Machine Learning Engineer Nanodegree

Capstone Project: The Efficacy of Multilayer Perceptron Algorithms in Predicting Bankruptcy

Brian Campbell
November 24, 2019

I. Definition

Project Overview

PROBLEM DOMAIN AND PROJECT ORIGIN

Many economists have claimed that one cannot predict individual winners and losers in the stock market and cite the Efficient Market Hypothesis to justify this claim (Investopedia). These economists believe that because one cannot pick individual stocks, investors should simply spread their risk amongst many stocks, i.e. investing in Exchange Traded Funds (ETF). While these economists are correct in identifying that traditional statistical methods have failed to identify individual winning and losing stocks, advancements in machine learning could make it possible to identify the winners and losers. Furthermore, there are a couple of benefits to investing in individual stocks than spreading risk in ETFs. First, investors can realize larger returns. Second, identifying financially sound businesses could encourage investors to take on more risks for social causes. For example, an investor might be willing to put more capital in a company that is developing the cure to Pancreatic cancer if a machine learning algorithm could reliably predict whether the pharmaceutical company was financially sound.

There is some research to suggest that machine learning can identify which companies are financially sound (Mattsson, Steinert and Dzemski, 2017). Bjorn

Mattsson tested gradient boosting (GB), random forest (RF), and multilayer perceptron (MLP) machine learning algorithms on a dataset of Polish companies from 2007 to 2012 (Mattsson, Steinert and Dzemski, 2017). 64 different financial features were used to predict the binary target variable, bankruptcy (Mattsson, Steinert and Dzemski, 2017). Out of the three algorithms, the GB algorithm was best able to predict whether a company declared bankruptcy (Mattsson, Steinert and Dzemski, 2017).

DATASETS AND INPUTS

The dataset consists of financial indicators from polish manufacturing companies from a five-year period, 2007-2012 (Mattsson, Steinert and Dzemski, 2017; Tomczak, 2016). 64 financial indicators, such as net profit / total assets, total liabilities / total assets, working capital / total assets, retained earnings / total assets, EBIT / total assets, sales / total assets, total assets / total liability, gross profit / total assets, gross profit / sales, etc, are recorded for each company in the dataset (See Appendix A). These financial indicators are the traditional key performance indicators (KPIs) used by financial analysts to assess the health of companies. The target variable is a binary bankruptcy variable, 1 = bankrupt and 0 = not bankrupt (Mattsson, Steinert and Dzemski, 2017; Tomczak, 2016). The dataset is further broken down by the year in which the financial indicators for a company were recorded, consisting of five subsets total (Mattsson, Steinert and Dzemski, 2017).

Furthermore, the dataset is highly unbalanced. The number of companies that did not go bankrupt far exceed the number that did (Mattsson, Steinert and Dzemski, 2017). To correct this, Mattsson et al used the oversampling technique, meaning they provided the classifier with an equal number of non-bankrupt and bankrupt companies to the classifier (Mattsson, Steinert and Dzemski, 2017). This analysis will use the oversampling technique.

Problem Statement

PROBLEM STATEMENT

The original problem to be solved was whether the MLP algorithm can be modified to perform better. Mattsson et al found that the RF and GB algorithms consistently outperformed the MLP algorithm (Mattsson, Steinert and Dzemski, 2017). While Mattsson et al modified some of their hyperparameters, their MLP algorithm included weight decay in the activation function and they applied dropout to their hidden layer nodes, it is unclear whether they modified the hyperparameters beyond these measures (Mattsson, Steinert and Dzemski, 2017). The original results of Mattson et al's MLP model would serve as a benchmark, and this analysis would seek to improve the performance of the MLP by modifying parameters, e.g. the number of hidden layers and nodes, and the hyperparameters, e.g. the activation functions and the loss optimizers.

However, upon further inspection of the dataset, Mattsson et al pre-processed the data incorrectly. Mattson et al broke down the dataset into five distinct datasets, a dataset for each of the five years (Mattsson, Steinert and Dzemski, 2017; Tomczak, 2016). They then created three different versions of the data for each year: the first had no null values in the data, while the second and third had null values (Mattsson, Steinert and Dzemski, 2017). Furthermore, version one had no feature variables regarding null values, version two had dummy variables created for each x feature, signaling whether a given value in the x feature was null, and version three had a null counter variable that counted the number of null values in a given row of data (Mattsson, Steinert and Dzemski, 2017). They then split the data for each year and each version of the data (see the Area Under the Curve, AUC, scores in Appendix B). The error that Mattson et al overlooked was that the second and fourth year datasets contained rows that contained at least one null value (see Appendices C and D). They probably imputed values for the nulls in the second and fourth years, which one cannot do in a dataset that has at least one null value in each row. If every row has at least one null value, then it is impossible to create a testing set without imputing values into it, which would introduce data leakage, model information outside the scope of the training set, into the testing set (Machine Learning Mastery, 2016; StackExchange, 2017). Data leakages lead to overly optimistic models, and avoiding obvious sources of data leakages, such

as imputing values into the testing data, is common practice in building machine learning models (Machine Learning Mastery, 2016; StackExchange, 2017). The violation of this practice necessitates that a different methodology from Mattson et al's be used.

The new problem this analysis addresses is given that Mattson et al's models are overly optimistic due to data leakage, how well should an MLP model, not corrupted by imputed values in the testing data, perform in predicting bankruptcy? It is highly likely that MLPs that do not suffer from data leakage in their train-test split should perform worse than Mattsson et al's models (Machine Learning Mastery, 2016; Mattsson, Steinert and Dzemski, 2017; StackExchange, 2017). Also, along with a new methodology that protects the model from data leakage, this analysis will also need a new benchmark for performance.

THE SOLUTION

The original solution for the original problem statement would have been to test several MLP algorithms with varying parameters and hyperparameters to determine whether modifications to the MLP algorithm can improve performance when compared to Mattson et al's benchmark. However, the new solution is to create a new methodology and a new benchmark in order to determine how well MLP models, unaffected by data leakage, can predict whether a company goes bankrupt. The new methodology would do away with breaking down the datasets by year, and simply use the final year's data as the testing data. The Udacity mentor who suggested this concept said:

> Typically, with time series data like this, we just use the first three years as the training data and the last as the testing data. Think about 'future' data here, as we couldn't do this for year 5 (future data). We need the testing set to be as similar as possible to 'real world' data as possible (Udacity Mentor Review # 1, 2019).

This analysis uses the concept described by the Udacity mentor, but it does not use data from year four as testing data, because year four does not have any rows of data that do not contain a null value (see Appendix C). However, this methodology is still feasible, because year five has 2,995 rows of data with no null values (see Appendix E) and these non-null rows will serve as the testing data. Furthermore, there will be four different versions of the dataset instead of three: the first has no null values (labeled as "No Nulls" in bankruptcy-model.ipynb), the second ("Nulls Only") has imputed null values, the third ("One Hot") has imputed null values and has a dummy variable for each X feature that signifies whether a value is null for that column, and the final dataset ("Sum") imputes null values and has a single variable that counts the number of null values in a given row (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). Furthermore, the benchmark models will be four logistic regression models, one for each of the datasets, as suggested by the Udacity mentor in the first review of the analysis (Udacity Mentor Review # 1, 2019). Finally, four MLP models will be implemented and compared to the benchmark.

Metrics

METRICS

The evaluation metric used is an AUC score (see Appendix F). The AUC is the area under the two-dimensional graph, the Receiver Operating Characteristic (ROC) curve (see Appendix G), with the true positive rate as the y-axis and the false positive rate as the x-axis (Google Developers, 2019). A model that has predictions that are 100% correct has an AUC of 1, while a model that has predictions that are 100% wrong have an AUC of 0 (Google Developers, 2019). AUC scores are used for two reasons. First, AUC is a good metric for a binary classification (bankrupt versus not bankrupt). Second, it is the metric used in Mattsson et al's models, and while their models are overly optimistic, there is nothing wrong with the metric (Mattsson, Steinert and Dzemski, 2017). Furthermore, in revising the scores for Mattson et al's models, the same score should be used from their experiment.

II. Analysis

Data Exploration

DATA OVERVIEW

The dataset consists of financial indicators from polish manufacturing companies from a five-year period, 2007-2012 (Mattsson, Steinert and Dzemski, 2017; Tomczak, 2016). 64 financial indicators, such as net profit / total assets, total liabilities / total assets, working capital / total assets, retained earnings / total assets, EBIT / total assets, sales / total assets, total assets / total liability, gross profit / total assets, gross profit / sales, etc, are recorded for each company in the dataset (See Appendix A). These financial indicators are the traditional key performance indicators (KPIs) used by financial analysts to assess the health of companies. The target variable is a binary bankruptcy variable, 1 = bankrupt and 0 = not bankrupt (Mattsson, Steinert and Dzemski, 2017; Tomczak, 2016). The dataset is further broken down by the year in which the financial indicators for a company were recorded, consisting of five subsets total (Mattsson, Steinert and Dzemski, 2017).

DESCRIPTIVE STATISTICS

Descriptive statistics were found for both the dataset with no null values and the two datasets with null values. The dataset with no nulls has 65 variables including the target variable, “bankrupt” (Campbell, 2019). The dataset has 12,964 rows of data, and the testing set (2,995 rows) as a percentage of the entire dataset is about 23.10%, which is a typical split for machine learning models. (Campbell, 2019). There is a wide variance in the standard deviations for each X feature variable. The minimum standard deviation for the X feature variables is about 0.19, the maximum is about 62,864.40, and the mean is about 2,868.88. The numbers show that these feature variables have different variances, which suggests that feature scaling will need to be used, considering that neural networks use some form gradient descent (Boris, 2019). Algorithms that use gradient descent in their loss function tend to weight features with

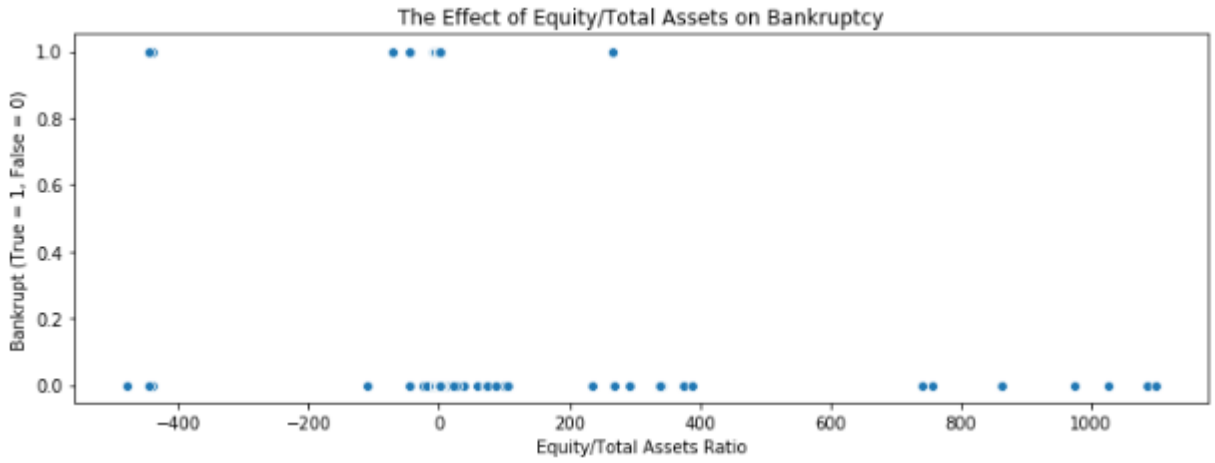
greater ranges more than those with smaller ranges, and feature scaling is used to prevent this bias in Mattsson et al's MLP models though standardization (Boris, 2019; Mattsson, Steinert and Dzemski, 2017). Furthermore, the number of bankrupt companies in the no nulls dataset, 232, is only 1.79% of the entire dataset. Essentially, a lazy algorithm could predict that a company will not go bankrupt, and it would be right with greater than 98% accuracy. To correct the imbalanced dataset, oversampling, or randomly resampling the bankrupt cases, will prevent an algorithm from simply choosing the majority case, not bankrupt, and is a technique used by Mattsson et al (Mattsson, Steinert and Dzemski, 2017; Udacity Mentor Review #1, 2019).

The "One Hot", "Sum" and "Nulls Only" datasets have 129, 66, and 65 variables, including the target variable, respectively (Campbell, 2019). The datasets have 46,023 rows of data, and the testing sets (2,995 rows) as a percentage of the entire dataset is about 6.51%, which is small for testing sets. (Campbell, 2019). Although the testing sets are small when compared to the entire datasets, they have enough rows, 2,995; it is acceptable to have a small testing set in relation to the total dataset as long as long as the testing set has a few thousand rows (Cross Validated, 2018a). Also, there is a wide variance in the standard deviations for each X feature variable in the three datasets with imputed null values. The minimum standard deviation for the X feature variables is about 0.82, the maximum is about 142,955.25, and the mean is about 13,494.89. The numbers show that these feature variables have different variances, which suggests that feature scaling will need to be used (Boris, 2019; Mattsson, Steinert and Dzemski, 2017). Furthermore, the number of bankrupt companies in the datasets, 2,184, is only 4.75% of the entire dataset. Oversampling will need to be used to correct the imbalanced datasets and prevent the MLP models from simply predicting the majority case, not bankrupt (Mattsson, Steinert and Dzemski, 2017; Udacity Mentor Review #1, 2019).

Exploratory Visualization

VISUALIZATION

Two x features variables, net profit / total assets (X1) and equity / total assets (X10) were examined to see whether correlations existed with bankruptcy (Mattsson, Steinert and Dzemski, 2017). X1 was selected for examination because the financial feature is a measure of how efficient companies are with their assets, and more efficient companies generally should not go bankrupt. Also, X10 was examined because equity/asset ratios are an indicator of how leveraged a company is, and how much a company owes its creditors should play a role in bankruptcy. These features were examined on two datasets, one without null values and one with imputed null values. Only on the imputed dataset did the X10 variable, equity / total assets, reveal a possible relationship (see the figure below). Although not a perfect correlation, companies with a high equity / total assets ratio, or companies that were not as highly levered, tended not go bankrupt (Campbell, 2019). However, although the bankrupt companies had low equity/total asset ratios, there were many companies with low ratios that did not go bankrupt (the company with the lowest recorded equity/total asset ratio did not go bankrupt). The visualization only suggests a weak correlation exists between the ratio and bankruptcy.



(Campbell, 2019)

Algorithms and Techniques

THE ALGORITHMS IN THE ANALYSIS

Two algorithms are used in the analysis: logistic regression models as a benchmark, and the MLPs. Logistic Regression is the simplest of all binary classifier models, which makes it a good benchmark model (Ameisen, 2018: Udacity Mentor Review # 1, 2019). MLPs are based upon Frank Rosneblatt's perceptron with a binary output, with the key difference being that MLPs have hidden layers (Nicholson, 2019). The number of epochs for the MLPs were 100 with a batch size of 50, which were the parameters in Mattsson et al's MLPs (Mattsson, Steinert and Dzemski, 2017). Mattsson's original hidden layers and output nodes per layer were not used, because they did not provide these parameters (Mattsson, Steinert and Dzemski, 2017). This analysis uses two hidden layers for the MLPs. For the MLP models that used the "No Nulls", "Nulls Only", and the "Sum" datasets, the input layer produced 32 output nodes, the second layer produced eight output nodes, the third layer produced eight output nodes, and the final layer produced one output node (Campbell 2019). For the MLP model that used the "One Hot" dataset, the input layer produced 64 output nodes, the second layer produced 32 output nodes, the third layer produced 16 output nodes, and the final layer produced one output node (Campbell 2019). The output nodes of the first three layers were increased due to the fact that the input layer processed 128 input variables while the other MLPs only processed 65 input variables at most.

THE ALGORITHMS IN MORE DEPTH

Logistic regression algorithms are used to predict binary outcomes, IE 1/0 or yes/no (Chandrayan, 2017). To do this, the algorithm first regresses a boundary line using the independent variables to best split the binary variable into 1s and 0s (Chandrayan, 2017). The boundary line is drawn using Maximum Likelihood Estimation, an iterative process of adjusting the coefficients of each independent variable to best split the outcome variable until no further improvements can be made in the split (Chandrayan, 2017). Once, the line is drawn, the algorithm can then classify the testing data using the sigmoid function, an activation function (see below):

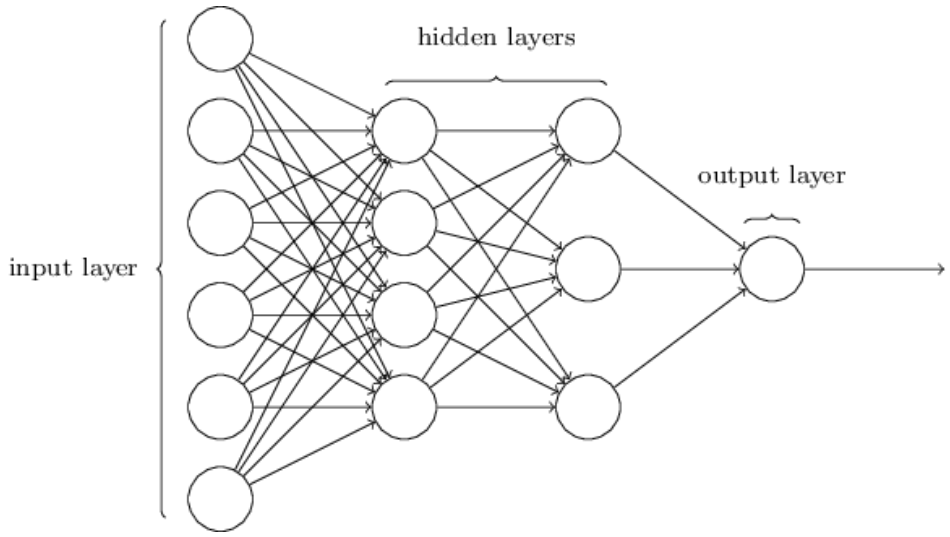
$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

(Narkhede, 2018)

If the sigmoid function returns a value greater than 0.5, then the algorithm predicts a 1 classification, and if a value of less than 0.5 is returned, the algorithm predicts a classification of 0 (Narkhede, 2018). The possible values of the sigmoid function are between 0-1, exclusive (Narkhede, 2018). Essentially, the sigmoid function is predicting the probabilities of a 1 classification. In a machine learning model, the boundary line is drawn on the training data, and the activation function, the sigmoid function, is used to determine the probability that a given row in the testing data is classified as 1 or 0.

Like logistic regression, MLPs also uses its independent variables to split its binary dependent variable (Kain, 2018). The algorithm processes data in layers (Kain, 2018). The input layer randomly initializes the weights (like coefficients in logistic regression, the weights are factors that are multiplied with their respective independent variable) for the independent variables, which are then summed and sent to the neurons in the hidden layer (Kain, 2018). Within a neuron, an activation function (the activation function does not necessarily need to be the sigmoid function) is used to calculate the probability of a binary classification of 1 (Kain, 2018). The output is then assigned a random weight factor to multiply the output, and then it is summed with other outputs with weights and sent to another neuron in the next layer in which an activation function is used within the neuron to determine the probability of a classification of 1 (Kain, 2018). The number of hidden layers in an MLP is arbitrary, but an MLP must have at least one hidden layer (Kain, 2018). The outputs of the last hidden layer (which could also be the first) are given random weight factors and sent to a single neuron in the output layer, which uses the sigmoid activation function to calculate the probability of a classification of 1 (Kain, 2018). The algorithm then makes predictions based upon the sigmoid function’s values, and the difference between the predicted classifications and the actual classifications are used to calculate the loss (Kain, 2018). Once the loss is calculated, it is fed backward through the network in a process called Backpropagation (Kain, 2018). During Backpropagation, a loss optimizer, usually stochastic gradient descent, is used to determine the weights that are associated with the least amount of loss, which are then saved for the next iteration of data to be sent through (Kain, 2018).

This process is repeated many times until the optimal weights are found that are associated with the lowest loss. See below for a visual representation of an MLP.



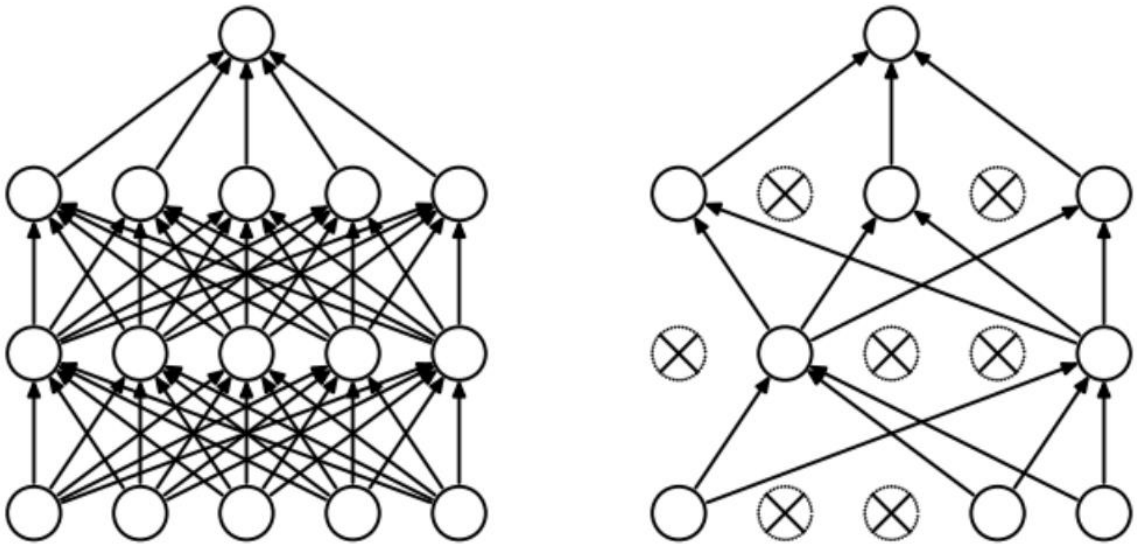
(Karim, 2016)

ALGORITHM TECHNIQUES

While the logistic regression models did not use many techniques or hyperparameters, the MLP models did. The logistic regression models used L2 regularization, which is an accepted convention for punishing weights in a model to prevent overfitting (McCaffrey, 2019). Overfitting is something that occurs when model weights become too big and overly specified for the training data, resulting in poor accuracy scores when the testing data is applied to the model, and L2 regularization seeks to decrease weights and increase test accuracy. Also, the MLP models use L2 regularization along with Dropout (Campbell, 2019). Mattson et al also used Dropout, a method of randomly shutting off nodes in the layers of an MLP to discourage the weights associated with those nodes from becoming too large and creating overfitting, in their models, and this analysis used it as well (Brownlee, 2019; Mattsson, Steinert and Dzemski, 2017).

THE TECHNIQUES IN MORE DEPTH

Dropout is a technique to prevent overfitting in MLPs (Budhiraja, 2016). Within an MLP, left unchecked, neurons develop codependencies with other neurons (Budhiraja, 2016). These codependencies cause the weights of some neurons to strengthen while others weaken (Budhiraja, 2016). The problem with neurons with too strong of weights is it leads to overfitting of the training data, resulting in poor accuracy scores on the testing data (Budhiraja, 2016). Dropout is a method of randomly shutting off neurons within a layer with an arbitrary probability (Budhiraja, 2016). By randomly shutting off neurons, neurons that would have weaker weights are given the chance to strengthen their weights, and neurons that would have excessively strong weights are not given the chance to overdevelop the weights (Budhiraja, 2016). Generally, MLPs with more moderate weights fit the training data better and realize better accuracy results on the testing set. See below for a visualization of Dropout (Budhiraja, 2016).



(a) Standard Neural Net (b) After applying dropout.

(Budhiraja, 2016)

While Dropout is exclusive to deep learning algorithms, L2 regularization can be used in many algorithms, including logistic regression and MLP (Karim, 2018). L2 regularization is used to penalize weights in order to avoid overfitting in an algorithm (Karim, 2018). L2 regularization takes the result of a loss function and adds the sum of the squared weights multiplied by an arbitrary factor, lambda (Karim, 2018). Loss

functions with L2 regularization with lambdas close to zero punish the weights very little, while loss functions with L2 regularization with high lambdas severely punish the weights (Karim, 2018). Too low of lambdas will not prevent overfitting, resulting in low testing set accuracy scores, and too high of lambdas will cause underfitting, which also results in low testing set accuracy scores (Karim, 2018). Typically, lambda is set somewhere between 0 and 0.1 (Brownlee, 2018). See below for a visualization of the L2 equation.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

(Karim, 2018)

THE DATA IN THE MODELS

There were four logistic regression benchmark models and four MLP models for each of the four datasets, “No Nulls,” “Nulls Only,” “One Hot,” and “Sum.” Neither the logistic regression nor the MLP models used a random train-test split; they used the fifth year’s data as the testing data (Udacity Mentor Review # 1, 2019). Furthermore, the logistic regression models, due to the fact that the datasets did not use a random train-test split, could not use the K-folds Cross Validation method, the standard for logistic regression models, for testing the AUC scores of the training data (Cochrane, 2018). Instead, Holdout Cross Validation was used to assess the AUC scores of the data (Campbell, 2019; Cochrane, 2018). Essentially, the training data was broken into two used to cross validate the fitted model and return the training AUC score (Campbell, 2019; Cochrane, 2018). Then the testing set was applied to the fitted models to determine the overall AUC score (Campbell, 2019; Cochrane, 2018). The same method of Holdout Cross Validation was also used on the MLPs, but the Holdout method is more commonly used in deep learning models than logistic regression models (Brownlee, 2019; Campbell, 2019; Cochrane, 2018).

BENCHMARK

There were no existing reliable benchmarks, Mattsson et al's models suffered from data leakage, and the four logistic regression models needed to be used to serve as the benchmark. First, logistic regression models make good benchmark models because they are the simplest forms of binary classifiers (Ameisen, 2018; Udacity Mentor Review # 1, 2019). Furthermore, apart from the fact that the logistic regression models did not suffer from the same data leakage problem as Mattsson et Al's models, the logistic regression model that used the "Nulls Only" dataset essentially served as a benchmark measure of how much AUC scores improved by simply imputing null values (Campbell, 2019). Mattsson et al never measured the effect of imputation on AUC scores (see Appendix B). The benchmark logistic regression AUC scores were as follows: 0.3675 for the "No Nulls" dataset, 0.6903 for the "Nulls Only" dataset, 0.6872 for the "One Hot" dataset, and 0.6822 for the "Sum" dataset (Campbell 2019). The datasets containing null values outperformed the "No Nulls" dataset. Furthermore, simply imputing null values increased AUC scores by 0.3228. The best model's, the "No Nulls" logistic benchmark, predictions were correct 69.03% of the time (Campbell, 2019). Essentially, the logistic regression models were good benchmarks because they were simple, they did not suffer from data leakage, and they also measured the effect of data imputation (Ameisen, 2018; Campbell, 2019; Mattsson, Steinert and Dzemski, 2017; Udacity Mentor Review # 1, 2019).

III. Methodology

Data Preprocessing

DATA IMPUTATION, STANDARDIZATION, AND OVERSAMPLING

In the initial exploration of the data, it was discovered that the majority of rows in the datasets had at least one null value, the feature variables had varying ranges, and the datasets were highly imbalanced (Campbell, 2019). To remedy these abnormalities, data imputation, standardization, and oversampling was applied to the datasets. First,

imputation was applied to the training sets of the “Nulls Only”, “One Hot”, and “Sum”, datasets. However, imputation was limited to the training sets in order to prevent data leakage into the testing sets (Cross Validated, 2018a). The “IterativeImputer” class from the Sklearn library was used to find the null values, which used regressions drawn from the other feature variables to predict the null values (Scikit-learn.org, 2019a). While it is not typical to impute null values into datasets, it made sense to do it in this case, because there were so many rows with null values and there could have been companies within the dataset that attempt to hide distress by simply not reporting important financial KPIs. Next, in all the datasets, the features were standardized, a common form of feature scaling that sets the mean of the feature variables to zero with a standard deviation of 1 so that the gradient descent loss optimizer in the logistic regression and MLP models would not be affected by a given feature variable with a wide range of data (Boris, 2019). The “StandardScaler” class from the Sklearn library was used to standardize the feature variables (Scikit-learn.org, 2019b). Finally, in all datasets, oversampling was used to prevent the algorithms from lazily predicting the majority class, not bankrupt (Mattsson, Steinert and Dzemski, 2017; Udacity Mentor Review #1, 2019). The “SMOTE” class from the Imblearn library was used to oversample the datasets (Becker, 2016).

Implementation

THE COMPLICATION IN IMPLEMENTING

Implementation of the algorithms went smoothly, with only one minor complication. While the train-test split had an arbitrary testing set, the fifth year of data, which prohibited the use of standard K-fold Cross Validation, this split and the use of Holdout Cross Validation did nothing to impede the processing of the data in the algorithms (Cochrane, 2018: Campbell, 2019). Both the logistic regression models and the MLPs processed data from the four datasets: “No Nulls”, “Nulls Only”, “One Hot”, and “Sum”. The purpose of the four datasets was to identify what effects adding null values had on the algorithms and what effects did adding descriptive variables concerning these null values, dummy variables for each input variable column that indicated whether a given

value was null (the “One Hot” dataset) and a null value counter that counted the number of null values for a given row (the “Sum” dataset), had on the algorithms’ performances, their AUC scores. (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). In running these datasets on the algorithms, there was only one complication, and that was for the MLPs. After each epoch, the model with the best weights and lowest validation loss needed to be recorded. This was only a complication in the sense that researching how to do this correctly was time intensive. The solution to recording the best weights was found in the Keras library using the “ModelCheckpoint” class (Campbell, 2019; Keras, 2019b). Appendix I shows the importation of “ModelCheckpoint” from Keras along with the “build_model” function that was used to build the MLPs. Other than that, there were no complications, including adding techniques to the MLPs, such as Dropout or L2 regularization (Brownlee, 2019; Campbell, 2019; McCaffrey, 2019; Mattsson, Steinert and Dzemski, 2017).

Refinement

METHODS USED TO IMPROVE PERFORMANCE IN THE MLP MODELS

Several methods of improvements were tested for the MLPs, but only two, Early Stopping and decreasing the learning rate in the stochastic gradient descent loss optimizer helped (Campbell, 2019). L2 regularization and dropout were not experimented on all that much due to the fact that they are well documented methods to prevent overfitting, most deep learning models include some form of regularization or dropout and it only made sense to incorporate them in the analysis (generally both). All the MLPs had Dropout with 50% probability and L2 regularization. Traditionally, the only loss optimizer that deep learning models used was stochastic gradient descent, but more modern deep learning models are experimenting with the new generation of loss optimizers, with varying results (Brownlee, 2016). This analysis experimented with RMSprop and Nadam loss optimizers, but found these optimizers performed worse than stochastic gradient descent (Keras, 2019c). One technique that slightly improved AUC scores was Early Stopping, or limiting the number of epochs when running an MLP helped AUC scores, suggesting that despite having methods to counter overfitting, IE

Dropout and L2 regularization, the models were especially sensitive to overfitting (Campbell, 2019; Cross Validated, 2018a). This analysis found that by slightly dropping the number of epochs from 100, the number of epochs Mattsson et al used in their analysis, to 75, all the MLP models' AUC scores improved slightly (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). However, these improvements were minor, within .05 AUC score point improvements and not recorded. However, the improvements in the AUC scores through adjusting the learning rates in the loss optimizer, stochastic gradient descent, were recorded (Campbell, 2019).

Adjusting learning rates in loss optimizers is one of the most effective methods in improving model performance in deep learning models (Zulkifli, 2018). This analysis used a general rule of thumb in optimizing loss in deep learning models, lowering the learning rate in the loss optimizer generally results in better loss optimization and higher accuracy scores (Campbell, 2019; Zulkifli, 2018). This analysis compared the performance of the different MLPs when using the default learning rate (0.01) against that of the low learning rate (0.001). All MLP models using the low learning rate improved their AUC scores by at least 0.0890 points (Campbell, 2019). The MLPs trained on the "No Nulls" dataset improved their AUC scores by 0.0890, from .5000 to .5890, by using the low learning rate (Campbell, 2019). The MLPs trained on the "Nulls Only" dataset improved by .3552 AUC score points, rom .3151 to .6703, by using the low learning rate (Campbell, 2019). The MLPs trained on the "One Hot" dataset improved by .1411 AUC score points, from .4916 to .6327, by using the low learning rate (Campbell, 2019). The MLPs trained on the "Sum" dataset improved by .1387 AUC score points, from .4916 to .6327, by using the low learning rate (Campbell, 2019). These results were significant; the models with the least improvement, the "No Nulls" model, improved its ability to correctly predict bankruptcy by 17.80% while the "Nulls Only" model, the model with the greatest improvement, improved its ability to predict bankruptcy by 112.73% (Campbell, 2019). However, despite the drastic improvements in the MLP models' AUC scores through lowering the learning rate, all the MLPs, except for the "No Nulls" model, failed to beat the AUC scores of their logistic regression model benchmarks (Campbell, 2019).

IV. Results

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

MLP AUC SCORES VERSUS BENCHMARK SCORES

All the final MLP models, except for the "No Nulls" model, failed to surpass their respective logistic regression benchmarks in their AUC scores (Campbell, 2019). The "Nulls Only" MLP model fell short of its logistic benchmark by .02 AUC score points, .6903 versus .6703 AUC score points (Campbell, 2019). The "One Hot" MLP model fell short of its logistic benchmark by .0545 AUC score points, .6872 versus .6327 AUC score points (Campbell, 2019). The "Sum" MLP model fell short of its logistic benchmark by .0367 AUC score points, .6822 versus .6455 AUC score points (Campbell, 2019). However, the "No Nulls" MLP model outperformed it logistic regression benchmark by 0.2215 AUC score points, .5890 vs .3675 AUC score points (Campbell, 2019). The best MLP model and the only one to beat its benchmark, the "No Nulls" MLP, is correct 58.90% in its bankruptcy predictions (Campbell, 2019; Google Developers, 2019). Furthermore, this MLP model improved its ability to correctly make bankruptcy predictions by 60.27% (Campbell, 2019).

ANALYZING THE RESULTS

The only expectation for the MLP models was that they should perform worse than Mattson et al's models (Machine Learning Mastery, 2016; Mattsson, Steinert and Dzemski, 2017; StackExchange, 2017). Mattson et al's MLP models made correct

bankruptcy classifications anywhere between 72-92% of the time, while these MLP models correctly predicted bankruptcy classification between 58.90-67.03% of the time (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). The MLPs in this analysis performed much worse than the MLPs in Mattsson et al's analysis; the best MLP in this analysis performed worse than Mattson et al's worst MLP (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). However, the variance in AUC score points between the MLPs in this analysis is lower than that of Mattson et al's MLPs, .0813 vs .2000 (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017).

There are several take-aways from the findings of the logistic regression and MLP models within this analysis. First, the MLP models do not appear to be particularly effective in predicting bankruptcy when they have imputed null values. Second, both the logistic regression and MLP models performed worse when descriptive variables describing missing values (the "One Hot" and "Sum" datasets), were run through the models when compared to the logistic regression and MLP models that only had imputed values (the "Nulls Only" dataset), which is contrary to Mattson et al's findings (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). Mattson et al had found that descriptive variables either indicating or counting null values helped improve AUC scores (Mattsson, Steinert and Dzemski, 2017). The MLPs on the "One Hot" and "Sum" datasets had lower AUC scores, .6327 and .6455 respectively, than the MLP on the "Nulls Only" data had, .6703 (Campbell, 2019). The logistic regression models on the "One Hot" and "Sum" datasets had lower AUC scores, .6872 and .6822 respectively, than the logistic regression model on the "Nulls Only" dataset had, .6903 (Campbell, 2019). Finally, the only MLP model that was better than the benchmark was the MLP using the "No Nulls" data, and that was not a very good model (Campbell, 2019). This model was correct 58.90% of the time in its bankruptcy classifications, which only beats random chance, 50%, by 8.90 percentage points (Campbell 2019).

However, one last question, the most important question, remains. Can one trust the results? These results are trustworthy considering that the MLPs performed worse than Mattsson et al's MLPs once the train-test split was modified to protect from data leakage (Machine Learning Mastery, 2016; Mattsson, Steinert and Dzemski, 2017;

StackExchange, 2017). However, one should not trust any of the models using imputed data. First, the models that had descriptive variables regarding the null values performed worse than the models built upon imputed data only (Campbell, 2019). If the models using variable that describe the null data had improved AUC scores, this would have suggested that the presence of null data could help predict bankruptcy, IE it is possible businesses were not reporting data to hide distress. However, these models were not able to verify that the presence of null data did anything to improve bankruptcy prediction. Therefore, when one observes the .6703 and the .6903 AUC scores of the logistic regression and MLP models built upon the “Nulls Only” dataset, a significant portion of the model is built upon assumed data as opposed to recorded data, meaning that a significant portion of their AUC scores is built upon assumptions rather than reported data (Campbell, 2019). Finally, the MLP model built upon the “No Nulls” can be trusted; it had a .5890 AUC score (Campbell, 2019). This model is built only on recorded data, and it does not rely on the assumptions that the models with imputed values rely on (Campbell, 2019). Furthermore, the model aligns with conventional economic thinking; it is difficult to predict the future of companies, and this model only beats random chance by 8.90 percentage points in predicting bankruptcy classification (Investopedia). Essentially, believing in this MLP is not far off from believing in conventional economic thought or random chance.

Justification

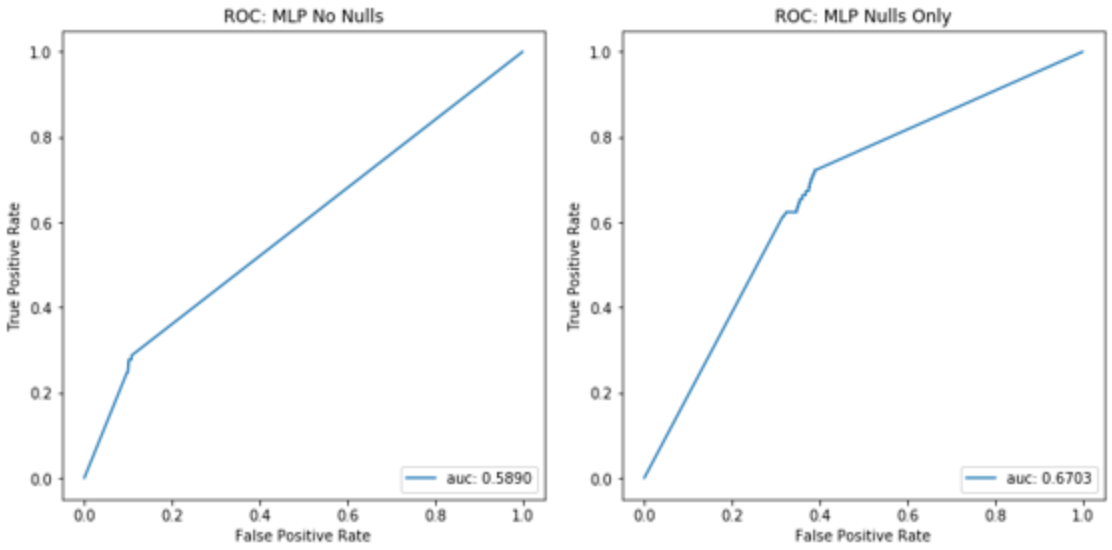
Overall, the MLP models performed worse than the logistic regression benchmark models. The “Nulls Only” MLP model fell short of its logistic benchmark by .02 AUC score points, .6903 versus .6703 AUC score points (Campbell, 2019). The “One Hot” MLP model fell short of its logistic benchmark by .0545 AUC score points, .6872 versus .6327 AUC score points (Campbell, 2019). The “Sum” MLP model fell short of its logistic benchmark by .0367 AUC score points, .6822 versus .6455 AUC score points (Campbell, 2019). However, the “No Nulls” MLP model outperformed it logistic regression benchmark by 0.2215 AUC score points, .5890 vs .3675 AUC score points (Campbell, 2019). The best MLP model and the only one to beat its benchmark, the “No

Nulls” MLP, is correct 58.90% in its bankruptcy predictions (Campbell, 2019; Google Developers, 2019). Furthermore, this MLP model improved its ability to correctly make bankruptcy predictions by 60.27% (Campbell, 2019).

These results are in line with expectations. Due to the Mattsson et al’s train-test split that introduced data leakage, his MLPs’ results were overly optimistic, and the MLPs’ results in this analysis, based upon a train-test split that eliminates data leakage, realized much lower AUC scores (Campbell, 2019; Machine Learning Mastery, 2016; Mattsson, Steinert and Dzemski, 2017; StackExchange, 2017). An additional finding that ran counter to Mattson et al’s findings was the dummy variables in the “One Hot” dataset and the “missing_counter” in the “Sum” dataset worsened AUC scores for both the MLPs and the logistic regression models (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). The results of this analysis should be preferred over Mattsson et al’s results, because the fitted model does not have access to future data (Campbell, 2019; Machine Learning Mastery, 2016; Mattsson, Steinert and Dzemski, 2017; StackExchange, 2017).

V. Conclusion

Free-Form Visualization



(Campbell, 2019)

While the ROC curves show that the MLP model built upon the “Nulls Only” data with a larger AUC score, 0.6703, this model should be rejected for the MLP model built upon the “No Nulls” data (Campbell, 2019). First, considering the MLPs with variables describing the null values, the models using the “One Hot” and “Sum” datasets, have lower AUC scores than the MLP “Nulls Only” model, it is highly likely that the presence of null values have no impact on a model’s AUC scores (Campbell, 2019). Therefore, it is also highly likely that any gains in the AUC score in the MLP “Nulls Only” model solely comes from assumed values. Therefore, this model should be rejected for the MLP “No Nulls” model, because this model is based upon less assumptions. Despite the MLP “No Nulls” lower AUC score of 0.5890, one can better trust the model’s predictions because these predictions are based upon recorded data rather than assumed data (Campbell, 2019). Finally, it should be preferred over the MLP “Nulls Only” model, because the MLP “No Nulls” model beat its logistic regression benchmark AUC score of 0.3675, while the MLP “Nulls Only” model failed to match or surpass its logistic regression benchmark AUC score of .6903 (Campbell, 2019).

Reflection

THE PROJECT PROCESS

I began the project with the assumption that because Mattsson et al published this thesis at the University of Gotehenburg's School of Business Economics and Law that it would be thoroughly vetted for errors by their supervisor (Mattsson, Steinert and Dzemski, 2017). This assumption cost me a tremendous amount of time. Initially, I copied their methodology of preprocessing the data, only to discover that due to errors during the preprocessing stages that their models suffered from data leakages (Machine Learning Mastery, 2016; StackExchange, 2017; Udacity Mentor Review # 1, 2019). With the help of a Udacity mentor, I used a new train-test split methodology to prevent data leakage, as well as create a "Nulls Only" dataset to observe the effect on AUC scores through solely null value imputation (Udacity Mentor Review #1). I also created a new benchmark, at the suggestion of the Udacity mentor, using logistic regression as my benchmark (Udacity Mentor Review #1). After reanalyzing my data, I recognized there was a need to feature scale and oversample all datasets in addition to imputing values on just the "Nulls Only," "One Hot", and "Sum" datasets (Campbell, 2019). After preprocessing, I built my logistic regression benchmark and MLP models. I used several techniques in attempting to improve the AUC scores, but the most significant change that improved the performance of my MLPs was the decrease in the loss optimizer's, stochastic gradient descent, learning rate from .01 to .001 (Campbell, 2019). In fact, this decrease in the learning rate improved the MLP "Nulls Only" model's ability to correctly identify bankruptcy classifications by 112.73%, which was the greatest improvement (Campbell, 2019). In analyzing the final models, all the MLPs performed worse than Mattsson et al's MLPs, as expected (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). However, all the MLP models, except for the "No Nulls" model, failed to beat the logistic regression benchmarks (Campbell, 2019). Furthermore, there was no evidence to suggest that presence of null data improved AUC score performance, and that models with imputed values were merely benefitting from assumed data (Campbell, 2019). Therefore, I rejected these models in favor of the MLP "No Nulls" model because it not only beat its benchmark model, but also it did it on real data, rather than assumed data (Campbell, 2019). Admittedly, this model, with an AUC score of 0.5890, is not a

great model, only beating random chance's ability to make bankruptcy classifications by 8.90 percentage points. However, while it does not do a great job in predicting bankruptcy classifications, I trust the model because it does not suffer from data leakage nor make predictions off assumed data (Campbell, 2019).

THE LESSON LEARNED

I learned a major lesson in completing this project: do not assume that just because the research comes from students from a good university, that it was thoroughly vetted by a professor who knows AI. Had I been more skeptical of Mattsson's et al's assumptions and methodology, I could have completed the project in half the time and not missed two of Udacity's deadlines. While the content of the project was challenging, the most challenging part of the project was finding the time to complete it while working full time and starting a new business.

THE FINAL MODEL

While the final model, the MLP "No Nulls" model, met my expectations, it performed worse than Mattson et al's models while beating the logistic regression benchmark, it's not a great model (Campbell, 2019; Mattsson, Steinert and Dzemski, 2017). It only beat random chance's ability to make a bankruptcy classification by 8.90 percentage points (Campbell, 2019). However, I would use this as a benchmark to predict bankruptcy in polish manufacturing companies for two reasons. First, it does not suffer from data leakage, and second, it uses recorded data, and not assumed data to make its predictions (Campbell, 2019). Although this should be the current benchmark for predicting bankruptcy in Polish manufacturing companies, I highly advise against financial institutions from using it, considering it barely beats random chance (Campbell, 2019).

Improvement

While I believe my MLP “No Nulls” model should serve as the benchmark for predicting bankruptcy in Polish manufacturing companies, I believe it could improve. I only experimented with three loss optimizers before settling on the conventional stochastic gradient descent optimizer (Campbell, 2019). While stochastic gradient descent is the standard, there are many other good optimizers that could have been implemented in the model; in fact, the Keras deep learning library lists seven distinct optimizers (Brownlee, 2016: Keras 2019c). Furthermore, momentum could be used in these optimizers, which is additional velocity added to the loss optimizer that allows it to find the global minimum quicker and avoid local minimas (Bushaev, 2017). The use of different optimizers as well as momentum have improved loss optimization and accuracy scores in deep learning models in the past, and it is likely that these techniques could improve upon my MLP “No Nulls” model (Brownlee, 2016; Campbell, 2019; Keras 2019c).

VI. Appendices

Appendix A

Attribute Information:

X1 net profit / total assets
X2 total liabilities / total assets
X3 working capital / total assets
X4 current assets / short-term liabilities
X5 [(cash + short-term securities + receivables - short-term liabilities) / (operating expenses - depreciation)] * 365
X6 retained earnings / total assets
X7 EBIT / total assets
X8 book value of equity / total liabilities
X9 sales / total assets
X10 equity / total assets
X11 (gross profit + extraordinary items + financial expenses) / total assets
X12 gross profit / short-term liabilities
X13 (gross profit + depreciation) / sales
X14 (gross profit + interest) / total assets
X15 (total liabilities * 365) / (gross profit + depreciation)
X16 (gross profit + depreciation) / total liabilities
X17 total assets / total liabilities
X18 gross profit / total assets
X19 gross profit / sales
X20 (inventory * 365) / sales
X21 sales (n) / sales (n-1)
X22 profit on operating activities / total assets
X23 net profit / sales
X24 gross profit (in 3 years) / total assets
X25 (equity - share capital) / total assets
X26 (net profit + depreciation) / total liabilities
X27 profit on operating activities / financial expenses
X28 working capital / fixed assets
X29 logarithm of total assets
X30 (total liabilities - cash) / sales
X31 (gross profit + interest) / sales
X32 (current liabilities * 365) / cost of products sold
X33 operating expenses / short-term liabilities
X34 operating expenses / total liabilities
X35 profit on sales / total assets
X36 total sales / total assets
X37 (current assets - inventories) / long-term liabilities
X38 constant capital / total assets
X39 profit on sales / sales
X40 (current assets - inventory - receivables) / short-term liabilities
X41 total liabilities / ((profit on operating activities + depreciation) * (12/365))
X42 profit on operating activities / sales
X43 rotation receivables + inventory turnover in days
X44 (receivables * 365) / sales
X45 net profit / inventory
X46 (current assets - inventory) / short-term liabilities

X47 (inventory * 365) / cost of products sold
X48 EBITDA (profit on operating activities - depreciation) / total assets
X49 EBITDA (profit on operating activities - depreciation) / sales
X50 current assets / total liabilities
X51 short-term liabilities / total assets
X52 (short-term liabilities * 365) / cost of products sold)
X53 equity / fixed assets
X54 constant capital / fixed assets
X55 working capital
X56 (sales - cost of products sold) / sales
X57 (current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X58 total costs /total sales
X59 long-term liabilities / equity
X60 sales / inventory
X61 sales / receivables
X62 (short-term liabilities *365) / sales
X63 sales / short-term liabilities
X64 sales / fixed assets

(Tomczak, 2016)

Appendix B

Table 5.2: AUC for the multilayer perceptron with different synthetic features created from the missing values. *None* signifies no synthetic features, *sum* corresponds to one synthetic feature representing the total number of missing values and *1-hot* a dummy variable representation of the missing values.

Synthetic feature	1stYear mean (std)	2ndYear mean (std)	3rdYear mean (std)	4thYear mean (std)	5thYear mean (std)
None	0.77 (0.02)	0.72 (0.02)	0.78 (0.03)	0.77 (0.02)	0.84 (0.03)
Sum	0.83 (0.03)	0.76 (0.02)	0.82 (0.03)	0.83 (0.03)	0.87 (0.02)
1-hot	0.92 (0.03)	0.83 (0.02)	0.87 (0.04)	0.89 (0.03)	0.91 (0.02)

(Mattsson, Steinert and Dzemski, 2017)

Appendix C

The visual assessment below assesses the unique values of the variable “missing_count” in the year two dataset. The first column is the “missing_count” value, or the number of null values in a given row. The second column is the number of rows that had the unique “missing_count” value in the first row. Notice that zero is not in the first column; it is not a unique value. Therefore, there are no null values in the year two dataset.

[340]:
```
# This determines whether 'missing_count'
# was successfully created in clean_yr2.
SUM_yr2['missing_count'].value_counts(dropna = False)
```

[340]:
```
1     3321
2     2777
3     1220
22     726
21     722
23     344
4      328
5      185
24      85
7       77
6       73
25      48
8       32
9       25
26      21
10      21
27      12
29      10
28       9
18       7
17       6
36       6
37       5
16       5
19       4
20       3
13       3
31       2
44       2
30       2
38       2
40       2
15       2
11       1
34       1
35       1
45       1
14       1
56       1
54       1
39       1
Name: missing_count, dtype: int64
```

(Campbell, 2019)

Appendix D

The visual assessment below assesses the unique values of the variable “missing_count” in the year four dataset. The first column is the “missing_count” value, or the number of null values in a given row. The second column is the number of rows that had the unique “missing_count” value in the first row. Notice that zero is not in the first column; it is not a unique value. Therefore, there are no null values in the year four dataset.

```
In [350]:
# This determines whether 'missing_count'
# was successfully created in clean_yr4.
SUM_yr4['missing_count'].value_counts(dropna = False)
```

```
Out[350]:
2     4034
3     3027
25     686
26     624
4      437
5      277
27     140
7       92
6       85
28      52
9       44
8       37
29      27
30      23
10      21
31      14
12      12
32      12
16      10
33       8
11       6
34       6
19       5
20       5
17       3
13       3
14       3
15       3
18       2
23       2
21       2
24       1
41       1
43       1
35       1
50       1
22       1
42       1
36       1
Name: missing_count, dtype: int64
```

(Campbell, 2019)

Appendix E

The visual assessment below assesses the unique values of the variable “missing_count” in the year five dataset. The first column is the “missing_count” value, or the number of null values in a given row. The second column is the number of rows that had the unique “missing_count” value in the first row. See the grey highlighted row below. The highlighted row reads that there are 2,995 rows of data that do not have a null value.

```
In [355]:
1 # This determines whether 'missing_count'
2 # was successfully created in clean_yr5.
3 SUM_yr5['missing_count'].value_counts(dropna = False)

Out[355]:
0     2995
1     2167
2      334
3      159
4       61
5       46
7       26
6       24
8       13
14       7
9        5
20       4
15       2
41       1
17       1
32       1
28       1
16       1
12       1
19       1
Name: missing_count, dtype: int64
```

(Campbell, 2019)

Appendix F

AUC: Area Under the ROC Curve

AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

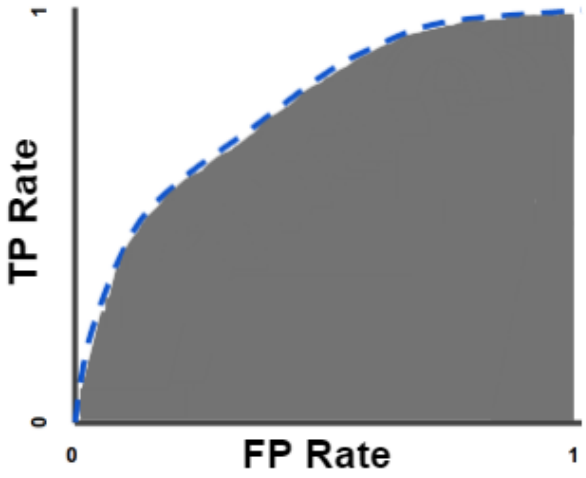
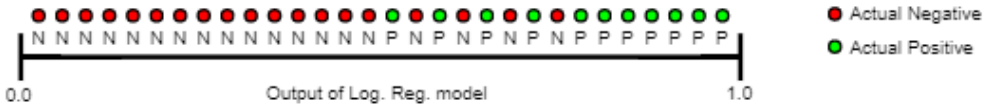


Figure 5. AUC (Area under the ROC Curve).

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:



(Google Developers, 2019)

Appendix G

ROC curve

An **ROC curve** (**receiver operating characteristic curve**) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

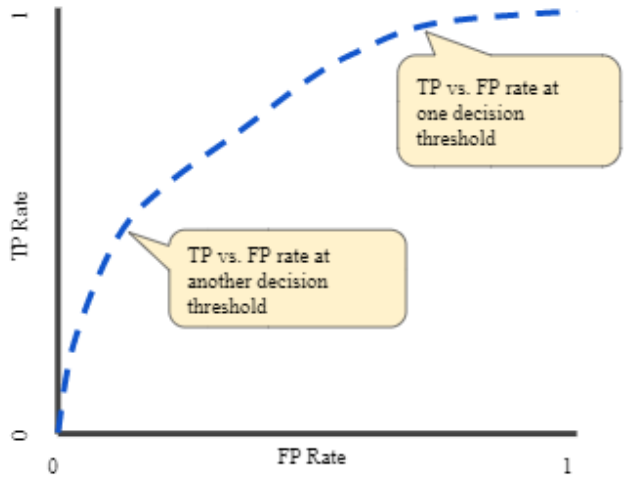
True Positive Rate (**TPR**) is a synonym for recall and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (**FPR**) is defined as follows:

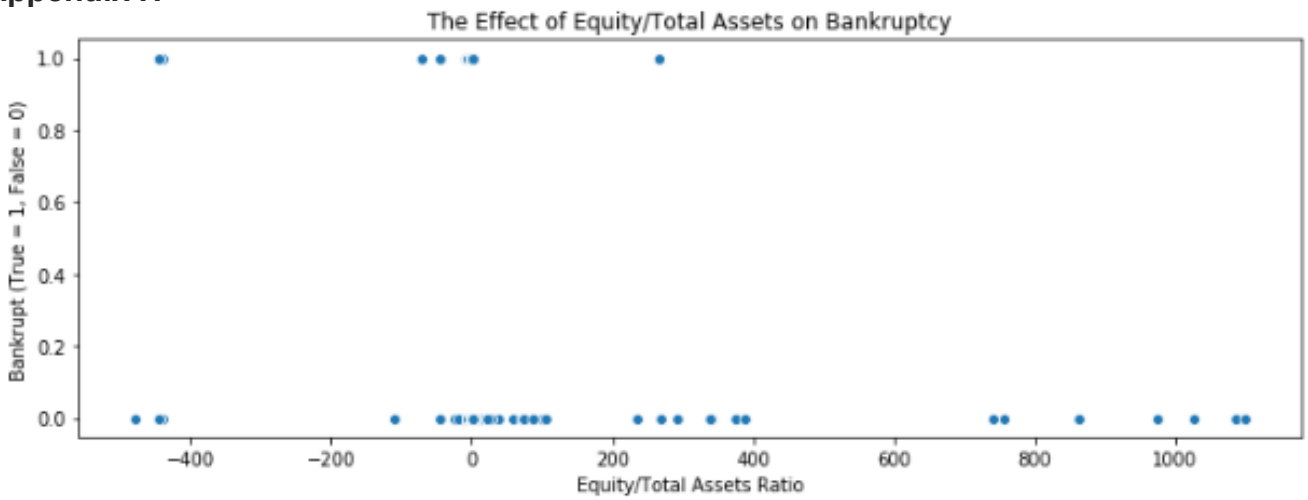
$$FPR = \frac{FP}{FP + TN}$$

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.



(Google Developers, 2019)

Appendix H



(Campbell, 2019)

Appendix I

```
# This imports the necessary libraries for the MLP.

# This imports the sequential model, the layers,
# the SGD optimizer, the regularizers from keras.
# This comes from Reference 5 in Referenes.
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD, RMSprop, Nadam
from keras import regularizers

# This imports checkpointer, which records the best weights
# for the algorithm.
# This comes from Reference 6 in References.
from keras.callbacks import ModelCheckpoint
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

```
def build_model(drop_rate, l2_factor, first_dense, second_dense,
                third_dense, hidden_act, out_act, x):
    dim_int = int(np.size(x,1))
    # This defines the model as a sequential model.
    # This comes from References 1 in References.
    model = Sequential()

    # This is the input layer.
    # This comes from References 1 & 3 in References.
    model.add(Dense(first_dense, activation = hidden_act,
        kernel_regularizer = regularizers.l2(l2_factor),
        input_dim = dim_int))
    model.add(Dropout(drop_rate))

    # This creates the first hidden layer.
    # This comes from Reference 7 in References.
    model.add(Dense(second_dense,
        activation = hidden_act,
        kernel_regularizer = regularizers.l2(l2_factor)))
    model.add(Dropout(drop_rate))

    # This creates the second hidden layer.
    # This comes from Reference 7 in References.
    model.add(Dense(third_dense,
        activation = hidden_act,
        kernel_regularizer = regularizers.l2(l2_factor)))
    model.add(Dropout(drop_rate))

    # This creates the output layer.
    # This comes from Reference 7 in References.
    model.add(Dense(1, activation=out_act))
    # This returns the model.
    return model
```

(Campbell, 2019)

VII. References

Ameisen, E. (2018). Always start with a stupid model, no exceptions.. [online] Medium. Available at: https://blog.insightdatascience.com/always-start-with-a-stupid-model-no-exceptions-3a22314b9aaa [Accessed 23 Nov. 2019].

Becker, N. (2016). The Right Way to Oversample in Predictive Modeling. [online] github. Available at: https://beckernick.github.io/oversampling-modeling/ [Accessed 23 Nov. 2019].

Boris, E. (2019). Demystifying Feature Scaling. [Blog] Medium. Available at: https://becominghuman.ai/demystifying-feature-scaling-baff53e9b3fd [Accessed 8 Nov. 2019].

Brownlee, J. (2016). How To Improve Deep Learning Performance. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/improve-deep-learning-performance/ [Accessed 24 Nov. 2019].

Brownlee, J. (2018). How to Use Weight Decay to Reduce Overfitting of Neural Network in Keras. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/how-to-reduce-overfitting-in-deep-learning-with-weight-regularization/ [Accessed 26 Nov. 2019].

Brownlee, J. (2019). Dropout Regularization in Deep Learning Models With Keras. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/ [Accessed 23 Nov. 2019].

Budhiraja, A. (2016). Learning Less to Learn Better—Dropout in (Deep) Machine learning. [online] Medium. Available at: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 [Accessed 26 Nov. 2019].

Bushaev, V. (2017). Stochastic Gradient Descent with momentum. [online] Medium. Available at: https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d [Accessed 25 Nov. 2019].

Campbell, B. (2019). predicting_bankruptcy. [online] GitHub. Available at: https://github.com/brian-campbell-said-mba18/predicting_bankruptcy/tree/master [Accessed 6 Nov. 2019].

Chandrayan, P. (2017). Machine Learning Part 3 : Logistic Regression. [online] Medium. Available at: https://towardsdatascience.com/machine-learning-part-3-logistics-regression-9d890928680f [Accessed 26 Nov. 2019].

Cochrane, C. (2018). Time Series Nested Cross-Validation. [online] Medium. Available at: https://towardsdatascience.com/time-series-nested-cross-validation-76adba623eb9 [Accessed 23 Nov. 2019].

Cross Validated. (2018). What happens if the test set is too small in machine learning?. [online] Available at: https://stats.stackexchange.com/questions/331117/what-happens-if-the-test-set-is-too-small-in-machine-learning [Accessed 8 Nov. 2019].

Cross Validated. (2018b). why too many epochs will cause overfitting?. [online] Available at: https://stats.stackexchange.com/questions/384593/why-too-many-epochs-will-cause-overfitting [Accessed 24 Nov. 2019].

Google Developers. (2019). Classification: ROC Curve and AUC | Machine Learning Crash Course | Google Developers. [online] Available at: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc [Accessed 6 Sep. 2019].

Investopedia. Efficient Market Hypothesis (EMH) Definition. [online] Available at: https://www.investopedia.com/terms/e/efficientmarkethypothesis.asp [Accessed 4 Sep. 2019].

Kain, N. (2018). Understanding of Multilayer perceptron (MLP). [online] Medium. Available at: https://medium.com/@AI_with_Kain/understanding-of-multilayer-perceptron-mlp-8f179c4a135f [Accessed 26 Nov. 2019].

Karim, R. (2016). Deep Learning via Multilayer Perceptron Classifier - DZone Big Data. [online] dzone.com. Available at: https://dzone.com/articles/deep-learning-via-multilayer-perceptron-classifier [Accessed 26 Nov. 2019].

Karim, R. (2018). Intuitions on L1 and L2 Regularisation. [online] Medium. Available at: https://towardsdatascience.com/intuitions-on-l1-and-l2-regularisation-235f2db4c261#f810 [Accessed 26 Nov. 2019].

Keras.io. (2019a). Activations - Keras Documentation. [online] Available at: https://keras.io/activations/ [Accessed 5 Sep. 2019].

Keras.io. (2019b). Callbacks - Keras Documentation. [online] Available at: https://keras.io/callbacks/ [Accessed 24 Nov. 2019].

Keras.io. (2019c). Optimizers - Keras Documentation. [online] Available at: https://keras.io/optimizers/ [Accessed 5 Sep. 2019].

Machine Learning Mastery. (2016). Data Leakage in Machine Learning. [online] Available at: https://machinelearningmastery.com/data-leakage-machine-learning/ [Accessed 6 Nov. 2019].

Mattsson, B. (2017). bamattsson/neural-bankruptcy. [online] GitHub. Available at: https://github.com/bamattsson/neural-bankruptcy [Accessed 6 Sep. 2019].

Mattsson, B., Steinert, O. and Dzemski, A. (2017). Corporate Bankruptcy Prediction using Machine Learning Techniques. [online] Available at: https://pdfs.semanticscholar.org/cd5f/aff7c02bcba0b3f7f438d4d1c38c3d30d43e.pdf [Accessed 4 Sep. 2019].

McCaffrey, J. (2019). The Difference Between Neural Network L2 Regularization and Weight Decay. [online] James D. McCaffrey. Available at: https://jamesmccaffrey.wordpress.com/2019/05/09/the-difference-between-neural-network-l2-regularization-and-weight-decay/ [Accessed 23 Nov. 2019].

Narkhede, S. (2018). Understanding Logistic Regression. [online] Medium. Available at: https://towardsdatascience.com/understanding-logistic-regression-9b02c2aec102 [Accessed 26 Nov. 2019].

Nicholson, C. (2019). A Beginner's Guide to Multilayer Perceptrons (MLP). [online] Skymind. Available at: https://skymind.ai/wiki/multilayer-perceptron [Accessed 23 Nov. 2019].

StackExchange. (2017). Cross Validated. [online] Available at: https://stats.stackexchange.com/questions/301350/imputing-missing-values-on-a-testing-set [Accessed 6 Nov. 2019].

Scikit-learn.org. (2019a). sklearn.impute.IterativeImputer — scikit-learn 0.21.3 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.impute.IterativeImputer.html [Accessed 23 Nov. 2019].

Scikit-learn.org. (2019b). sklearn.preprocessing.StandardScaler — scikit-learn 0.21.3 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html [Accessed 23 Nov. 2019].

Tomczak, S. (2016). UCI Machine Learning Repository: Polish companies bankruptcy data Set. [online] Archive.ics.uci.edu. Available at: https://archive.ics.uci.edu/ml/datasets/Polish+companies+bankruptcy+data [Accessed 5 Sep. 2019].

Udacity Mentor Review # 1. (2019). Capstone Project: The Efficacy of Multilayer Perceptron Algorithms in Predicting Bankruptcy.

Walimbe, R. (2017). Avoiding look ahead bias in time series machine learning modelling. [online] Linkedin.com. Available at: https://www.linkedin.com/pulse/avoiding-forward-bias-time-series-machine-learning-rohit-walimbe-1 [Accessed 8 Sep. 2019].

Zulkifli, H. (2018). Understanding Learning Rates and How It Improves Performance in Deep Learning. [online] Medium. Available at: https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 [Accessed 24 Nov. 2019].