UNIVERSITY OF GOTHENBURG
SCHOOL OF BUSINESS, ECONOMICS AND LAW

Predicting corporate financial distress

A deep neural network approach

Eklund, Adam
Lundgren, Petter

Thesis: 30 credits
Program: Accounting and Financial Management
Academic level: Master
Semester/year: Spring, 2022
Supervisor: Mari Paananen

Abstract

Background. Predicting bankruptcy is of great importance for creditors, investors and other stakeholders. Early warning signs of financial distress allow stakeholders to take action to minimize the negative consequences of a bankruptcy. Altman Z-score is one of the most used credit scores for predicting financial distress and this study reassesses the Altman Z-score using a neural network approach to compare the accuracy of the models within a 6-year time horizon.

Objective. The objective with this study is threefold. First, we test what hypertuning settings that yield the most accurate result for FDP using a neural network model. The second objective is to test if the accuracy of the Altman Z-score improves when the model is reassessed using the same variables but with a neural network approach. The last objective with this study is to test what happens to the accuracy of the assessed Altman Z-score when it is tested on a global data set.

Method. The method applied on the problem of predicting bankruptcy is a Recurrent Neural network (RNN) with Long short-term memory (LSTM). This is a machine learning technique commonly used for forecasting on time-series but could potentially yield accurate prediction on categorial outputs as well. By feeding the RNN-LSTM with Altman's original financial ratios and applying previous time-steps, the model trains on large amount of data and tests on a perfectly balanced test-sample between bankrupt and non-bankrupt firms within 6 years.

Result. The original Altman Z-score performed 48.91% accuracy for predicting bankruptcy in the manufacturing industry when tested on a domestic data set. Our reassessed Altman model performed an 87.45% accuracy on the same data set. When the data set was extended to a global data set, the Altman Z-score performed a 50.1% accuracy for predicting bankruptcy while our model performed an accuracy of 72.95%.

Conclusion Our result shows that the reassessed Altman Z-score using a neural network method outperforms the original Altman Z-score in terms of accuracy. This finding strengthens the theory that intelligent models, like the neural network, can outperform statistical models, like MDA, by loosening some of the assumptions that must hold true in the statistical models.

The accuracy of the reassessed Altman Z-score decreases in accuracy when it is tested on a global data set compared to a domestic one. This indicates one of two things. Either the increased data points on a global data set is not enough to compensate for the increased heterogeneity, or the variables in our models were originally chosen for a domestic data set and are not well suited for a global data set.

Acknowledgement

First, we would like to thank our supervisor PhD Mari Paananen for her guidance during this report. Her expertise in accounting and bankruptcy has provided us with valuable inputs when completing this report. Secondly, we would like to thank our girlfriends and our families for their continuous support throughout the master thesis.

Table of Content

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
FDP	Financial Distress Prediction
LPM	Linear Conditional Probability Model
LSTM	Long Short-Term Memory
MDA	Multi Discriminant Analysis
NN	Neural Network
OOS	Out of Sample
RNN	Recurrent Neural Network
WRDS	Wharton Research Data Services

1. Introduction

The importance of predicting financial distress and bankruptcy can hardly be overstated as these occurrences bring severe consequences for creditors, investors, managers, and numerous other stakeholders. This has been widely recognized by researchers and financial distress prediction (FDP) has been an ongoing research field for the past five decades, yet the subject is just as important today as it was in its origin. Bankruptcy and financial distress methods to accurately predict financial distress are of great value as it gives an early warning and the chance to act proactively to minimize the potential damage. Yet, existing literature tends to use a fairly short time horizon of 1-2 years, which limits the model's ability to provide early warning signs. This study addresses this issue as we construct a model that manages to predict bankruptcy with 87% accuracy with in a 6-year time horizon.

In the early stage of FDP, discriminant analysis was widely used to establish which financial ratios that the model should include. Beaver (1966) tested the power of accounting data and the explanatory power of financial ratios when predicting failure but did not manage to highlight any ratios with predictive accuracy. In the following decade, researchers continued Beaver's work and made use of classical statistical techniques to build a model for FDP (Beaver, 1966; Altman, 1968; Altman et al, 1977; Meyer & Pifer, 1970; Ohlson, 1980). The most famous model FDP model was introduced by Altman (1968). This model classified companies as bankrupt or non-bankrupt by calculating a credit score based on five financial ratios where each ratio was assigned an individual weight, the model is known as Altman Z-score and was long seen as the gold standard for predicting bankruptcy. However, from the 90's and onwards, technical improvements and digitalization have allowed for more sophisticated models to address the problem.

Predicting financial distress can improve significantly by making use of big data and intelligence models (Bahrammirzaee, 2010). As the world is becoming far more digitalized, we now have access to huge data sets where companies' transactions are stored, e.g., in 2010, Walmart handled more than 1 million transactions per hour and had a database of 2.5 petabytes (Cukier, 2010). This huge amount of data provides opportunities to analyze and make predictions based on large data sets, however, it calls for an automated and efficient way to handle the amount of data available. By utilizing machine learning, you can train the machine to automatically detect and predict patterns in the data set (Murphy, 2012), which could be very

useful in FDP (Bahrammirzaee, 2010; Odom & Sharda, 1990; Chen & Du, 2009). The aim of this study is to test the theory that machine learning can improve FDP. It will do so by reassessing the Altman Z-score, using the same variables as in the original Altman Z-score model but utilizing a machine learning method rather than a multi discriminant model (MDA).

Reassessing the Altman Z-score with a machine learning approach has been attempted before. Odom & Sharda (1990) conducted the first study to reassess the Altman Z-score models using machine learning. The study used the same variables as the original model but utilizes a neural network model (NN) rather than MDA that were used by Altman, this loosened some of the assumptions required in the original case and allowed for a more realistic model. The result suggested that the NN model performed better than the original Altman Z-score in predicting bankruptcy (Odom & Sharda, 1990). However, Odom & Sharda (1990) used a data sample of only 129 companies and the model only predicted bankruptcies one year ahead. In this study, we use a global data set with observations from 137 276 financial statements, and we utilize different hypertuning settings to improve the accuracy for FDP on a longer time horizon. Since 1990, there have been several studies supporting the hypothesis that machine learning models can outperform classical statistical models in predicting bankruptcy (Chen & Du, 2009; Fletcher & Goss, 1993; Leshno & Spector, 1996; Bahrammirzaee, 2010;). Yet, none of these studies have been performed on a global data set.

1.1 Problem description and research questions

Altman Z-score has been revised previously, however, it has never been done on a global data set and it has not been done with our specific model (RNN-LSTM) and its accompanied dataset. Relying on a domestic data set limits the analysis in two ways. First, no general cross-country conclusions can be made about the models as all observations are limited to one country. Secondly, by restricting the data set to a single country you limit the data available to feed the model with. The size of the data set is a significant factor for the ultimate success of any machine learning model (Murphy, 2012). This study will make use of a global data set with 137 276 financial statements, which addresses the issue with data availability. Additionally, we will use the same financial ratios as in Altman (1968), but we will use a deep neural network model (DNN). Specifically, a Recurrent Neural Network with Long short-term memory (RNN-LSTM), designed to predict on larger time-series data sets. Hyperparameter tuning in terms of number of lagged periods, activation function, iterations and more, is a big part of the machine

learning performance and will also be our aim to evaluate in this study. Hence, this study aims to answer the following questions:

1. How does our Recurrent Neural Network model perform in predicting financial distress compared to the original Altman Z-score within a 6-year time-horizon?
2. How does our Recurrent Neural Network model perform when predicting financial distress on a global data set compared to a domestic one?
3. What hypertuning yields the highest accuracy in the reassessed Altman Z-score Neural network model?

The rest of the thesis will be structured in the following way. The next section will contain a literature review, theoretical framework, and hypothesis development. The third section will provide a comprehensive description of our model, the data, and the method we have used. In the fourth section we will present our result. The fifth section will be a discussion where we analyze the finding. In the sixth section we summarize this study with our conclusion, and lastly, we will provide our code.

2. Literature review & theoretical framework

This section will be divided into three parts. First, we will provide a literature review with a detailed review of the original Altman Z-score. Altman (1968) is the model that we will reassess in this study and a proper understanding of the model is therefore needed. Additionally, we will provide a summary of the most relevant literature regarding FDP using machine learning. Secondly, we will include a theory section where we list the theories that motivated us to choose the methodology for this study. Lastly, we will state the hypothesis of what we expect to find.

2.1 The Altman Z-score, (1968).

Altman (1968) developed a score for predicting the likelihood that a manufacturing company would go bankrupt, this score is known as the Altman Z-score and is one of the most well-known models for predicting financial distress. Altman (1968) used multiple discriminant analysis (MDA) to classify the companies as bankrupt or not bankrupt based on financial ratios. The model was originally built using data from 66 manufacturing companies in the US where 50% were firms that went bankrupt during the period of 1946–1965. To minimize the effect of firm size, Altman restricted the non-bankrupt firms to companies with assets above 1 million

dollar and less than 25 million dollars. Initially, 22 different financial ratios were considered. These were the 22 ratios that have proven to be significant indicators of financial distress or corporate problems in previous studies. Out of these 22 ratios, Altman chose the five ratios that performed the best result based on four criteria. (1) observations of the statistical significance, (2) the inter-correlation between the variables, (3) their predictive accuracy and (4) judgment of the analyst. Out of the 22 financial ratios gathered from financial statements, Altman classified these into five standard ratio categories. These categories were liquidity, profitability, leverage, solvency, and activity. The final model of the Altman Z-score is shown in eq. 1.

$$Z = 0.0012X_1 + 0.014X_2 + 0.033X_3 + 0.006X_4 + 0.999X_5 \qquad (1)$$

$X_1 = Working\ Capital/Total\ Assets$

$X_2 = Retained\ Earnings/Total\ Assets$

$X_3 = Earnings\ before\ Interest\ and\ Taxes/Total\ Assets,$

$X_4 = Market\ Value\ of\ Equity/Book\ Value$

$X_5 = Sales/Total\ Assets$

The model then generates a score according to the equation that is equal to the sum of the predetermined weight times the actual ratios. A Z-score closer to zero means that the company is financially distressed while a high Z-score means that the company is financially safe. Altman made some important observations regarding the Z-score that should be mentioned. Altman found that almost all companies with a Z-score below 1.8 went bankrupt, and most companies with a Z-score above 3 did not go bankrupt. His model was very accurate in classifying companies with Z-score below 1.8 or above 3. However, companies with Z-score between 1.8-3 were in a “gray area”. It was in this “gray area” that Altman's model made most of the misclassifications and Altman made two important conclusions about the Z-score. First, Altman concluded that almost all companies with a Z-score below 1.8 went bankrupt. Secondly, the Z-score that determines whether a company should be classified as bankrupt or non-bankrupt was 2.675 in his data set. That is, companies with a Z-score below 2.675 should be classified as bankruptcy and companies with a Z-score above 2.675 should be classified as non-bankrupt in this model.

The model showed 95% accuracy for predicting bankruptcy one year prior to bankruptcy. However, as the time horizon increases, the prediction accuracy decreases sharply. Two years

prior to bankruptcy the accuracy dropped to 75%, 3 years prior to the bankruptcy the accuracy was 48%, four and five years prior to the bankruptcy the accuracy was 29% respectively 36%.

Critics claim that the model's inability to account for deferred income and when companies make large one-time write downs are shortcomings of the Z-score that can lower the accuracy. Furthermore, the Z-score produces a score with a weak intuitive interpretation, and the Z-score is only applicable for linear classification problems (Zhang et al. 1999).

There have been several suggestions of classification techniques to predict financial distress and the data usually have been gathered from financial statements. Research has been conducted using linear conditional probability models (LPM) and regression analysis such as logit and probit models. Meyer and Pifer (1970) implemented a LPM for predicting bankruptcy while Martin (1977) and Ohlson (1980) used logit regression analysis to predict bank failure and business failure, respectively. In summary, the time after the development of the Z-score there have been univariate (Beaver, 1966), multiple discriminant (Altman, 1968), multiple regression (Meyer & Pifer, 1970), logistic regression (Dimitras, Zanakis, & Zopounidis 1996), factor analysis (Blum, 1974) and more techniques that have been used in several studies. However, the strict assumptions in ordinary linear statistics such as normality, linearity and independence among the independent variables limits the applications in the real world. These issues can be mitigated by using a machine learning approach.

2.2 Bankruptcy prediction using neural network

Wilson & Sharda (1994) reassess the original Altman Z-score model using a neural network method, by utilizing a neural network approach they could loosen some of the assumptions that must hold true for MDA. Their sample consisted of 129 US manufacturing companies where 64 were classified as bankrupt and 65 were classified as not bankrupt, the companies were collected between 1975-1982. The model used the same variables as the Altman Z-score, however, the prediction horizon was only tested for one year prior to the bankruptcy. Wilson & Sharda (1994) found that neural networks outperformed MDA in all aspects, with an accuracy as high as 97% in classifying bankrupt companies one year prior to the bankruptcy. However, the short time horizon of only one year is an obvious limitation for this study.

Since the 90s, NN has shown promising capabilities of predicting financial distress and can be considered the model with the highest predictive ability (Jeong et al. 2009). Table 1

demonstrates recent studies that implemented neural networks when predicting financial distress and bankruptcy, showing the accuracy of the specific research implemented in NN, where accuracy is classified as the number of times the model accurately predicts a firm failure, i.e., financial distress. The result is quite remarkable in terms of its high values and is thought to be soundproof of the capabilities of neural networks performance in a more complex nature of financial ratios.

Table 1: Summary of papers using Neural Network for FDP

Authors	***Year***	***Country/region***	***Accuracy of NN***	***Time Horizon***
Fletcher and Goss	*1993*	*US*	*89%*	1 year
Wilson and Sharda	*1994*	*China*	*97%*	1 year
Zhang	*1999*	*US*	*88%*	1-2 years
Chen and Du	*2009*	*Taiwan*	*82%*	2 years
Paule-Vianez	*2020*	*Spain*	*97%*	1 year
Gregova	*2020*	*Slovakia*	*91%*	1 year
Altman	*2020*	*Finland*	*94%*	1-2 years

The research has been conducted over several different countries and with different sets of financial variables. The accuracy of the neural networks in predicting financial distress has a range between 82 – 97%. One must keep in mind that these studies also differ in terms of horizon i.e., the time until the firm experiences financial distress or file for bankruptcy. The studies often use a short time horizon of one or two years. However, all studies conclude that artificial neural network (ANN) outperforms ordinary statistical methods, logit regression, logistic regression, support vector machine, decision tree, random forest, and gradient boosting techniques in terms of accuracy when predicting financial distress.

As shown in Table 1, Altman (2020), almost 50 years after the development of Altman's Z-score, conducted a comparative study of the accuracy and efficiency of five different methods for predicting financial distress in Finland. The research showcases that ANN once again proves to be a better predictor than other methods. However, the article describes limitations and implications of the development of an effective prediction model for a longer time horizon and concludes that it is a very challenging task due to the instability of financial ratios, fluctuations of the economic cycle. As financial ratios are often unstable over time and might not be reliable in terms of annual information, the need for non-financial measures is important. The problem is that it is hard to compare these limited measures over different countries. Therefore, previous research focuses on single countries to eliminate this issue (Altman, 2020).

2.3 Theoretical framework

2.3.1 Linear Discriminant Analysis

Linear discriminant analysis (LDA) was first presented by Fischer (1936) and is a statistical technique for classification where LDA separates two or more classes in two categories. LDA finds a linear combination that maximizes the mean between the groups while it minimizes the variation within the groups. LDA requires the independent variables to be continuous variables and the dependent variable to be a categorical variable (Green, Salkind, & Akey, 2000; Poulsen & French, 2008). This criterion holds true in the Altman Z-score as the financial ratios used as the independent variables are continuous variables while the dependent variable, in this case whether a company is bankrupt or not bankrupt, is a categorical variable.

For LDA to yield an efficient result, it must hold true for certain assumptions. These assumptions are multivariate normal distribution, homoscedasticity, multicollinearity, and independence. Huberty (1975) claims that LDA is rather resilient towards violations of the assumptions. However, Büyüköztürk & Çokluk (2008) argues otherwise and claims that one of the shortcomings in LDA is that it is difficult to meet all the assumptions, which lowers the efficiency of the model. This is supported by both Wilson & Sharda (1994) and Ahmed et al. (2010) who argue that one of the advantages of the intelligent models compared to the LDA is that the intelligent models loosen some of the assumptions that must hold true in statistical models.

2.3.2 Neural networks

Neural networks are a type of machine learning, and the name and structure are based on the functionality of the human brain with neurons interconnected through a web of layers. The figure below shows how a DNN looks with its neurons and layers. (IBM, 2020)

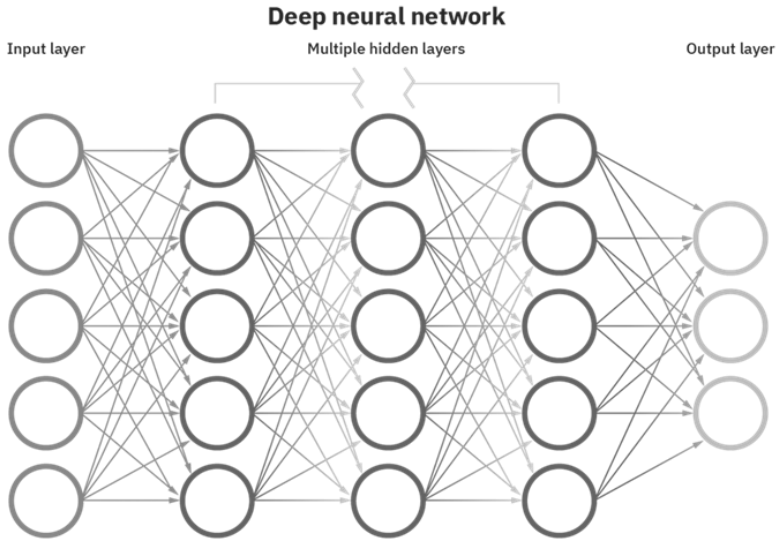

Figure 1: Deep Neural Network

Neurons can be viewed as an own univariate regression function with an intercept and a coefficient i.e., the weight. The equation of each neuron is made up of the summation of neurons from the previous layer multiplied by the weight, and at last a bias is added.

$$N_k = \sum_{i=0}^{k} (I_k * W_k) + b_k \tag{2}$$

Eq. 2 shows the sum of the previous layers neurons times the weight plus a bias within each neuron. The ANN are constructed of hidden, input and output layers all with its own set of neurons, within the neurons an activation function exists with a specific threshold. The threshold is like a gatekeeper that only passes through an output that is above the specific threshold. If the output is passed through, the neuron is considered activated and sends the data towards the next layer.

2.3.2.1 Logistic regression

Logistic regression is a popular algorithm within machine learning and supervised learning techniques. The main objective of the regression is to predict a categorical dependent variable with the help of specifically chosen independent variables. The regression function predicts the categorial output which could either be a zero or one and return a probability of either classes occurring. Unlike the ordinary linear regression which predicts a continuous number, the logistic regression uses a specific activation function in the neural networks final layer. In the case of this study a binary logistic regression is a more appropriate description in relation to the problem when predicting whether a company fails or not within a specific time-horizon.

2.3.2.2 Activation function

The most common activation function in a binary logistic regression neural network is the sigmoid activation function. The activation function map predicted values to a specific probability between 0 or 1. The sigmoid function uses a specific decision boundary or threshold value which classify values into their different classes.

$$p \geq 0.5, Class = Bancruptcy$$
$$p < 0.5, Class = No\ bancruptcy$$

If the model returns a percentage below the specified threshold the model interprets that value as the probability of the specific class occurrence. Eq. (3) below shows the mathematical model behind the sigmoid function.

$$S(z) = \frac{1}{1 + e^{-z}} \quad (3)$$

s(z) = Output between 0 and 1 i.e the probability of classification.

z = Variable input to the function

e = Base of natural log

With this activation function and threshold predetermined, a prediction can be made, and a loss function calculates the model's performance in the validation process.

2.3.2.3 Loss function & accuracy

The loss function in a binary classification problem is called Cross-entropy or Log loss. Each of the predicted probability of a class occurrence is compared to the actual desired output of zero or one, where a loss value is calculated and penalized of the distance from the actual expected value. When the predicted outcome is further away from the actual value, the log loss will increase rapidly and is therefore important to measure when evaluating the neural network. The loss is calculated according to eq.4 below.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} \quad y_i \, log(p(y_i)) + (1 - y_i) log(1 - p(y_i)) \tag{4}$$

The computed accuracy within this type of NN with a binary output, is simply the total of a predicted class divided by the total count of actual values. This frequency is returned as a binary accuracy.

2.3.2.4 Overfitting

There are major limitations to consider when building ANN. If the model has too little capacity it cannot learn the real-world problem, and at the same time, with too much capacity the model learns too well on the training data. The latter is called overfitting, and it happens when the model has learnt the time-series training data too well and performs badly on test data.

- **Underfit Model**. The model fails to learn the problem and performs poorly on training data.
- **Overfit Model**. The model succeeds in learning the problem and performs well on training data but fails on the out of sample test data.
- **Good Fit Model**. The model performs well in both cases.

A good fit model can be considered in a bias- vs variance trade-off where an underfitting model has a low variance but high biasness. An overfit model has a low bias and high variance. Too much capacity in terms of overfitting means that the model puts too much emphasis on noise because of the high complexity in the underlying data, which can be due to many variables with little correlation with the problem. To overcome this issue, one should either reconsider changing the data or introducing regularization. In our model, we implement two techniques of regularization called early stopping, dropout-rate and batch normalization. The early

stopping monitors the model's performance in a validation set and stops the process of training when performance starts to decline. The drop-out rate is a percentage of random neurons that is skipped and reduces the sensitivity of specific weights which solves the problem of co-adaptation which happens when neurons are too heavily dependent on the inputs, they receive. Finally, batch normalization normalizes each mini subset or batch within the whole dataset to ease the computer capacity needed and the speed of the training process.

2.3.2.5 Optimizers

During the training of the neural network, it is important to tweak the different weights to make as correctly predictions as possible. The question is to which extent that the model can tweak and change the parameters of the model and when it should do it. This is where optimizers come to play and tie together the loss function and model parameters and update the model. There are numerous different optimizers that change and update the model as accurately as possible for the specific purpose and problem. The loss function, explained earlier, is the optimizer's guidance and tells whether it does its job correctly or not.

In this study, two optimizers are considered thanks to its popularity and easy usage within neural networks. Adaptive Moment Estimation (Adam) and Stochastic Gradient Descent (SGD) is compared to ultimately find the best fit for the specific purpose. SGD and ADAM are both optimizers possible to use in a binary classification problem. It is possible to tune the learning rate parameter that tells the optimizer how far to change the weights in the layer.

2.3.2.6 Architecture

A common architecture of ANN is the recurrent neural network (RNN) which is useful for drawing conclusions regarding past events within the data thanks to a temporal dimension. Through a feedback loop the model contains memory of past performance and takes it into consideration as it begins with the process again. This network uses its memory to strengthen the predictions and accuracy and excels in modeling patterns within the sequential data. The network copies the same structure several times and uses each replica as the input for the next iteration, and the output also contains information about the past learning process as well as the current state. Ordinary neural networks are called feed-forward neural networks and assume that the variables are independent of each other, however if data is structured through a sequence for example in our case using "lagged" variables then we need to modify and

incorporate the dependencies between these lags. RNN can do this by storing previous information within the so-called multi-layer perceptron that has an additional time variable.

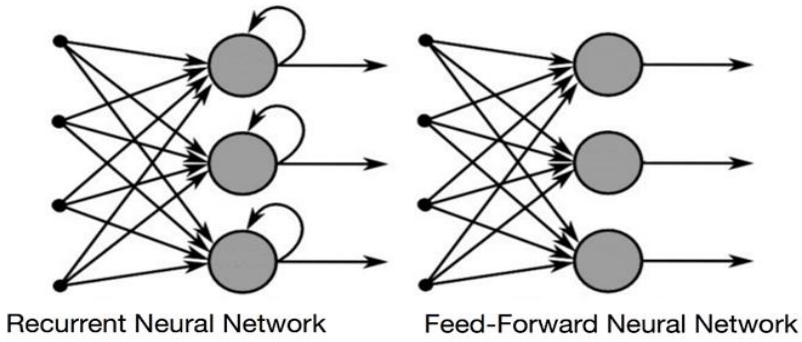

Figure 2: Recurrent vs Feed-forward network

A subgroup to RNN is the long short-term memory neural network (LSTM) which is structured with gates that restrict which specific value that can pass through the layers. LSTM is considered one of the most successful models when modeling time series data and is thus sutible for our study since we essentially try to predict indicators of financial distress through lagged time series values. LSTM is a popular architecture of the RNN, and it was first introduced as a solution to the vanishing gradient problem. This problem stems from the fact that backpropagation and gradient-based learning methods use iteration, and each weight receives an update proportionally to the partial derivative of the error function in respect to the previous weight. In some cases, this change in weight is so small that it could completely stop the training process. LSTM solves this issue by addressing the problem of long-term dependencies and uses "cells' in the hidden layers with input, output and forget gates. These gates essentially control the information flow between the cells.

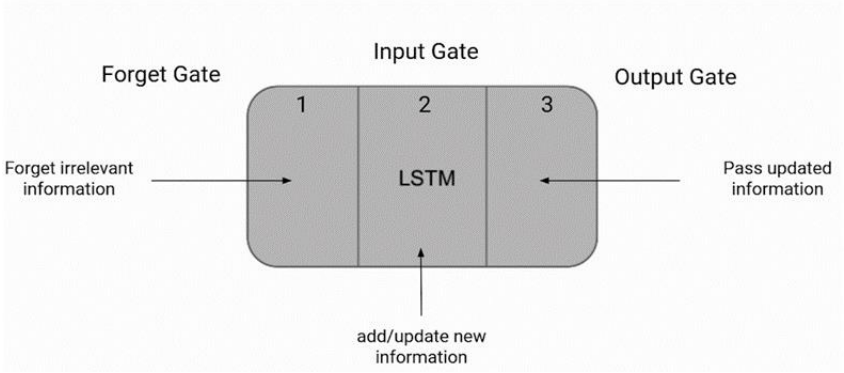

Figure 3: Shows the gates within the LSTM Cell that process the information differently.

2.4 Hypothesis

Accuracy: A model like the neural network loosens some of the assumptions from the MDA and we expect that an ANN model will increase the accuracy of MDA models. Therefore, we state the following hypotheses:

Hypothesis 1a (H1a). *A deep NN model yields higher accuracy for bankruptcy predictions than the Altman Z-score.*

Global data set: The heterogeneity in a global data set will most likely lower the accuracy of the model. However, by using a global data set the data available increases significantly and allows us to make predictions based on more observations, which should improve the performance of a neural network model. We expect the increased data points to make up for the disturbances in a global data set. Furthermore, we believe that the NN model will benefit more from the increased data points than the Altman Z-score. Therefore, we state the following hypotheses:

Hypothesis 2a (H1a). *The discrepancy in accuracy between a NN model and the Altman Z-score increases when a global data set is used.*

Hypothesis 2b (H2b). *A deep neural network model will yield at least the same accuracy on a global data set as on a domestic data set.*

3. Method

3.1 The Model

Our model will replicate the Altman Z-score and use the same variables as Altman (1968), but with the important difference that we will utilize a RNN approach. In a RNN model, the model itself assigns weights to the variables by optimization constraints rather than using the predetermined weights in the Altman Z-score. Our model will train on a large training dataset and try to classify the probability of bankruptcy within 6 years. A neural network is often referred to as a black box since it is not possible to see the predictive power or weights of the specific ratios. However, it is possible to closely monitor the accuracy of the model whenever ratios, parameters or data is changed. In our case we will use the following structure of ratios where the RNN will train and create our prediction model like eq 5 below.

$$Z = W_1 X_1 + W_2 X_2 + W_3 X_3 + W_4 X_4 + W_5 X_5 \quad (5)$$

$X_1 = Working\ Capital/Total\ Assets$
$X_2 = Retained\ Earnings/Total\ Assets$
$X_3 = Earnings\ before\ Interest\ and\ Taxes/Total\ Assets,$
$X_4 = Market\ Value\ of\ Equity/Book\ Value$
$X_5 = Sales/Total\ Assets$
$W_i = assigned\ weights\ from\ the\ optimization\ constraint$

The financial ratios used in the Altman Z-score model have proven to be efficient variables for predicting financial distress. However, although MDA models are widely used for classification problems, their effectiveness can be limited by the statistical assumptions that must hold true in MDA and by using an ANN model, you loosen these assumptions. We expand the work of previous research such as Altman (2020) by using a RNN as its architecture thanks to its backpropagation capabilities and long-term memory. By applying this methodology on a high-dimensional data, covering multiple lagged variables ranging between different nations, we expect the model to achieve at least the same accuracy thanks to the RNN capabilities to detect patterns in historic data even though a longer prediction time-horizon is used.

The ANN is created with the programming language Python with its packages Pandas, SciPy, Numpy, Matplotlib and the High-level API Keras on top of Tensorflow. The programming code contains the data preparation, splitting and reshaping and construction of the actual ANN. In the previous section we highlighted some key aspects of logistic regression neural networks and the architecture of RNN with LSTM. Our model will use a setup based on these key components such as the optimizers ADAM and SGD as well as the loss function, binary cross entropy or log loss. To evaluate the model, we simply use the accuracy metric which shows how many correct guesses the model makes.

3.1.1 Variable selection

The variables in our model will be the same as in the Altman Z-score (1968) as we are replicating his model but with a RNN approach. The dependent variable is a categorical variable that can take the values of either 1, if the company should be classified as bankrupt, or 0 if the company should be classified as non-bankrupt. The definition that has been used for bankruptcy in this study is when a company is under liquidation or has filed for bankruptcy. All data is gathered from Wharton Research Data Services (WRDS) and the data tables CompuStat, containing global and North American annual financial statements. All ratios are calculated from CompuStat data bae. The independent variables used in this study are:

$X_1 = Working\ Capital/Total\ Assets$

$X_2 = Retained\ Earnings/Total\ Assets$

$X_3 = Earnings\ before\ Interest\ and\ Taxes/Total\ Assets,$

$X_4 = Market\ Value\ of\ Equity/Book\ Value$

$X_5 = Sales/Total\ Assets$

3.2 Hyperparameters

When initializing the training process and evaluating the result on the test data, it is often not possible to create an optimal model on the first base run. However, it is possible to increase the model's accuracy by fine tuning the hyperparameters. Hyperparameters are determined prior to the model and have a big effect on the outcome. Since we are using a LSTM, we will consider changing hyperparameters that have the most effect on this type of architecture. A big part of our findings is presented with these changes in hyperparameters and comparison between the different models essentially shows the optimum result for our specific dataset.

Following list contains the hyperparameters that is tuned during the process of training the model's:

1. Number of nodes and hidden layers
2. Number of nodes in a layer
3. Batch normalization
4. Dropout-rate
5. Number of lags
6. Prediction horizon

These hyperparameters create and structure the ANN. The depth of the network is dependent on the number of hidden layers and the complexity is based on the numbers of neurons or units within the layers. All LSTM networks must contain a dropout-rate that prevents overfitting in training by randomly bypassing neurons and reducing the sensitivity of specific weights. Therefore, adding this dropout-rate could decrease the potential overfitting and still allow the model to be complex in terms of more layers and neurons. The number of lags will determine how many periods in the past that the model should use to be able to predict the future, whilst the prediction horizon is how many periods into the future the model should be able to predict. All these parameters influence the model performance and are highlighted in the result section.

3.3 Model evaluation

As mentioned earlier, the model produces both a loss value as well as an accuracy both in-sample and out-of-sample (OOS). In sample is simply the data used for training and an in-sample performance metric is calculated. The loss indicates the model's training performance and should decrease on every epoch until convergence occurs. The in-sample accuracy highlights how many correctly guessed bankruptcies and non-bankruptcies exist in relation to the total number. The out-of-sample accuracy is calculated in the same way, but we will add a function that calculates the four different types of correct and wrong predictions the model makes and compare this to the original model of Altman. Since the sigmoid output layer produces a probability of a class occurrence, we just simply convert this probability to its nearest binary value, 1 for bankrupt and 0 for non-bankrupt. See table 2 for a demonstration of the four different correct or false predictions.

Table 2: Evaluation matrix for the model

Actual outcome	**Predicted Outcome**	
	Bankrupt	Non-Bankrupt
Bankrupt	Correct	Type II
Non-Bankrupt	Type I	Correct

To evaluate our result, we will first run the original Altman Z-score and use that as a benchmark. Then, we run our model on the same data set and compare the accuracy of our reassessed Altman Z-score model to the accuracy of the original Altman Z-score. We will also show the evolution of our model and demonstrate which setup yielded our specific result.

3.4 Data

3.4.1 Metadata

We gather our dataset through a connection through Wharton Research Data Services (WRDS) and accessing the data table CompuStat containing global financial yearly data. Through an API-request we receive over 1.3 million firm year observations of financial data ranging from multiple companies, countries, years, and sectors. Since Altman's (1968) original study was conducted on manufacturing firms, we filter down our dataset to the same subset by looking at each company's annual statement sic codes, and only pick those representing the manufacturing industry. At first glance, the dataset was unbalanced over time and a stable number of firms reporting financial data did not start until 1985 to 2021. Therefore, we made the choice to only use data from 1985 forward.

The dataset contains a total of 1,920 (2.5%) bankrupt firms out of a total 73,072 firms. The dataset includes 124 countries. North America is approximately 32% of the total and the rest 67%. USA is the number one largest part of the total dataset (28%) followed by Japan, India, China which together comprise a total of 52% of the dataset. Before highlighting the description of the dataset alone, some preprocessing must be done.

3.4.2 Preprocessing

Data preprocessing is a crucial step in machine learning as the quality of the data and information contained directly impacts the efficiency and accuracy of the model. We must preprocess our data before using it as an input in our model. We are handling the null values, standardization and creating the categorial value before entering the training process. The preprocessing part reduces the number of financial statements a lot because machine learning techniques often performs better with a balanced dataset, i.e., equal number of bankrupt firm's vs non-bankrupt. This is true in our case since we have equally many bankrupt vs non-bankrupt firms in both our training and test data. This balancing is done by random selection of bancrupt and non-bancrupt firms without replacement, which means that no company will exists twice in either of the training or test sample to avoid overfitting. The final dataset will therefore depend on the number of lags and time horizons. As we increase the interval before bankruptcy more data is available for the model to train on.

3.4.2.1 Categorical output

As our goal is to predict the probability of bankruptcy within specific horizons, we create a variable named “deleted” indicating that the company will in the future go bankrupt. The question arises “when will this occur?”. The dataset contains a cut date, in other words the year when the company went bankrupt and was excluded from the database. By calculating the difference between this variable and the year of the financial statement we receive a variable that shows the number of years until bankruptcy. We use this variable in our different prediction horizon scenarios. For example, when we want to train our model to predict bankruptcy within 6 years, we set our target variable to equal one when the year until bankruptcy equals two. This is called a categorical variable and is often referred to in machine learning as a classification problem since the goal is to classify the predictors to a corresponding category of either one or zero, i.e., bankrupt or non-bankrupt.

3.4.2.2 Normalization and winsorization

The process of normalization and winsorization is straightforward thanks to prebuilt packages within python. Machine learning works much better when the features or independent variables are on a relative similar scale to one another and normally distributed. We do this by standardizing the data so that the standard deviation will be equal one. We use the prebuilt standardscaler that standardizes each feature by subtracting the mean and scaling it to the unit variance. Unit variance is the same as dividing the values by the standard deviation. The

winzorarization is done by cutting off each end of the normal distribution tail with predetermined quantiles of 95% and 5%. We chose these quantiles to minimize the negative effect of outliers.

3.4.2.3 Splitting the data

As mentioned before, to evaluate the model's performance both in-sample and out-of-sample we need to have different sub-samples of the dataset. A common mistake in this process when using lagged variables is the effect of spill-over, which means that the subsamples contain duplicates of the same data from the same periods. We solve this issue by populating the different sub-samples with unique firms, which means that the model trains on one set of firms independent of time and tests on completely new set of firms. Normally this process could be done with different sizes of training and testing data, we chose a commonly used splitting of 80% training and 20% testing data.

3.4.3 Final dataset

As mentioned earlier some heavy preprocessing of the metadata has been conducted and two datasets have been created, one containing all the financial data of bankrupt firms and the other with non-bankrupt financial data. After preprocessing we get the following distribution of financial statements per country, see Figure 4. It shows that the USA is still the number one most common country in the dataset with Japan, India, China, and Canada as approximately equal in size as the USA. The remaining part of 48% is other countries with smaller portions within the data. From 128 countries the dataset now contains 81 unique countries.

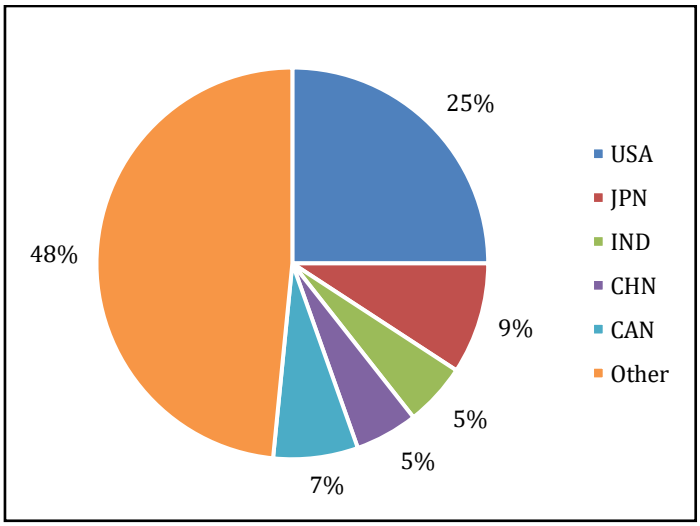

Figure 4: Financial statements per country

We can also demonstrate the number of financial observations between the periods represented in the Figure 5 below. We see an increase from 1985 to 2021.

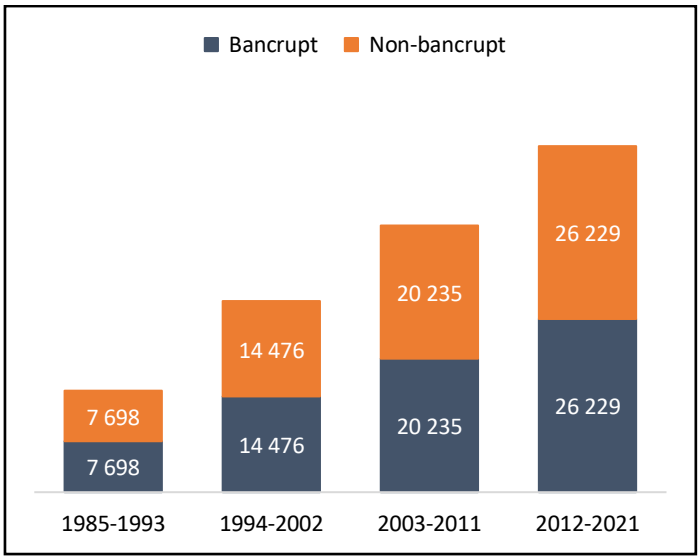

Figure 5: Financial statements per yearly interval

Lastly, we highlight the characteristics of the final dataset containing the different financial variables. We do not highlight the dependent variable since the balance between zeros and ones are 50/50 and therefore the mean is stable at 0.5. However, the five ratios after winsorization and standardization showed the following characteristics, in table 3 below.

Table 3: Altman's ratios on a global dataset after winsorazation and standardasation

	A	**B**	**C**	**D**	**E**
Count	137 276	137 276	137 276	137 276	137 276
Mean	0,47	0,88	0,77	0,16	0,42
Std	0,23	0,14	0,15	0,19	0,23
Min	0	0	0	0	0
25%	0,33	0,86	0,71	0,05	0,25
50%	0,47	0,89	0,77	0,09	0,38
75%	0,63	0,94	0,85	0,18	0,55
Max	1	1	1	1	1

4. Results

The following section will show our result from the testing. First, we demonstrate the evolution of our model and the fine tuning of parameters, as well as the findings during the process of training and evaluating the model's performance. Then we present the result from the two benchmark models using the Altman Z-score with different types of relaxations from the original study. The first benchmark shows the Altman Z-score accuracy on our dataset but filtered on a single country like Altman's first study in 1968. In section two we highlight the difference in accuracy when using a global dataset. Lastly, we compare the accuracy of our model when it is tested on a domestic data set compared to a global data set.

4.1 Evolution of the neural network

The initial setup of our model used the optimizer SGD, a dropout rate of 0.2, batch normalization and 50 epochs. The number of lagged variables with different time prediction horizons was tested to see which combination yielded the best out-of-sample accuracy and minimum loss.

At the first nine tries we saw that a deeper neural network performed with the highest out-of-sample accuracy but with a high loss. However, this model showed signs of overfitting since the relationship between the loss function and the out-of-sample accuracy did not align with each other. As mentioned before, a higher loss indicates that the model is not learning from the input data but still it is able to predict on the out-of-sample testing with at least 60% accuracy. By just randomly picking the number of lags and prediction horizon we still received a model that is better than just randomly guessing if a firm fails within different time horizons or not. See result in Table 4.

Table 4: First run of training the model

Horizon	Lags	epochs	Batch norm	Batch size	Optimizer	Layers	In-sample Accuracy	In-sample Log loss	Validation Accuracy	OOS Accuracy
5	5	50	TRUE	32	SGD	[128, 64, 64, 32,32,16,16,8 ,8,4,4, 1]	66,4%	65,0%	63,0%	62,8%
5	5	50	TRUE	16	SGD	[128, 64, 32,16,8,4, 1]	62,0%	64,0%	63,0%	64,7%
5	4	50	TRUE	16	SGD	[64, 32,16,8,4, 1]	62,0%	65,0%	64,0%	64,2%

5	4	50	TRUE	16	SGD	[32,16,8,4, 1]	61,0%	66,0%	63,1%	63,7%
5	3	50	TRUE	16	SGD	[16,8,4, 1]	62,5%	65,0%	63,9%	64%
10	5	50	TRUE	32	SGD	[128, 64, 32,16,8,4, 1]	61,0%	65,0%	59,0%	59%
5	10	33	TRUE	12	SGD	[64, 32,16,8,4, 1]	63,4%	62,0%	59,0%	63,0%

In the second round we wanted to test the effect of changing the optimizer within the network from SGD to ADAM with a learning rate of 0.001 and lowering the number of epochs that the model uses for training, since we noticed that the model started to overfit a bit when running for too long. As seen in the table 5, we increased the number of lags and prediction horizon to see the effect on performance and concluded that the best possible combination of the two was between 5 to 7.

Table 5: Second run of training the model

Horizon	Lags	epochs	Batch normalization	Batch size	Optimizer	Layers	In-sample Accuracy	In-sample Log loss	Validation Accuracy	OOS Accuracy
6	6	22	TRUE	32	ADAM 0.001	[64, 32,16,8,4, 1]	65,1%	63,0%	64,9%	66,0%
7	7	33	TRUE	32	ADAM 0.001	[64, 32,16,8,4, 1]	66,0%	61,0%	65,0%	67,5%
5	8	33	TRUE	32	ADAM 0.001	[32,16,8,4, 1]	65,0%	62,2%	65,0%	64,4%
5	10	33	TRUE	32	ADAM 0.001	[64, 32,16,8,4, 1]	67,5%	60,8%	66,7%	65,0%
10	10	33	TRUE	12	ADAM 0.001	[128, 64, 32,16,8,4, 1]	68,0%	59,3%	63,9%	64,5%
7	7	33	TRUE	12	ADAM 0.001	[64,32,16,8,4, 1]	66,0%	62,1%	66,7%	67,0%

We continued with the last setup of the last model in Table 5 and removed the batch normalization. We found the best model so far with an OOS-accuracy of 68.2% that was able to predict bankruptcy within 7 years and using the development of Altman's Z-score ratios from 7 time periods back. In our final test round, we increased the number of epochs to 128 and tried different combinations of lags and time-horizons. See result in Table 6.

Table 6: Third run of training the model

Horizon	Lags	epochs	Batch normalization	Batch size	Optimizer	Layers	In-sample Accuracy	In-sample Log loss	Validation Accuracy	OOS Accuracy
7	7	33	FALSE	32	ADAM 0.001	[64,32,16,8,4, 1]	64,3%	58,9%	65,8%	68,2%
7	7	128	FALSE	32	ADAM 0.001	[64,32,16,8,4, 1]	75,6%	49,2%	70,7%	69,8%
8	8	128	FALSE	16	ADAM 0.001	[64,32,16,8,4, 1]	78,0%	48,3%	71,4%	72,6%

6	6	128	FALSE	16	ADAM 0.001	[64,32,16,8,4, 1]	84,8%	37,7%	69,0%	70,0%
6	6	128	FALSE	16	ADAM 0.001	[128,64]	92,3%	25,2%	73,0%	72,9%

This, at last yielded the highest OOS-accuracy with the lowest loss with a network only containing one hidden layer with 64 neurons and an input layer with 128 neurons. This in terms is called a shallow network instead of a deep neural network which in some cases could be more applicable depending on the nature of the dataset. This model used a time horizon of 6 years and 6 lags.

With an OOS-accuracy of 72.9% on a global dataset and a loss of 25.2% we reached our final model. Table 7 shows how this model was reached with all the different runs demonstrated in chronological order from top to bottom. See table 7.

Table 7: Evolution of our model in chronological order

Horizon	Lags	epochs	Batch normalization	Batch size	Optimizer	Layers	In-sample Accuracy	In-sample Log loss	Validation Accuracy	OOS Accuracy
5	5	50	TRUE	32	SGD	[128, 64, 64, 32,32,16,16,8 ,8,4,4, 1]	66,4%	65,0%	63,0%	62,8%
5	5	50	TRUE	16	SGD	[128, 64, 32,16,8,4, 1]	62,0%	64,0%	63,0%	64,7%
5	4	50	TRUE	16	SGD	[64, 32,16,8,4, 1]	62,0%	65,0%	64,0%	64,2%
5	4	50	TRUE	16	SGD	[32,16,8,4, 1]	61,0%	66,0%	63,1%	63,7%
5	3	50	TRUE	16	SGD	[16,8,4, 1]	62,5%	65,0%	63,9%	64%
10	5	50	TRUE	32	SGD	[128, 64, 32,16,8,4, 1]	61,0%	65,0%	59,0%	59%
5	10	33	TRUE	12	SGD	[64, 32,16,8,4, 1]	63,4%	62,0%	59,0%	63,0%
6	6	22	TRUE	32	ADAM 0.001	[64, 32,16,8,4, 1]	65,1%	63,0%	64,9%	66,0%
7	7	33	TRUE	32	ADAM 0.001	[64, 32,16,8,4, 1]	66,0%	61,0%	65,0%	67,5%
5	8	33	TRUE	32	ADAM 0.001	[32,16,8,4, 1]	65,0%	62,2%	65,0%	64,4%
5	10	33	TRUE	32	ADAM 0.001	[64, 32,16,8,4, 1]	67,5%	60,8%	66,7%	65,0%
10	10	33	TRUE	12	ADAM 0.001	[128, 64, 32,16,8,4, 1]	68,0%	59,3%	63,9%	64,5%
7	7	33	TRUE	12	ADAM 0.001	[64,32,16,8,4, 1]	66,0%	62,1%	66,7%	67,0%
7	7	33	FALSE	32	ADAM 0.001	[64,32,16,8,4, 1]	64,3%	58,9%	65,8%	68,2%
7	7	128	FALSE	32	ADAM 0.001	[64,32,16,8,4, 1]	75,6%	49,2%	70,7%	69,8%
8	8	128	FALSE	16	ADAM 0.001	[64,32,16,8,4, 1]	78,0%	48,3%	71,4%	72,6%
6	6	128	FALSE	16	ADAM 0.001	[64,32,16,8,4, 1]	84,8%	37,7%	69,0%	70,0%

6	6	128	FALSE	16	ADAM 0.001	[128,64]	92,3%	25,2%	73,0%	72,9%

4.2 Benchmark

In this section we will present our results from the benchmark test where we run our NN model and compare it to the result we obtain when we run the original Altman Z-score on the same data set. Some clarification on how we have used the Z-score as a benchmark should be mentioned to help the reader to interpret the result. In Altman (1968) he found that his model performed very well in classifying companies with extreme Z-scores. Companies with Z-score below 1.8 were most likely heading for bankruptcy and companies with a Z-score above 3 were financially safe. However, companies with a Z-score between 1.8 and 3 were referred to as in a "gray area", the vast majority of the misclassifications were on companies with a Z-score between 1.8 and 3. Altman (1968) found the critical value to be 2.675 in his data set, that is, companies with a Z-score below 2.675 should be classified as bankrupt and companies with an Z-score above 2.675 should be classified as non-bankrupt. We have considered this when we have run the testing. Therefore, we run two Altman Z-score tests for each benchmark, one where we classify companies with a Z-score below 1.8 as bankrupt and one where we classify companies with a Z-score below 2.675 as bankrupt.

Lastly, one important difference from the original Altman Z-score compared to this test should be highlighted. The original Altman Z-score runs a separate test for each time horizon, that is, it runs five separate tests to predict bankruptcy 1 to 5 years prior to the occasion. However, our model run solely one test where the time horizon is set to an interval between 0-6 years,

4.2.1 Benchmark - Domestic

This test was designed to replicate the original Altman Z-score as closely as possible. Both our model and the Altman Z-score were tested on a same single country data set, consisting of a balance set of bankrupt and non-bankrupt manufacturing firms in the US. The criterion for this benchmark test was to determine if our NN model could outperform the original Altman Z-score in classifying bankrupt companies. Table 8 represents the outcome for bankruptcy predictions using the original Altman Z-score where companies with a Z-score below 2.675 was classified as bankrupt.

Table 8: Altman Z-score bankrupt < 2.675, domestic data set

Actual outcome	**Predicted Outcome**	
	Bankrupt	Non-Bankrupt
Bankrupt	19.48%	31.16%
Non-Bankrupt	20.53%	28.82%

Looking at the result in Table 7, we see that the total accuracy for bankruptcy prediction using the original Altman Z-score with a critical value of 2.675 is 48.3%. The type 1 error, when the model classifies a company as non-bankrupt, but the actual outcome is bankrupt, is 31.16%. The type 2 error, when the model classifies the model as bankrupt, but the actual outcome is non-bankrupt is 20.53%.

In the second test we again ran the original Altman Z-score but used the 1.8 as the critical value for the Z-score. That is, companies with a Z-score below 1.8 were classified as bankrupt, see the result in Table 9.

Table 9: Altman Z-score < 1.8, domestic data set

Actual outcome	**Predicted Outcome**	
	Bankrupt	Non-Bankrupt
Bankrupt	12.38%	38.85%
Non-Bankrupt	12.24%	36.53%

When the critical value is set to 1.8, the model performs an accuracy of 48.91%, that is an improvement of 0.61% compared to when the critical value is set to 2.675. The type 1 error is 38.85% and the type 2 error is 12.24%.

Lastly, we tested how our NN model performed in classifying bankrupt companies on the same data set. Table 10 shows the result from our NN model.

Table 10: Our NN model on a domestic data set

Actual outcome	**Predicted Outcome**	
	Bankrupt	Non-Bankrupt
Bankrupt	44.35%	6.80%
Non-Bankrupt	5.75%	43.10%

Our NN model performed an accuracy of 87.45% in classifying bankrupt companies on a domestic data set. This is a significant improvement compared to our benchmark of 48.3% and 48.9% respectively. The type 1 error in our NN model is 6.80%, this is a significant improvement compared to the best benchmark model which had a type 1 error of 31.16%. The type 2 error in our NN model was 5.75, even this is a significant improvement compared to our best benchmark model which had a type 2 error of 12.24%.

4.2.2 Benchmark - Global

In this section, we repeat the testing from the previous section, however, we are moving further away from the original Altman Z-score as we now are testing on a global data set consisting of a balanced data set of bankrupt and non-bankrupt manufacturing firms globally.

Table 11 shows the result from the original Altman Z-score in a global data set where the critical value was set to 2.675.

Table 11: Altman Z-score bankrupt < 2.675, global data set.

Actual outcome	**Predicted Outcome**	
	Bankrupt	Non-Bankrupt
Bankrupt	27.76%	22.24%
Non-Bankrupt	29.97%	20.03%

The total accuracy for bankruptcy prediction using the original Altman Z-score with a critical value of 2.675 is 47.79%. The type 1 error is 22.22% and the type 2 error is 29.97%.

In the second test we again ran the original Altman Z-score but used the 1.8 as the critical value for the Z-score, see the result in Table 12.

Table 12: Altman Z-score bankrupt < 1.8, global data set.

Actual outcome	**Predicted Outcome**	
	Bankrupt	Non-Bankrupt
Bankrupt	18.37%	31.63%
Non-Bankrupt	18.21%	31.79%

When the critical value is set to 1.8, the model performs an accuracy of 50.16%, that is an improvement of 2.37% compared to when the critical value is set to 2.675. The type 1 error is 31.63% and the type 2 error is 18.21%.

Lastly, we tested how our NN model performed in classifying bankrupt companies on the same global data set. Table 13 shows the result from our NN model.

Table 13: Our NN model on a global data set

Actual outcome	**Predicted Outcome**	
	Bankrupt	Non-Bankrupt
Bankrupt	38.22%	11.78%
Non-Bankrupt	15.27%	34.73%

Our NN model performed an accuracy of 72,95% in classifying bankrupt companies on a domestic data set. This is a significant improvement compared to our benchmark of 47.47% and 50.16% respectively. The type 1 error in our NN model is 11.78%, this is a significant improvement compared to the best benchmark model which had a type 1 error of 22.24%. The

type 2 error in our NN model was 15.27%, this is a slight improvement compared to our best benchmark model which had a type 2 error of 18.21%.

4.3 Domestic vs global data set

For the last test we compared how our NN model performed on the balanced domestic data set compared the balanced global data set. The results are shown in Table 14.

Table 14: Comparing domestic and global data set

NN Model	Domestic data set	Global data set
Correctly predicting bankrupt	44.35%	38.22%
Correctly predicting non-bankruptcy	43.10%	34.73%
Type 1 error	6.8%	11.78%
Type 2 error	5.75%	15.37%
Total Accuracy	**87.45%**	**72.95%**

The total accuracy drops from 87.45% to 72.95% when the global data set is used compared to the domestic. Furthermore, the NN model performs better accuracy in all four categories when a domestica data set is used.

5. Discussion

The result shows that a neural network and its architecture depends highly on the nature of the data. There are countless combinations of hypertuning parameters and different numbers of layers and neurons to choose between. The importance is to determine the best match of the specific dataset and the problem. Our objective was to predict a categorical variable, in this case bankruptcy using five ratios originally used by Altman (1968), on a global dataset to compare it with two different benchmarks. In the following section we discuss our findings in terms of the model itself as well as the findings compared to our original hypothesis.

5.1 The neural network

Our key findings when building, training and evaluating the model showed the following important configurations.

Batch Normalization - as a technique was not beneficial for our model since it is often used for much deeper neural networks to decrease the computational power and time needed to train. The complexity within our data is probably not that high since we only use five ratios and a relatively small number of lags. Since batch normalization takes each batch and normalizes the data specifically on these mini batches it only decreases the ability to find relationships between historical data within each batch. Our findings suggest that it is enough to standardize the whole dataset and remove batch normalization to increase the accuracy as well as the loss, specifically on our data.

Optimizer - By moving from a more standard choice of optimizer which usually is Stochastic Gradient Descent we saw that adaptive moment estimation (Adam) was a better fit for our model. This was a surprise since SGD usually performs better on SNN and Adam performs better on DNN. However, we did not get the chance to test SGD on a shallower network since this was our initial setup and could potentially be something to further investigate to see if the model could perform with even higher accuracy. However, ADAM is known to be a good fit thanks to its fast ability of convergence. We saw this convergence in the evolution of our network. The loss function did not move until ADAM was introduced and then started to move towards a minimum (local or global) with a decreasing trend which is the goal when training a neural network.

Shallow Neural Network **-** We found that a shallow neural network performed better than a deep neural network. This finding does not contradict our first hypothesis that an intelligent model like the neural network loosen some of the assumptions from the MDA. However, it showed that in this case that a deep neural network was not the best suit for this problem and data but rather a shallow one. As mentioned before, this doesn't mean that there does not exist a setup with more layers that could perform with higher accuracy, but in our scope and limitation of the study, that was our findings.

5.2 Benchmark domestic

One of the objectives with this study was to address whether a reassessed Altman Z-score using a NN approach could improve the accuracy of bankruptcy prediction. We choose to reassess the Altman Z-score as it is one of the most famous models for predicting financial and stated the following hypothesis:

Hypothesis 1 (H1a): A deep neural network model yields higher accuracy for bankruptcy predictions than the Altman Z-score with in a six-year time horizon

In our first test where we tested on a domestic data set, the Altman Z-score yielded an accuracy of 48.3% when the critical Z-score was set to 2.675 and 48.91% when the critical Z-score was set to 1.8. On the same data set, our NN model yielded an accuracy of 87.45%. See Table 15.

Table 15: Comparing the result of Altman Z-score vs NN on a domestic data set

Variables	**Altman Z-score < 2.675, Domestic**	**Altman Z-score <1.8, Domestic**	**NN model**
Correctly predicting bankrupt	19.48%	12.38%	44.35%
Correctly predicting non-bankruptcy	28.82%	36.53%	43.10%
Type 1 error	31.16%	38.85%	6.8%
Type 2 error	20.53%	12.24%	5.75%
Total Accuracy	**48.3%**	**48.91%**	**87.45%**

Altman (1968) suggested that the critical value, in his original data set, is a Z-score below 2.675. However, he acknowledges that Z-score between 1.8 and 3 is somewhat of a grey area and this is where his models make most of its misclassification. Yet, he claims that almost all companies with a Z-score below 1.8 went bankrupt. To avoid any criticism regarding the choice of Z-score, we have included both the 1.8 Z-score and the 2.675 Z-score as critical values in our benchmark model. Obviously, the Altman Z-score with a critical value of 1.8 yields a higher type 1 error, that is predicting non-bankruptcy when the actual outcome is bankruptcy, then the Altman Z-score with a critical value of 2.675. When the critical value is set to 2.675, this relationship is reversed. Then the type 2 errors increase, that is predicting non-bankrupt when the actual outcome is bankrupt. However, what's more important is that irrespective of the critical value, our NN model outperforms the Altman Z-score on all four categories. This finding supports our hypothesis that a NN model improves the accuracy of Altman Z-score.

5.3 Benchmark global

A global data set introduces more heterogeneity in the data set which will complicate the pattern recognition, however, by utilizing a global data set we increase the data points. We hypothesized that a NN model would benefit more from the increased data point compared to an MDA model and stated the following second hypothesis.

Hypothesis 2a (H1a). *The discrepancy in accuracy between a NN model and the Altman Z-score increases when a global data set is used.*

To test this hypothesis, we ran the exact same test as in 6.1 but with the one difference that we ran this test on a global data set. We validated the result by comparing the difference in accuracy between our NN model and the Altman Z-score on a domestic data set against the difference on a global data set. See result in Table 16.

Table 16: Comparing the result of Altman Z-score vs NN on a global data set

Variables	**Altman Z-score < 2.675, Global**	**Altman Z-score < 1.8, Global**	**NN model**
Correctly predicting bankrupt	27.76%	18.37%	38.22%
Correctly predicting non-bankruptcy	20.03%	31.79%	34.73%
Type 1 error	22.24%	31.63%	11.78%
Type 2 error	29.97%	18.21%	15.37%
Total Accuracy	**47.79%**	**50.16%**	**72.95%**

Comparing the result in Table 16 to the result in Table 15, we see that the accuracy of the original Altman Z-score is approximately the same whether we use a domestic or a global data set. The accuracy yielded with a critical value of 1.8 decreases from 48.3% to 47.79% and the accuracy yielded with a critical value of 2.675 increases from 48.91% to 50.16%. However, the accuracy of the NN model drops from 87.45% on a domestic data set to 72.95% on a global data set. This resulted in a decrease in discrepancy in terms of accuracy between the original Altman Z-score and our NN model. The discrepancy between the original Altman Z-score and our NN model was 38.54% when the critical value was 1.8 and 39.15% when the critical value was 2.675 for a domestic data set. The discrepancy decreased to 22.79% when the critical value was 1.8 and 25.16% when the critical value was 2.675 on a global data set, see Table 16.

Table 17: Comparing the discrepancy in accuracy between domestic and global data set

Models	Accuracy	Discrepancy vs NN
Z-score < 2.675, Domestic	48.3%	39.15%
Z-score < 1.8, Domestic	48.91%	38.54%
NN, Domestic	87.45%	
Z-score <2.675, Global	47.79%	25.16%
Z-score <1.8, Global	50.16%	22.79%
NN, Global	72.95%	

This result suggests a reverse relationship to the one we hypothesized. The discrepancy between the Altman Z-score and our NN model decreases as we use a global data set rather than a domestic one. As the result shows, the Altman Z-score is only marginally affected when a global data set is used, however, the accuracy of the NN model drops significantly on a global data set compared to the domestic. This is not what we expected to find, however, the data implies that the additional heterogeneity imposed using a global data set has more severe consequences for the NN model than the Altman Z-score. One plausible reason for this could be that the NN model relies heavily on pattern recognition, and when additional heterogeneity is imposed using a global data set it affects the NN model more than the Altman Z-score.

5.4 Domestic vs Global data set

The final objective of this study was to test whether the NN model could perform at least the same accuracy on a global data set as on a domestic one. We hypothesized that the NN model would benefit enough from the increased data points in a global data set to compensate for the additional heterogeneity and we stated the following hypothesis.

Hypothesis 2b (H2b). *A deep neural network model will yield at least the same accuracy on a global data set as on a domestic data set.*

The result for this test can be seen in table 14. The result suggests that the NN model with a domestic data set yields better accuracy in all four categories than a NN model with a global

data set. This drop in accuracy indicates that the increased data points is not enough to compensate for the additional heterogeneity. However, there is one obvious limitation when applying our NN model to a global data set. Our NN model is a reassessment of the original Altman Z-score where the variables were chosen for a domestic data set. Thus, it is reasonable to assume that our models suffer from some omitted variable bias. Therefore, this should not be treated as evidence that NN models in general can't perform the same accuracy on global data sets, rather, it shows that the assessed Altman Z-score don't perform as well on a global data set as on a domestic one. It is plausible that NN models could utilize the increased data points better if the models used additional variables to match the global data set. Then, the NN models might as well perform at least the same accuracy on a global data set that on a domestic one.

6. Conclusion

There were three main objectives with this thesis. The first was to create a model with as high accuracy as possible on our given dataset based on several finetuning of parameters. The result showed high accuracy on both the US market and on the global dataset. Previous research has performed an out-of-sample accuracy between 82-97%. However, these models utilized several additional variables for predicting bankruptcy and, more importantly, a time horizon of only 1-2 years, which is too close to the bankruptcy to act as a warning signal. Thus, based on the recurrent neural network in this study, we can see that lagged variables of Altman's original ratios yields high predictive power on bankruptcy for up to five years prior to the bankruptcy, this indicates that warning signals could be detected at an earlier stage using NN. Suggestions for further research is to use RNN with LSTM cells but on an even wider range of financial and non-financial ratios to incorporate the relationship between previous time periods to predict on a longer horizon.

The second objective was to reassess the Altman Z-score using a NN approach to improve FDP within a 6-year time horizon. Our NN model managed to outperform the Altman Z-score in all four categories that were measured, and it significantly improved the total accuracy. This finding strengthens the theory that intelligence models, like NN, can outperform statistical models by loosening some of the assumptions that must hold true in statistical models. While the NN model outperformed the MDA on all categories, the high accuracy of our NN model

suggests that the ratios utilized in the Altman Z-score are still valid predictors of financial distress.

The last objective was to compare the result of our NN model on a domestic data set compared to a global one. The result shows that the accuracy drops significantly when a global data set is used compared to a domestic. This finding weakens our hypothesis that the increased number of data points in a global data set will compensate for the additional heterogeneity. However, the variables used in this study were originally chosen for a domestic data set, and it is plausible that additional variables are needed to capture the variety in a global data set. Therefore, we do not reject the hypothesis completely, rather, we encourage future research to utilize the NN with LSTM on a global data set but including additional variables to control the heterogeneity in a global data set.

The model presented in this thesis showed promising results for both domestic and global data sets. However, one limitation of this study should be mentioned. The OOS accuracy, compared to previous literature, is high and could potentially indicate that there exist some problems that should be mentioned with respect to the model trustworthiness. First, we believe that our test data is, due to preprocessing and data cleaning, to small and could be an unrepresentative data sample. This could have been addressed by designing a robust test harness before getting started with evaluating the model's accuracy. Secondly, this study utilizes an interval ranging from 1 to 6 years as the time horizon, and the companies have been chosen randomly without replacement. Therefore, it is possible that our data set consist of an overweight of companies that are close to bankruptcy compared to companies that are 5-6 years from bankruptcy. This will naturally improve the accuracy of our model. This issue could be solved by using a more controlled data set with equally weighted numbers of different number of time horizons, to make sure that the number of years prior to bankruptcy is evenly spread through the data set. However, this will lower the number of observations in the data set and this method was there for rejected in this study. Yet, we suggest that future research should balance the data set to see how this affect the accuracy of the model and design a robust test harness.

7. References

Ahmed, N. Amir, A. Hisham El-Shishiny. Neamat El Gayar (2010) An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econometric Reviews Volume 29, 2010 - Issue 5-6*:

Altman, E. L. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The Journal of Finance, 23(3), 589–609.*

Altman, Edward, Haldeman and Narayanan (1977). A new model to identify bankruptcy risk of corporations. *Journal of Banking and Finance, 1 (1977), pp. 29-54.*

Altman, Edward I. Małgorzata Iwanicz-Drozdowska, Erkki K. Laitinen & Arto Suvas (2020) A Race for Long Horizon Bankruptcy Prediction, *Applied Economics, 52:37, 4092-4111*

Bahrammirzaee, A. (2010) A Comparative Survey of Artificial Intelligence Applications in Finance: Artificial Neural Networks, Expert Systems and Hybrid Intelligent Systems. *Neural Computing and Applications volume 19, 1165–1195*

Beaver, W. H. (1966). Financial Ratios As Predictors of Failure. *Journal of Accounting Research, 4,* 71–111.

Blum, M. (1974) FAILING COMPANY DISCRIMINANT-ANALYSIS. *Journal of Accounting Research, 1974, vol. 12, issue 1, 1-25.*

Büyüköztürk, S. Çokluk, Ö. (2008) Discriminant Function Analysis: *Concept and Application. Eurasian Journal of Educational Research, 33, pp 73/92 2008.*

Chen, Wei-Sen, and Yin-Kuan Du. 2009. Using neural networks and data mining techniques for the financial distress prediction model. *Expert Systems with Applications 36: 4075–86.*

Cukier, K. (2010) Data, Data Everywhere: A Special Report on Managing Information. *The Economist, 394, 3-5.*

Dimitras,A.I. Zanakis, S.H. & Zopounidis, C. (1996) A survey of business failures with an emphasis on prediction methods and industrial applications. *European Journal of Operational Research, Volume 90, Issue 3, 10 May 1996, Pages 487-513.*

Fisher, R.: The use of multiple measurements in taxonomic problems. *Annals of Eugenics 7(7), 179–188 (1936)*

Fletcher, D., & Goss, E. (1993). Forecasting with neural networks: An application using bankruptcy data. *Information & Management, 24(3), 159-167.*

Gregova, Elena, Katarina Valaskova, Peter Adamko, Milos Tumpach, and Jaroslav Jaros. 2020. *Predicting financial distress of slovak enterprises: Comparison of selected traditional and learning algorithms methods. Sustainability 12: 3954.*

Huberty, C, (1975) Discriminant Analysis. Review of Educational Research, *Vol. 45, No.4, Pp. 543-598.*

IBM. (2020). Neural Networks. Retrieved from IBM Cloud Education. Received from: https://www.ibm.com/cloud/learn/neural-networks

Jie Sun, Hui Li, Qing-Hua Huang, Kai-Yu He. (2012). Predicting financial distress and corporate failure: A review from the state-of-the-art definitions, modeling, sampling, and featuring approaches. *School of Economics and Management, Zhejiang Normal University, Jinhua, Zhejiang 321004, PR China*

Min JH, Jeong C (2009) A binary classification method for bankruptcy prediction. *Expert Syst Appl 36: 5256–5263*

M. Leshno, Y. Spector, Neural network prediction analysis: The bankruptcy Case. *Neurocomputing 10 (1996) 125–147.*

Murphy, K. (2012) Machine Learning A Probabilistic Perspective

Odom. M,Sharda. M (1990) A neural networks model for bankruptcy prediction. *Proceedings of the IEEE International Conference on Neural Network, 2, pp.* 163-16

Ohlson, J. A. (1980). Financial Ratios and the Probabilistic Prediction of Bankruptcy. *Journal of Accounting Research*, *18*(1), 109–131.

Paul A Meyer and Howard W Pifer, (1970), Prediction of Bank Failures, Journal of Finance, 25, (4), 853-68

Paule-Vianez, Jessica, Milagros Gutiérrez-Fernández, and José Luis Coca-Pérez. 2020. Prediction of financial distress in the Spanish banking system: *An application using artificial neural networks. Applied Economic Analysis 28: 69–87.*

Wilson, R.L. and Sharda, R. (1994) Bankruptcy Prediction Using Neural Networks. *Decision Support Systems, 11, 545-557.*

Zhang, Guoqiang, Michael Y. Hu, B. Eddy Patuwo, and Daniel C. Indro. 1999. Artificial neural networks in bankruptcy prediction: *General framework and cross-validation analysis. European Journal of Operational Research 116: 16–32.*

8. Appendix

8.1 The Neural Network Code

```
# prepare data for lstm
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
# for modeling
from keras.layers.recurrent import LSTM
from keras.models import Sequential
from keras.layers import Dense, Dropout, BatchNormalization, Embedding
from keras.callbacks import EarlyStopping
from IPython.display import clear_output, display
import random
import math
# Import the lib
from ibm_watson_studio_lib import access_project_or_space

import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
client_6330849a6f2a43348552861bc970243f = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='YqNr3mlm9C7uGLbvtfXvDhqaVTZRsdKYuM4rtlzFQLhz',
```

```
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

body = client_6330849a6f2a43348552861bc970243f.get_object(Bucket='masterml-
donotdelete-pr-yvuffkgfgimyba',Key='data.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

data = pd.read_csv(body)
```

In [ ]:

```
def create_variables(prediction_horizon, data):
    #creating the distress column with 1 as bancrupt.
    df = data.copy()
    arr_distress = [2, 3, 6]
    df.loc[~df['dlrsn'].isin(arr_distress), 'deleted'] = 0
    df.loc[df['dlrsn'].isin(arr_distress), 'deleted'] = 1
    df.drop('dlrsn', axis=1, inplace=True)

    #creating the year column of bancruptcy year.
    df["dldte"] = pd.to_datetime(df['dldte'], format='%Y%m%d')
    df['dldte'] =  pd.to_datetime(df['dldte']).dt.year
    df["dldte"] = df["dldte"].fillna(2122)

    #creating the year to bancruptcy column
    df["year_to_b"] = df["dldte"] - df["fyear"]
    #df = df[df['year_to_b'] >= prediction_horizon] #Removes all the rows that went
bancrupt in more recent year than prediction_horizon.
    df.loc[df['year_to_b'] <= prediction_horizon, 'y'] = 1 #
    df.loc[df['year_to_b'] > prediction_horizon, 'y'] = 0

    #Calculating all the variables and doing some winsorazation
    df["A"] = df["wcap"]/df["at"] #Altmans A working capital
```

```
df["B"] = df["re"]/df["at"] #Altmans B retained Earnings/Total Assets (RE/TA)
df["C"] = df["ebit"]/df["at"] # Earnings before interest / Total Assets
df["D"] = (df["at"] - df["lt"])/df["lt"] #vi saknar book value / total liabilities
df["E"] = df["sale"]/df["at"] #

#remove Nan after calculations
df = df[~df.isin([np.nan, np.inf, -np.inf]).any(1)]

#after the winsorazation calculate the Z-score of each statement
df["Z"] = (df["A"]*1.2+df["B"]*1.4+df["C"]*3.3+df["D"]*0.6+df["E"]) #altmans z-score

#Winsorazation
for i in ["A","B","C","D","E"]:
    low = df[i].quantile(0.05)
    high = df[i].quantile(0.95)
    df[i] = np.where(df[i] < low , low ,df[i])
    df[i] = np.where(df[i] > high, high ,df[i])

#scale all the to take values between -1 and 1.
scaler = MinMaxScaler(feature_range=(0, 1))
normalised_data = scaler.fit_transform(df[["A", "B","C","D","E"]])
df[["A", "B","C","D","E"]] = normalised_data

# encode class values as integers
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
encoder.fit(df.y)
encoded_Y = encoder.transform(df.y)
df.y = encoded_Y

sic_list = [20,21,22,23,24,25,26,27,30,31,32,33,34]
#fic_list = ["USA"]
df['sic_new'] = df.sic.astype(str).str[:2]
df['sic_new'] = pd.to_numeric(df['sic_new'])
```

```
    df = df[df['sic_new'].isin(sic_list)]
    #df = df[df['fic'].isin(fic_list)]
    print(df[["A","B","C","D","E","Z","y"]].describe())

    #create list of
    dft = df[df.y == 1].copy()
    bancrupt_list = dft.gvkey.unique()
    df_b = df[df.gvkey.isin(bancrupt_list)]
    df_a = df[~df.gvkey.isin(bancrupt_list)]
    print(df_b.y.describe())
    return df_a, df_b
```

In [ ]:

```
def lstm_data_preparation(a, b, split, num_steps):
    """ Changes data to the format for LSTM training for sliding window approach and
splitting the data between test and train"""
    # Prepare the list for the transformed data
    x_train_list, y_train_list, x_test_list, y_test_list, y_test_info = list(), list(), list(), list(), list()

    count = 1
    print(len(b.gvkey.unique()))
    #tFill the lists with bancrupt-firms data
    for x in b.gvkey.unique():
        df_temp = b[b.gvkey == x]
        x_data, y_data, y_info = df_temp[["A", "B","C","D","E"]], df_temp['y'],
df_temp[["fic","Z","year_to_b"]]
        for i in range(0,x_data.shape[0],num_steps):
            # compute a new (sliding window) index
            end_ix = i + num_steps

            # if index is larger than the size of the dataset, we stop
            if end_ix >= x_data.shape[0]:
                break
```

```
        # Get a sequence of data for x
        seq_X = x_data[i+1:end_ix+1]

        # Get only the last element of the sequency for y
        seq_y = y_data[end_ix:end_ix+1]
        # Get only the last element of the sequency for y_info
        seq_i = y_info[end_ix:end_ix+1]

        # Append the list with sequencies
        if seq_y.values == 1:
            if count < split:
                x_train_list.append(seq_X)
                y_train_list.append(seq_y)
                count = count + 1
            elif count == split:
                x_test_list.append(seq_X)
                y_test_list.append(seq_y)
                y_test_info.append(seq_i)
                count = 1

    #now we will fill the lists with non-bancrupt data
    count = 1
    fill = len(y_test_list)+len(y_train_list)
    filled = 0
    print(fill)
    list_a = np.random.choice(a.gvkey.unique(),size=fill*2) #randomly picks non-bancrupt
firms with n = no. of bancrupt firms * 2
    print(len(list_a))
    #loop through non-bancrupt firms and add these values
    for z in list_a:
        #clear_output(wait=True)
        df_temp = a[a.gvkey == z]
        x_data, y_data, y_info = df_temp[["A", "B","C","D","E"]], df_temp['y'],
df_temp[["fic","Z","year_to_b"]]
```

```
        #get one random year sample of the firm

        i = random.randint(0, x_data.shape[0])
        end_ix = i + num_steps

         # if index is larger than the size of the dataset, we stop
        if end_ix < x_data.shape[0]:
            if filled <= fill:
                filled = filled + 1
                # Get a sequence of data for x
                seq_X = x_data[i+1:end_ix+1]

                # Get only the last element of the sequency for y
                seq_y = y_data[end_ix:end_ix+1]
                seq_i = y_info[end_ix:end_ix+1]

                if count < split:
                    x_train_list.append(seq_X)
                    y_train_list.append(seq_y)
                    count = count + 1
                elif count == split:
                    x_test_list.append(seq_X)
                    y_test_list.append(seq_y)
                    y_test_info.append(seq_i)
                    count = 1

# Make final arrays
x_train_array = np.array(x_train_list)
y_train_array = np.array(y_train_list)
x_test_array = np.array(x_test_list)
y_test_array = np.array(y_test_list)
y_info_array = np.array(y_test_info)

#Shuffeling
```

```
    array1 = x_train_array
    array2 = y_train_array
    array3 = x_test_array
    array4 = y_test_array
    array5 = y_info_array

    shuffler1 = np.random.permutation(len(array1))
    x_train_array = array1[shuffler1]
    y_train_array = array2[shuffler1]

    shuffler2 = np.random.permutation(len(array3))
    x_test_array = array3[shuffler2]
    y_test_array = array4[shuffler2]
    y_test_info = array5[shuffler2]

    return x_train_array, y_train_array, x_test_array, y_test_array, y_test_info
```

In [ ]:

```
def create_model(dropout):
    model = Sequential()
    model.add(LSTM(128, activation='relu', input_shape=(num_steps+1, num_features),
return_sequences=False))
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(dropout))
    model.add(Dense(1, activation='sigmoid'))

    from tensorflow.keras.optimizers import Adam
    opt = Adam(learning_rate=0.001)

    # compile the model
    model.compile(optimizer = opt,
                  loss='binary_crossentropy',
                    metrics=['accuracy'])
```

```
    es = EarlyStopping(monitor='val_accuracy',
                       mode='max', # don't minimize the accuracy!
                       patience=25,
                       restore_best_weights=True)
    # now we just update our model fit call
    history = model.fit(x_train,
                y_train,
                callbacks=[es], #
                epochs=50, # you can set this to a big number!
                batch_size=16,
                validation_split=0.1,
                shuffle=True,
                verbose=1)
    return model, history
```

In [ ]:

```
#data = pd.read_csv('data.csv')

#create assumptions.
horizon = 6
num_steps = 6
num_features = 5
dropout = 0.2

df_a, df_b = create_variables(horizon, data)
x_train, y_train, x_test, y_test, y_test_i = lstm_data_preparation(df_a, df_b, split=5,
num_steps=num_steps+1) #split = 5 means that it will take 4 companies to train, and 1 to
test i.e, approx 80% vs 20% split.
print(len(x_train),len(x_test), len(y_test_i))
ann, ann_history = create_model(dropout)
y_hat = ann.predict(x_test)
```

In [ ]:

```
rmse = math.sqrt(mean_squared_error(y_test, y_hat))
mse = mean_squared_error(y_test, y_hat)
```

```
print(rmse, mse)
```

In [ ]:

```
df = pd.DataFrame(columns = ['Correct/Wrong', 'Z-score',"bancrupt","bancrupt_y",
"fic","ANN"])
country = 0
score = 1
years = 2

# append rows to an empty DataFrame
for i in range(0,len(y_test)):
    #TYPE I and II
    if (int(round(y_hat[i][0], 0)) == 0) & (y_test[i] == 1):
        ANN = 1
    elif (int(round(y_hat[i][0], 0)) == 1) & (y_test[i] == 1):
        ANN = 2
    elif (int(round(y_hat[i][0], 0)) == 0) & (y_test[i] == 0):
        ANN = 3
    elif (int(round(y_hat[i][0], 0)) == 1) & (y_test[i] == 0):
        ANN = 4

    if (y_test_i[i][0][score] > 2.675) & (y_test[i][0] == 1): #om företaget har en score < 1.8
och går i konka
        df = df.append({'Correct/Wrong' : 1, 'Z-score' : y_test_i[i][0][score] , 'bancrupt' :
y_test[i][0], 'bancrupt_y' : y_test_i[i][0][years], "fic" : y_test_i[i][0][country], "ANN":
ANN}, ignore_index = True)
    elif (y_test_i[i][0][score] < 2.675) & (y_test[i][0] == 1):
        df = df.append({'Correct/Wrong' : 2, 'Z-score' : y_test_i[i][0][score] , 'bancrupt' :
y_test[i][0], 'bancrupt_y' : y_test_i[i][0][years], "fic" : y_test_i[i][0][country],"ANN": ANN},
ignore_index = True)
    elif (y_test_i[i][0][score] > 2.675) & (y_test[i][0] == 0):
        df = df.append({'Correct/Wrong' : 3, 'Z-score' : y_test_i[i][0][score] , 'bancrupt' :
y_test[i][0], 'bancrupt_y' : y_test_i[i][0][years], "fic" : y_test_i[i][0][country],"ANN": ANN},
ignore_index = True)
```

```
    elif (y_test_i[i][0][score] < 2.675) & (y_test[i][0] == 0):
        df = df.append({'Correct/Wrong' : 4, 'Z-score' : y_test_i[i][0][score] , 'bancrupt' :
y_test[i][0], 'bancrupt_y' : y_test_i[i][0][years], "fic" : y_test_i[i][0][country],"ANN": ANN},
ignore_index = True)
```

In [ ]:

```
print("Altmans test:")
print("Type I:", round(df["Correct/Wrong"].value_counts()[1]/len(df)*100,2),"%")
print("Type II:", round(df["Correct/Wrong"].value_counts()[4]/len(df)*100,2),"%")
print("bancrupt - guessed bancrupt",
round(df["Correct/Wrong"].value_counts()[2]/len(df)*100,2),"%")
print("non bancrupt - guessed non-bancrupt:",
round(df["Correct/Wrong"].value_counts()[3]/len(df)*100,2),"%")
```

In [ ]:

```
print("Type I:", round(df.ANN.value_counts()[1]/len(df)*100,2),"%")
print("Type II:", round(df.ANN.value_counts()[4]/len(df)*100,2),"%")
print("bancrupt - guessed bancrupt", round(df.ANN.value_counts()[2]/len(df)*100,2),"%")
print("non bancrupt - guessed non-bancrupt:",
round(df.ANN.value_counts()[3]/len(df)*100,2),"%")
```