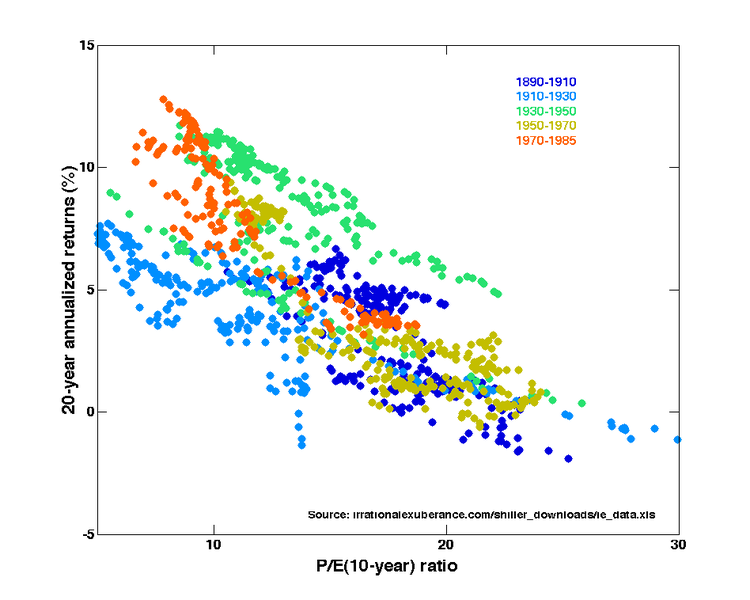
The mathematical relationship that exists between the different variables that will be explored, which are going to be built in the following experiments, are all machine learning models. “Machine learning refers to creating and using models that are learned from data.” (Grus, 2019)

As we know the answers of expected values of the data that will train these models, then the models are supervised. Additionally, these specified expected values will require the generated models to output probability predictions. Putting it all together, these models are supervised learning algorithms.

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. 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 versions of EMH:

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.

# Methodology

The flowchart below summarizes \*\*\*\*\*



These are the steps that we will follow to build our model and find the stock winners:

1. Collect data
   1. Randomly sample 200 stocks from the S&P 500 where 150 of those will be used for our training dataset and the other 50 to test and validate the models.
   2. Determine the list of features based on the most common fundamental values that are used to valuate companies.
   3. Using the Alpaca API described above, create the necessary API calls in Python to obtain the list of features determined on the previous step for each of the selected companies at timeframe 𝑋(1).
   4. Obtain the adjusted stock price of the same list of randomly sampled 200 stocks at future time 𝑋(2) with new API calls in Python.
2. Clean the Data
   1. Identify and determine the percentage of missing values for each of the features.
   2. Replace the stocks that have more than 50% of the data missing
   3. For each feature, impute the missing values using methods such as:
      1. Mean, mode, median
      2. Random values
      3. Regression using another feature of the dataset
3. Classify the Data
   1. Generate a label with options {*Winner, Loser}* for each randomly sampled stock. Where *Winner* is given to the stocks that had a higher adjusted stock price when compared at time 𝑋(2) against past time 𝑋(1), else *Loser* is provided.
4. Build and validate the models
   1. Use the cleaned dataset to build between 3 supervised classification models such as SVM, Logistic Regression and Neural Networks.
   2. Apply PCA to identify the principal components and re-train the models to see if there is an improvement in accuracy.
   3. Compare the models and select the model that best classifies the stock data into labels *{Winner, Loser}* using metrics such as F1 Score, Precision, Recall and Accuracy.
   4. Cross-validate the model to quantify the overfitting in each of the models.
   5. Perform hyperparameter tuning on the best performing model.
5. Bagging (Bootstrap Aggregation)
   1. Assemble the prediction results of different models with the goal of improving classification accuracy.

# 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 will be 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.

## **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 |

## **Missing Data**

There is expectation for the collected data to have missing entries. This could generate problems in the models and result in subpar results. Therefore, different strategies will be performed for each of the machine learning models:

1. Delete and Replace: Any stock that has 50% of its features missing will be replaced by another randomly selected stock.
2. {Mean/Mode/Median} Substitution: This strategy will replace missing values with Mean, Mode and Median, whichever yields the best classification results will be used for the final model.
3. Regression estimation: This strategy uses available data from other features to compute a regression model in order to provide a prediction of the missing feature.

## **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. Therefore the delta of each feature over a 1 year period is the set of variables from which all models will be built.

In example, 3 features for 3 stocks will be shown and the delta calculated:

Year1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

Year2: \*\*\* Waiting for Klaus to get this data \*\*\*\*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stock | bookValuePerShare | cashAndEquivalents | debtToEquityRatio | earningPerBasicShare | salesPerShare |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

These data points with a 1 year difference will then serve to generate the predictor variables, which are the deltas:

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

## **Data Preparation**

Scales of the features selected can vary highly, data transformation was then explored. Data standardization can be in the form of normalization or standardization. To choose the data preparation technique, prediction accuracy was calculated with the original data, normalized data and standardized data. Normalization was performed through Max-Min normalization. *Figure 2* shows that standardized data provided the best performing models. This may be because standardization is more robust to outliers.

*Figure 2: Prediction Accuracy Through Data Transformation*

Standardization was applied to all features in order to improve the performance of the classification models. In addition, if standardization does not take place, then there would be a violation of some of the assumptions of models to be built. As Support Vector Machine and Logistic Regression both assume that all features are centered around zero and have a variance in the same order. The following formula was used:

Where is the original feature value, is the mean of the feature vector and is the standard deviation.

## **Data Separation**

If no data separation is applied and all data is used to train the models, the resulting model will be weak against generalization to further unseen data. Therefore, training data will consist in 80% of the data and test data will be the remaining 20%. Where the partition will be made randomly.

## **Principal Component Analysis**

Principal component will also be explored per machine learning model. As the prediction power of each model and converge time could see benefits from transforming the features into the principal components that explain the most variability of the variable to predict. In addition, there is concern for the correlation between features, being that they are all fundamental information of companies. Therefore, applying PCA would help generate components that have no correlation between them.

*Figure 3,* 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 convergence time as the reduction of the dimensions of the data could improve the speed of convergence. In addition, prediction power could be benefited from the reduction in noise generated by the possibility of correlated features.

Chart, histogram

Description automatically generated

*Figure 3: Variability Explained by Each Principal Component*

# Model Evaluation

## **5.1 Logistic Regression**

Data: where binary responses

Model: Probability of success given predictors

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

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.

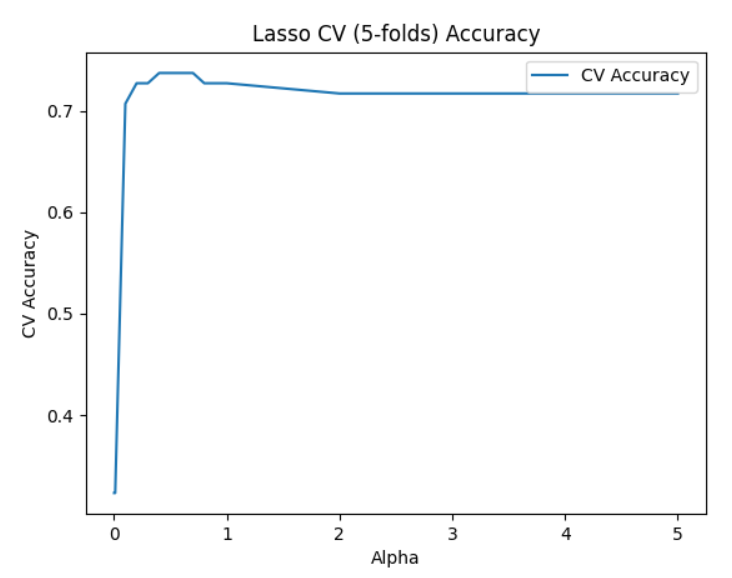
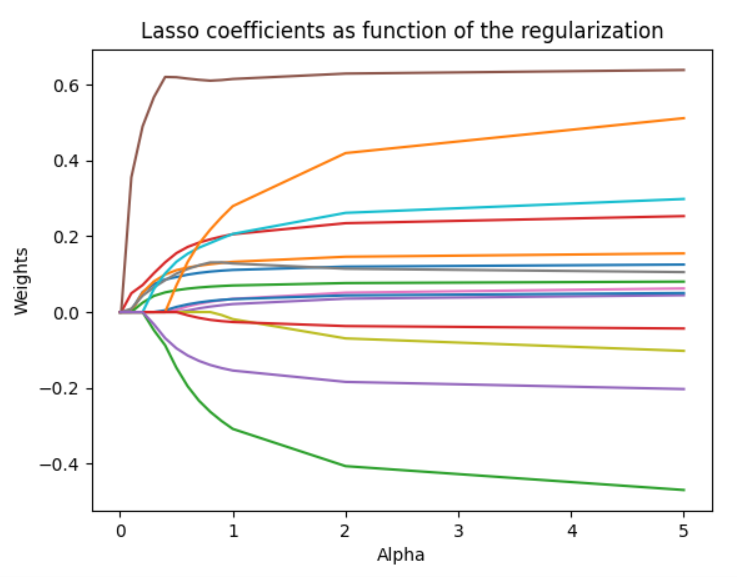
Without any hyperparameter tuning, the accuracy of the logistic model is at **66.66%.** Given the high number of features and the suspicion of correlation between them, there is potential for hyperparameter tuning to result in a more robust model with a higher prediction power.

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 |

It can be seen that Logistic Regression with both Lasso and Ridge penalization, low C values which is the inverse of the regularization strength 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%.**



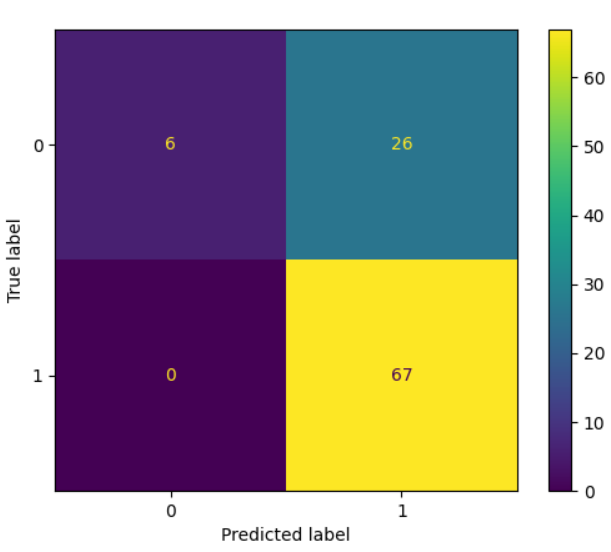
*Figure 4: Solution Path & Accuracy Through Alphas*

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 shown ahead. Where, all “Losers” are 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 5: Confusion Matrix*

## **5.2 Support Vector Machine**

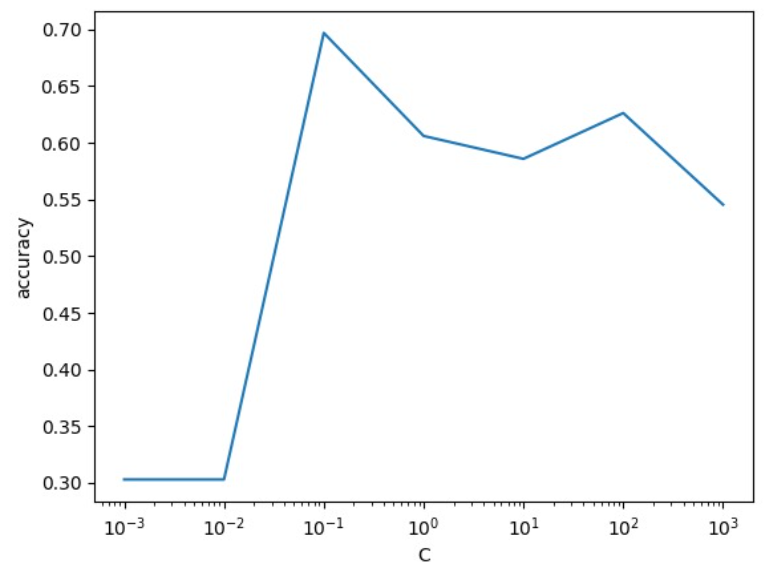
We look for hyperplanes that as far as possible while still having all the points classified closer to them. Kernel methods can be used to allow the hyperplane to not be linear. The algorithm determines support vectors, or data points at which the parallel lines are being pivoted from. The objective is to minimize the sum of the squares of the coefficients. Expressed as:

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

Hyperparameter tuning for best model performance can be performed on:

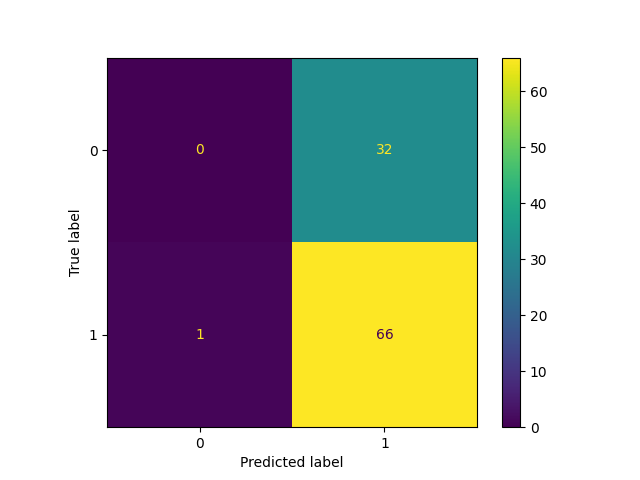
* Kernel: Kernel to be used in the algorithm in order to transform the hyperparameter form a linear to a non-linear classifier.
* Parameter (0.1 to 100): Penalty for each misclassified data point. If small, then penalty is low however a large penalty might lead to overfitting.
* Gamma (0.0001 to 10): Parameter that influences the kernel construction, for example for radial basis function it controls the distance of influence of a single training point.

Accuracy as function of C = 1/



*Figure 6: Accuracy vs C*

Confusion Matrix



*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: input layer, hidden layer and output layer. A weight ( is assigned to each of the connections between a neuron of a layer and all neurons of the past layer. Then, take all activations from previous layers and their weighted sum is calculated to calculate the activation of this node. To restrict the activation values into {0,1} values, then a sigmoid function is applied. Lastly, a bias ( is applied to make sure that a neuron is active only after a specified threshold.

These summarizes to:

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 and of the cost function with respect to any weight. Resulting in:

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

During the optimization of our model, the following **hyperparameters** where 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.

\*\*\*Charts to add:

1. Accuracy vs one of the parameters
2. Accuracy: PCA vs Non-PCA

Confusion Matrix

Chart

Description automatically generated

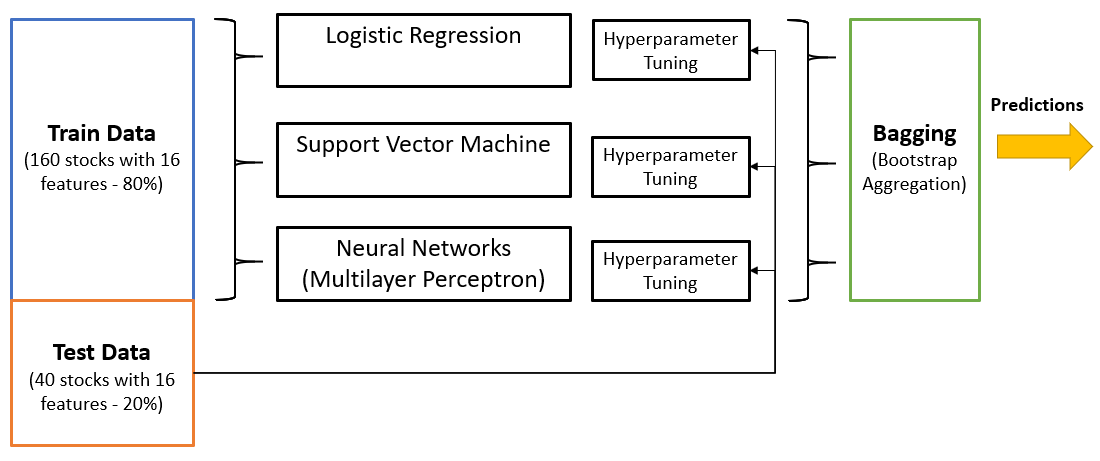
*Figure : Description of Confusion Matrix*

\*\*\*\* inbalanced dataset

## **5.4 Bagging (Bootstrap Aggregation)**

After each individual machine learning model is tuned and tested, the bagging algorithm is applied to assemble the prediction results of different models with the goal of improving classification accuracy. Bagging will allow for the weakness of each model to be minimized out by the prediction power of the other models.

The class that was more commonly predicted throughout the different models is chosen per stock. *Figure X,* shows a diagram that 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.



*Figure : Description of Bagging*

# Final Results

\*\*\*Charts to add:

1. Confusion Matrix
2. F1 Score, Recall, accuracy,
3. AUC and ROC

\*\*\* Something about Azure

Predict and evaluate our initial hypothesis under the following objectives:

1. Determine best performing classification model:
   1. Generate charts showing the comparison of accuracy metrics from the cross- validation testing results for each of the models.
   2. Generate charts showing the comparison of time to train for each classification model and compare.
2. Test initial hypothesis against best performing model:
   1. Perform hyperparameter tuning with the best performing model, with the objective of maximizing prediction accuracy.
   2. Based on output from best performing model and charts determine whether we should reject or fail to reject our null hypothesis. Where an expected classification overall accuracy is expected to be at least better than *50%* for the best performing model.

# 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    1. Convert JSON from API call responses to a Pandas Dataframe object    2. 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    1. Research Alpaca.Markets API documentation    2. 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 |

# References

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

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

# 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