Bankruptcy Prediction by Deep Learning and Machine Learning Methods

Parisa Zahiri

**A thesis
in the Department of
Mechanical, Industrial and Aerospace Engineering**

**Presented in Partial Fulfillment of the Requirements
for the Degree of
Master of Applied Science (Industrial Engineering)**

**at Concordia University
Montreal, Quebec, Canada**

November 2022

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By: **Parisa Zahiri**

Entitled: **Bankruptcy Prediction by Deep Learning and Machine Learning Methods**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Industrial Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

____Chair
Dr. Hossein Hashemi Doulabi

____Examiner
Dr. Suchit Ahuja

____Examiner
Dr. Hossein Hashemi Doulabi

____ Thesis Supervisor(s)
Dr. Masoumeh Kazemi Zanjani

____Thesis Supervisor(s)
Dr. Salim Lahmiri

Approved by ____
Dr. Martin D. Pugh Chair of Department or Graduate Program Director

____

Dr. Mourad Debbabi Dean of Faculty

November 1, 2022

Abstract

Bankruptcy Prediction by Deep Learning and Machine Learning Methods

Parisa Zahiri

Bankruptcy prediction plays a crucial role in today's businesses to survive in a competitive world. For avoiding the risk of bankruptcy, researchers have conducted significant research in field of artificial intelligence for predicting bankruptcy. However, the performance of deep learning methods is not well understood. To address this research gap, we make the following main contributions: We applied deep learning methods into Polish datasets in addition to traditional machine learning techniques. We applied several versions of convolutional neural networks and artificial neural networks to several datasets created from the available dataset. Specifically, we created 5 extra datasets for each year in addition to the entire datasets for five years. We incorporated some techniques to balance the datasets and measured the impacts these techniques have on performance measures. This step is important because the datasets are imbalanced, i.e., the proportion of firms experiencing bankruptcy is much lower than those who did not go bankrupt. For deep learning techniques, we also explored preprocessing approaches and measured their impacts on results. Specifically, we used validation on the same datasets of studies in the literature and compared our results with those available in the literature with the same test bed. Our results shed light on the impact of preprocessing and balancing techniques in deep learning, as well as different architectures for deep learning methods. We observed improvement, compared to the literature, in terms of accuracy and provided insights on the value of different deep learning architectures and preprocessing on the sensitivity of the results.

Acknowledgments

I am extremely thankful to my supervisors Dr. Masoumeh Kazemi Zanjani from the Department of Mechanical, Industrial and Aerospace Engineering (MIAE) and Dr. Salim Lahmiri from the Department of Supply Chain and Business Technology Management at the John Molson School of Business for their noble supervision, support with full encouragement and enthusiasm, valuable suggestions, and motivating advice. This thesis would not have been possible without your trust in me and your guidance from the initial step in research that allowed me to develop an understanding of the subject.

I wish my father was alive to be proud of me like he always was. I am grateful for my mother whose constant love and support keep me motivated and confident. My accomplishments and success are because she believed in me. Also, I owe my deepest gratitude to my husband, Amin. I am forever thankful for the unconditional love and support throughout the entire thesis process and every day. I am again thankful from my brother, who is more than a brother for me, and my sister with her support every time I was upset.

Dedication

I dedicate this research project to truly brave Iranian women!

Woman Life Freedom

Contents

List of Figures

List of Tables

List of Abbreviations

The following abbreviations are used in this manuscript:

DL	Deep Learning
ML	Machine Learning
LR	Logistic Regression
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DBNN	Deep Belief Neural Network
KNN	K-Nearest Neighbor
SVM	Support Vector Machine
GB	Gradient Boosting
GBDT	Gradient-boosted decision trees
LightGBM	Light Gradient Boosting Machine
CNN_1D	Convolutional Neural Network One _Dimensional
CNN_2D	Convolutional Neural Network Two _Dimensional
CNN_2D_ResidualNetwork	Convolutional Neural Network Two_Dimensional Residual Network
SMOTE	Synthetic Minority Oversampling Technique

1. Introduction

This research seeks to provide a flexible framework to address essential questions regarding bankruptcy prediction. In the competitive world in which we live in, being aware of bankruptcy laws is very essential for both investors and shareholders in difficult financial conditions. In 1542, England was the first country in which the bankruptcy laws were established. Although since then the bankruptcy laws have changed in many countries, its main concept and motivation is intact. In fact, the business or individuals who cannot refund their great debts declare bankruptcy to receive protection for paying back their debts. With these bankruptcy laws in place, it is of utmost importance that the companies would be able to predict their chance of bankruptcy in the future to be able to come up with different plans. One main ingredient to this end is the ability to predict their financial situation and specifically the chances of bankruptcy (Altman and Hotchkiss, 2010).

Predicting bankruptcy has a significant impact on the market. Using bankruptcy prediction techniques may make markets more sustainable as it avoids future financial crisis. Also, one of the reasons for bankruptcy prediction goes back to the feeling of responsibility among auditors, creditors and stakeholders towards the future of their business. By doing so, the government and investment managers who lead financial institutions, banking business, manufacturing industry, etc. will be able to decide on what actions to take to save their companies from failure conditions and economic loss. Thus, the importance of predicting bankruptcy is increasing day by day, and to this end, the researchers have suggested several financial indicators for bankruptcy prediction, such as net revenue, net profit, liabilities, etc. Regardless of the advantages of prediction of bankruptcy, it is important to note that the cost of misclassification of predicting bankruptcy could be higher than the bankruptcy cost. While it is true that filing for bankruptcy lets companies to start their businesses again, but it may have a disastrous impact on their businesses in the future as these companies would not be able receive loans from the banking system anymore (Thorgren and Williams, 2020). Therefore, the researchers have made significant effort to predict the bankruptcy with the least error probability.

Inspired from the research conducted by Zięba et al. (2016) the main goal of this thesis is to develop new models, relying on deep learning (DL) approaches, in order to forecast companies' bankruptcy and validate them via Polish companies data sets, that is produced from Emerging Markets Information Service (EMIS) in the manufacturing sector. These data sets include 64 quantitative features and are categorized into a binary classification including still operating and bankrupt companies with a detailed explanation about their features, which we will explain later in this thesis. We chose this data set because many firms, relative to other countries and time periods, went bankrupt in the manufacturing sector since 2004 in Poland, making it a rich data set for analysis. Also, this comprehensive data set includes many financial features for a long period of time according to Zięba et al.(2016). Our second goal is to compare the results obtained from DL models with the results of traditional machine learning (ML) methods, such as logistic regression by considering normalized and non-normalized input data.

The above-mentioned goals can be further cascaded into the following objectives:

1) To use data preprocessing to clean up the raw data. Data preprocessing includes the detection of missing and imbalanced data, and normalization to increase the predictive power of models.

2) To apply ANN on the testing data set for the forecasting period of one year.

3) To explore both accuracy and sensitivity of ANN models after tunning with different hyperparameters.

4) To explore and investigate several DL approaches, namely, CNN models on five Polish data sets (for each year) and the entire five-year data set. Within the framework of CNN models, we also aim to explore various data preprocessing approaches to understand which of them improves the CNN models' performances.

5) To apply Logistic Regression on the annual and entire five-year data sets.

6) To compare the results of DL models with ML models in terms of various performance metrics by considering normalized and non-normalized data

7) To greatly improve the accurate rate of detection of risky companies.

8) To compare our results with those obtained in previous works used on the same database.

The rest of this thesis is structured as follows: Chapter two discusses similar studies on bankruptcy prediction that rely on statistics, economics and ML approaches in addition to similar research studies conducted on Polish data set, and also thesis contributions are summarized in this chapter. The third chapter describes the methodology adopted to achieve the objectives of this research. Chapter fourth discusses different data sets and the preprocessing approaches implemented on unbalanced data sets. The results of applying all DL and ML models on Polish data sets are provided in chapter five. Chapter six analyzes the outcomes and compares the results obtained from different prediction models. Chapter seven summarizes the concluding remarks and future research directions.

2. Literature Review

2.1. Bankruptcy Prediction

The bankruptcy prediction has progressed in late 1960s by the aid of statistical methods. The research conducted by Beaver (1966) and Altman (1968) are recognized as seminal studies in this field. In particular, Beaver (1966) and Altman (1968) used, respectively, Univariate Analysis and Multivariate Discriminant Analysis (MDA) to predict bankruptcy. Initializing generalized linear models by Ohlson (1980) in credit scoring area provided a number of benefits including the ability to assess the certainty of predictions and examine the impact of every indicator independently. Bellovary et al.(2007) reviewed various bankruptcy prediction approaches published over a long period of time, and compared their results. One of the first studies cited in Bellovary et al.(2007) was the research conducted by Merwin (1942) that introduced the importance of financial indicators in predicting bankruptcy.

Employing ML algorithms in the field of risk assessment, such as financial distress prediction, has increased since 1990s (Atiya, 2001). Some successful ML methods, including Support Vector Machine (SVM) (Shin et al., 2005), Extreme Gradient Boosting (EXGboost) (Chen et al., 2015), Bagging, Boosting and Random Forest (RF) (Barboza et al., 2017) were used to predict bankruptcy. Furthermore, a strategy relying on neural networks (NNs), trained with back-propagation and focusing on automatic feature extraction on the data, have been widely utilized in several studies (e.g., (Zhang et al., 1999) and (Tsai and Wu, 2008)). In fact, researchers are inclined to use these NN methods more than traditional ML models. For instance, Shah and Murtaza (2000) achieved a high prediction accuracy in the context of 60 firms including 6 bankrupts and 54 non-bankrupts by using NN method. Also, ensemble classifier has been received attention by Alfaro et al. (2008) in the bankruptcy prediction field.

Recently, significant scientific work in the field of bankruptcy prediction has emerged. For instance, Barboza et al. (2017) used new and standard ML models and introduced new features such as operating margin, change in return-on-equity, change in price-to-book, growth measures related to assets, sales, and number of employees, as predictive variables. As the result of this research, the authors claimed that the bagging, boosting, and random forest models perform better than other standard models available in the literature and adding the above-mentioned features would improve the accuracy of the model.

In another research, Liang et al. (2016) claimed to be among the first, combining different categories of financial ratios (FRs) and corporate governance indicators (CGIs) for bankruptcy prediction. The take home message of that paper is that combining the input features can help the accuracy but not in a significant way.

Mai et al. (2019) in another research introduced DL models for bankruptcy prediction using both structured (accounting-based and market-based) and unstructured Managerial Discussion and Analysis (MD&A) from 10-K filings inputs with 36 predictor variables, and compared the accuracy ratio of each DL model.

Matin et al. (2019) proposed a hybrid model by combining a Convolutional Neural Network (CNN) model and numerical variable. The authors used 3 different classification methods to classify the extracted features, including Neural Networks, Gradient boosted trees, and Logistic Regression. They also demonstrated that extracting patterns in the annual report text can improve the accuracy of the prediction model.

The study of Zhao et al. (2017), applied Kernel Extreme Learning Machine (KELM), which is designed for increasing the robustness of Extreme Learning Machine (ELM), to predict bankruptcy. The authors concluded that the proposed model beats other methods such as support vector machines, extreme learning machine, random forest, fuzzy k-nearest neighbor method (based on particle swarm optimization) and the Logit model.

Ding et al. (2015) used deep CNNs with 4 layers on real data sets to predict share price. The text from financial news were selected, and vectorized as an input to the proposed model. The authors concluded that the accuracy increased compared to previous models.

2.2. Bankruptcy Prediction on the Polish data set

Artificial Intelligent methods including ML and DL techniques have been extensively applied on the Polish data set in order to predict financial failure.

Zięba et al. (2016) used some extension of Extreme Gradient Boosting to classify companies, and they claimed that these results can be applied in every data in this field of research. Also, they developed a new method, called synthetic features, to ensure that data accurately represents higher-order statistics.

The goal of the research in García et al. (2019) was to investigate whether or not there is a connection between different types of positive samples and performance of ensemble classifiers, such as Bagging, AdaBoost, Random Subspace, Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Example, Décor (DECORATE), Rotation Forest, Random Forest and Stochastic Gradient Boosting. The authors used 14 real data sets and categorized them into five different types of groups. The Polish data sets belonged to unsafe category. At the end, by using the above techniques for unsafe groups, they concluded that Decorate produced a better accuracy in comparison with other ML methods.

The aim of another research Xiaomao et al. (2019) was to figure out how much every feature affects the prediction accuracy of different models. After examining the results of applying some feature selection approaches on the Polish data set, the authors concluded that by using SHapley Additive exPlanation (SHAP) method, the (mean) accuracy will increase compared to other techniques.

Marso and El Merouani (2020) suggested a hybrid model, comprising of a feedforward NN and cuckoo search algorithm, to forecast financial distress of companies in Polish data sets. After performing data preparation, they looked at the prediction power (accuracy) of three distinct models: LR, Back Propagation feedforward NN (BPNN), and Conic Section Function NN (CSFNN) on the data sets corresponding to one year and three years before bankruptcy. The results of applying CSFNN method ultimately outperformed other algorithms. They showed that the accuracy of using CSFNN were %90.30 and %82.70 for one year and three years before bankruptcy, respectively. Also, in another study, (Marso and EL Merouani, 2020) used Artificial Bee Colony (ABC) combined with an ANN model, called ABCNN; Back Propagation Neural Network (BPNN); and Multiple Discriminate Analysis (MDA) for Corporate Bankruptcy Prediction (CBP). The accuracy of ABCNN technique were 92.04 and 80.94 for one year and three years before bankruptcy.

Smiti and Soui (2020) proposed Borderline Synthetic Minority oversampling technique with Stacked AutoEncoder (BSM-SAES) approach to balance the Polish data sets and decreased the dimensionality of variables for predicting bankruptcy.

Lahmiri et al. (2020) used three distinct financial data sets featured with different types of attributes, including quantitative data set (similar to the first-year Polish companies' data), qualitative data set and credit scoring data set, which was the combination of quantitative and qualitative data. The ensemble classifiers like AdaBoost, LogitBoost, RUSBoost, subspace, and Bagging were applied on those data sets. The results showed that the performance of AdaBoost with 0.0532 lowest error was better other ensemble financial classification methods for this Polish data set, and RUSBoost, LogitBoost, Bagging and Subspace had in order the lowest error after AdaBoost.

After using five different classifiers, including RF, DT, ANN, KNN, and LR, Mahapatra et al. (2020) came to the conclusion that using preprocessing techniques such as filling missing data, normalizing, SMOTE increased the accuracy and F1-score apart from the PCA approach. The authors did not mention their exact accuracy numbers in their paper.

Quynh and Phuong (2020) developed a prediction model that combines three methods, including RF, GB, and Bagging. Then, they were applied to the whole 5-year data from the Polish data set, after imputing missing values based on SMOTE approach. The authors chose theses 3 models among 7 models including DT, LR, RF, Bagging, AdaBoost, GB, and EXB because of their high accuracy in comparison to others. Finally, the accuracy percent 99.52, AUC percent 99.78, and F1-score percent 95.12 were achieved by these three models.

In the study of Pisula (2020), the authors used Polish data sets where their financial statements belonged to 2010-2018. They demonstrated that boosting ensembles including boosted trees, Stochastic Gradient Boosting Machine (GBM), boosted C5.0 trees and boosted logit outperformed bagging and stacking ensembles.

The research of Soui et al. (2020) was implemented with the aim of contributing to the bankruptcy prediction by Stacked Auto-Encoders (SAE) and SoftMax classifiers. A two-layer auto-encoder which followed by a SoftMax layer were used on the Polish data set corresponding to the first year. The proposed model obtained 98% accuracy and %96.1, where the AUC is greater than the one obtained in (Zięba et al., 2016).

In another research, Shahee and Ananthakumar (2021) used an overlap-sensitive ANN, and implemented this method on the Polish data set. By applying this method, they found the overlapping features, and addressed the class imbalance and class overlapping by weighting the observations according to the position of their space before training the neural network. The results of AUC were 0.726, 0.673, 0.659, 0.739 and 0.787 respectively for each year.

Aljawazneh et al. (2021) used three different data sets including polish data to evaluate the performance of various DL models and five ensemble approaches. After applying oversampling technique, the authors applied Deep belief network (DBN), Long-short term Memory (LSTM), Multilayer Perceptron With 6 Layers (MLP 6L), RF, SVM, K_NN, and XGBoost. MLP_6L with SMOTE_ENN technique recognized as the best method with %99.67 Accuracy. One shortcoming of this work was that they modified the number of total samples in their studies.

The initial stage of research of Keya et al. (2021) was the study of application of Recursive Feature Elimination (RFE). This technique removes the weak features of the Polish data set, so the researchers reached 12 significant attributes from 64 attributes by using this method. The data was then subjected to the SMOTE technique. They applied AdaBoost, Decision tree, Random Forest, J48, and Bagging after identifying the key features and balancing the data on the Polish data set. The Bagging approach surpassed other methods with an accuracy of 97%.

Zhang et al. (2021) developed a credit scoring prediction model which includes three stages with enhanced outlier adaptation on Polish data sets. Those stages are Bagging-based Local Outlier Factor (BLOF) based outlier adaptation method, the dimension-reduced feature transformation method, and the stacking-based ensemble learning method with PCA. In addition, five base classifiers: XGBoost, GBDT, Adaboost, RF, and LightGBM were selected for ensemble operation according to their superior performance on AUC indicator. The ensemble model with PCA attained the best credit scoring predictive power for the first, second- and third-year data sets; whereas, the dimension-reduced feature transformation method achieved the best results for the fourth and fifth year data sets.

Another study on bankruptcy prediction based on Polish data sets belonged to the works of Jain et al. (2021). Firstly, the authors changed the imbalanced data to a balanced data by using SMOTE technique. Then, by using the fuzzy rough set, they decreased the size of the data set. In this step, they applied 6 ML algorithms on 5 Polish data sets, and finally they concluded that RF gave the high accuracy and specificity of %96.7 and %96.9 respectively on the second-year data.

In the study of Ren and Weiss (2021), the researchers applied different ML methods such as RF, DT, XGBoost and LR on Polish data set, one year before bankruptcy and a Chinese data set. Prior to implementing these approaches, the authors tried to identify the important features for theses models (except for) LR. The findings showed that while operation-related variables had more impact in the Polish data, asset-related features are more important in predicting bankruptcy in China. Also, they concluded that the XGBoost method achieved 97% accuracy after applying that feature selection strategy on the Polish data set.

Acharjya and Rathi (2021) selected 15 attributes from 64 attributes of the Polish data set as chief features by using the PCA technique. The main goal of their work was to make a comparison between statistical, rough computing, and mixed computing methodologies. Then, Rough Set (RS), Rough Set hybridization of Neural Network (RSNN), Rough Set hybridization of Binary-Coded Genetic Algorithm (RSBCGA), Rough Set hybridization of Real-Coded Genetic Algorithm (RSRCGA), Fuzzy-Rough hybridization of Real-Coded Genetic Algorithm (FRSRCGA) were applied to the data. Finally, the results showed that the accuracy of RSBCGA outperformed other methods.

2.3. The Existing Gaps in The Literature and Thesis Contributions

There is still a great need for research to advance the methods for predicting companies' bankruptcy, although there has been much research in this field according to what we reported in the literature review section. Many studies have focused on forecasting bankruptcy for different financial institutions, companies, and the government by statistical and artificial intelligence approaches (Altman, 1968; Atiya, 2001; Beaver, 1966; Fedorova et al., 2013; Min and Lee, 2005; Ohlson, 1980). One of the most effective methods of AI that have contributed to predict bankruptcy are conventional ML approaches such as logistic regression, support vector machine, naïve bayes, decision tree etc. However, parsing the literature we observe that there are insufficient studies that applied DL techniques to bankruptcy prediction.

More specifically, we observe insufficient studies on Polish data set. In fact, the result of our literature review revealed that there is a need for applying more specific CNN models on this data set. The reason might be the challenge to convert the data and expand its dimension in order to apply such DL models.

Moreover, it is not still clear what the impacts of different preprocessing techniques will be on a variety of performance measures when combined with DL approaches. This is important since the data set we are working with is imbalanced. That is, the proportion of bankrupted companies is way less that those that did not go bankrupt. Therefore, it is crucial to have an insight about the application of preprocessing techniques and DL approaches into our dataset.

The goal of this thesis is to predict the bankruptcy of Polish companies in the 1st, 2nd, 3rd, 4th, and 5th year by improving, making new models and extending the work of Zięba et al. (2016). We used the data in each year as well as the entire data set to create six data sets and analyzed them. These data sets include financial data of bankrupt and non-bankrupt companies. This study focuses on the bankruptcy prediction of Polish manufacturing enterprises between 2000 and 2013.

However, the methodologies utilized should be valid and relevant to other industries as well. In contrast to previous works which emphasized on ML methods, we seek to find out how well DL models such as different CNN methods achieve the accuracy, sensitivity and specificity.

The main contributions of this thesis can be summarized as follows:

1) Designing, applying and validating a CNN_1D model on the data with expanding the dimension of the data. In particular, we designed a format of data to be a shape of (64*1) such that CNN techniques can be applied to our non-image data set in a straightforward manner.

2) Designing, applying and validating CNN_2D and CNN_2D Residual Network models on the data with expanding the dimension of the data. This requires converting the shape of the data from (64*1) to (8*8*1). These three new CNNs models are novel and have not been explored in previous studies.

3) Evaluating whether or not various approaches of data preprocessing may enhance the power of CNNs methods for predicting the bankruptcy likelihood. This is particularly critical due to the unbalanced nature of our data set.

4) In terms of applying ANN methods, our literature review revealed that just one paper thoroughly discussed about this model on our data set. However, it was not clear how much the accuracy and sensitivity could be improved by tunning various hyperparameters. Therefore, we investigated how many hidden layers and which optimizer could result in improvement in both accuracy and sensitivity.

5) Comparison between the performance of the ML models, including LR with the three CNNs models and ANN models on six data sets in order to assess the performance of our proposed models.

6) Improving the accurate rate of detection of risky companies

7) Comparison between our results with those obtained in previous works validated on the same database.

3. Methodology

In this section, we describe the details of ML and DL techniques that are used for bankruptcy prediction on the Polish data set. The summary of our methodology is provided in the flow chart in Figure 1.

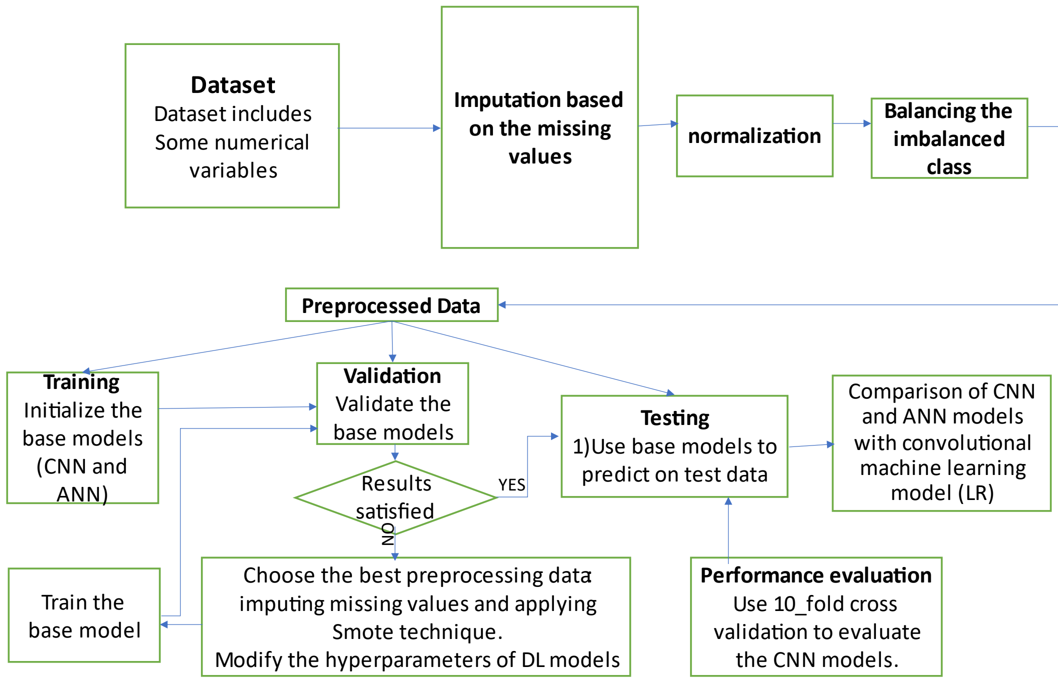
Dataset
Dataset includes Some numerical variables
Imputation based on the missing values
normalization
Balancing the imbalanced class
Preprocessed Data
Training
Initialize the base models (CNN and ANN)
Validation
Validate the base models
Results satisfied
YES
NO
Testing
1)Use base models to predict on test data
Comparison of CNN and ANN models with convolutional machine learning model (LR)
Train the base model
Choose the best preprocessing data imputing missing values and applying Smote technique.
Modify the hyperparameters of DL models
Performance evaluation
Use 10_fold cross validation to evaluate the CNN models.

Figure 1. Flow chart of proposed methodology

In a broad view, we use three techniques: 1- logistic regression as a benchmark, 2- artificial neural networks (ANNs), and 3- convolutional neural networks (CNNs). In addition, we use some techniques such as synthetic minority oversampling technique (SMOTE) in order to address the issue of imbalance in the Polish data set. Also, we applied the algorithm which imputes the missing data with mean for handling the missing values of the data set. In the remainder of the section, we will provide details on each of the methodologies that we have used. The description of these methods is adapted from. (Bishop and Nasrabadi, 2006)

3.1. Logistic Regression

Logistic regression is perhaps one of the widely used techniques of machine learning/data science in applications, where the response variable is dichotomous. For example, it is used in medical applications (Tu, 1996), social sciences (Pituch and Stevens, 2015), engineering (Khoshgoftaar and Allen, 1999), business (Wood, 2006), and economics (Strzelecka et al., 2020) to name a few. In essence, it is used for problems where the response variable can take binary values and several independent variables are available in the data set. In other words, logistic regression takes an input vector $\boldsymbol{x}$ and corresponding target variable $t \in \{0,1\}$ and decide (probabilistically) whether the input data belongs to 0 class C_0 or 1 class C_1. In big picture, logistic regression seeks to create a model for the conditional probability $p(C_k|x)$ for the inference, which falls under the umbrella of generalized linear models. Specifically, one can use Bayes' rule to find the posterior probability based on class conditional densities $p(x|C_k)$ and prior $p(C_k)$. Specifically, when we have two classes, we have the following formula for a generalized linear model

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x|C_1)p(C_1)+p(x|C_2)p(C_2)} = \frac{1}{1+exp(-a)} = \sigma(a) \qquad 3.1$$

where

$$a(x) = log\frac{p(\boldsymbol{x}|C_1)p(C_1)}{p(\boldsymbol{x}|C_2)p(C_2)} \qquad 3.2$$

and $\sigma(a)$ is the logistic sigmoid function given by the following formulation noting that $exp(.)$ is the exponential function

$$\sigma(a) = \frac{1}{1+exp(-a)}. \qquad 3.3$$

The S-shaped logistic function depicted in Figure 2 plays an important role in classification algorithms and it has useful properties such as

$$\sigma(-a) + \sigma(a) = 1. \qquad 3.4$$

In logistic regression the main modeling assumption is that the function $a(x)$ is linear in x. In particular, in logistic regression, we have

$$y(x) = p(C_1|x) = \sigma(w^\top x) \tag{3.5}$$

where w is the weight vector for the linear model and $\top$ denotes the transpose of a matrix (vector).

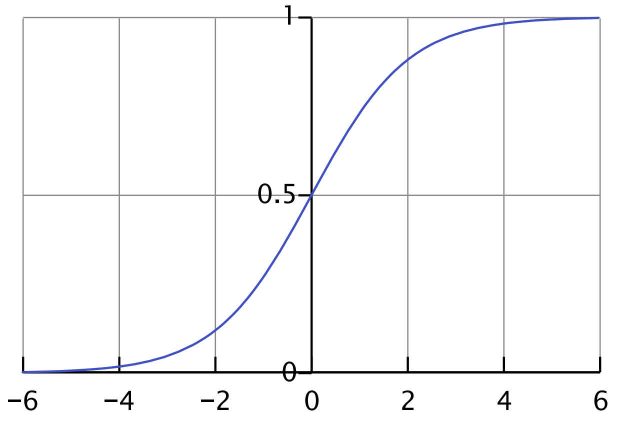

Figure 2: Logistic sigmoid function

Although the linear assumption at first glance might look restrictive, the activation function $\sigma(.)$ makes the overall relationship nonlinear. Moreover, its application in many domains revealed that logistic regression performs quite robustly. In this model the number of parameters corresponds to the number of features, i.e., $x = (x_1, x_2, \dots, x_M)$; that is, the feature vector is M dimensional, there will be M parameters corresponding to $w = (w_1, w_2, \dots, w_M)$. Clearly, if the number of features increases, the number of model parameters will also increase. One approach to reduce the number of parameters is to use basis functions or other reduction techniques.

We next describe how to estimate the parameters of the LR model, represented by w. The main approach is the maximum likelihood estimation. Let (x_n, t_n) for $n = 1,2, \dots, N$ be our data set where x_n is the nth data point and $t_n \in \{0,1\}$ be the binary response associated with the nth data point. Let $t = (t_1, t_2, \dots, t_N)$ and $y_n = p(C_1|x_n)$. The likelihood function is given by

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n}(1 - y_n)^{1-t_n}. \tag{3.6}$$

Then, cross entropy error function is defined as

$$E(w) = -logp(t|w). \quad 3.7$$

The goal is to minimize the above error function. Taking the derivative of this function yields

$$\nabla E(w) = \sum_{n=1}^{N} (y_n - t_n) x_n \quad 3.8$$

where $y_n = \sigma(a_n)$ and $a_n = w^\top x_n$. Unlike the usual regression model for which equating the derivative to zero provides a closed form solution; in logistic regression we must use iterative methods. Specifically, the standard method is to use Newton-Raphson iterative approach to find the minimum of the error function. The main ingredient for the algorithm is the Hessian matrix, denoted by H which is calculated by taking the all the partial second derivatives of $E(w)$ with respect to w. Then, the iterative method uses the following equation

$$w^{new} = w^{old} - H^{-1} \nabla E(w). \quad 3.9$$

Next, we characterize the derivative and the Hessian. Let X be the $N \times M$ design matrix whose nth row is simply $x_n^\top$, which plays an important role in linear models. Then we have

$$\nabla E(w) = X^\top (y - t) \quad 3.10$$
$$H = X^\top R X$$

where R is a diagonal matrix with $R_{nn} = y_n(1 - y_n)$, which can be interpreted as the variance of response. Although H is not constant like in the usual linear regression, it is easy to show that H is positive definite, making the error function convex with a unique optimal point.

This algorithm along with its advanced versions to cope with numerical issues are available in the library of Python and is used for the numerical results.

3.2. Artificial Neural Networks

Artificial neural networks (ANNs) or simply neural networks are a class of ML/DL techniques widely used in many fields, such as facial recognition (Singh et al., 2020), stock market prediction (Vui et al., 2013), social media (Wong et al., 2017), aerospace (Sangwan et al., 2015), defense (Zhang et al., 2018), healthcare (Sordo, n.d.), handwriting analysis (Djamal et al., 2013), and global change (Liu et al., 2010). In fact, the applicability of standard statistical methods such as logistic regression will become limited because of the so called curse of dimensionality. When the size of the problem becomes large, as is in real-world problems, the number of features significantly increase, which makes the problems of training and validation difficult. In addition, the performance of the standard statistical methods may significantly decline.

One approach to address these issues is to use basis functions that can be adaptive in nature. That is, we can assume parametric forms for the basis functions and allow their parameters to change during the training phase. This is the big picture idea of ANNs or alternatively feed-forward NNs, or multilayer perceptron. By using these basis functions, the model becomes cleaner and more compact although more effort must be put forth for training purposes because these basis functions introduce a significant amount of nonconvexity to the model.

The NN terminology originated in biological systems where scholars were seeking to present information processing by mathematical models (McCulloch and Pitts, 1943) and (Rosenblatt, 1961). In fact, some arguments have been made for the plausibility of this approach by connecting it to the basic mechanism of nature. However, the focus in ML is to provide reliable prediction for a variety of applications as mentioned above. ANN and its variants have been quite successful over the last couple of decades. Next, we will explain the mathematical foundations and models for ANNs and then describe parameter training and validation.

3.2.1. Mathematical Model

Recall that in generalized linear models with input vector $\boldsymbol{x}$ and parameter vector $\boldsymbol{w}$ the relationship between the input and output is presented by

$$y(\boldsymbol{x}, \boldsymbol{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x})\right) \qquad 3.11$$

where $\phi_j(\cdot)$s are basis functions and $f(\cdot)$ is an activation function. For example, in a simple linear regression, both basis functions and activation function are identity. The main mathematical idea for ANNs is to: 1) make the basis functions depend on the parameters; and 2) allow the parameters to be modified in the training phase. There are many ways to achieve these two modifications, but ANNs use a specific structure to this end by considering layers and activation functions.

First, define M linear combination of input variables as

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{3.12}$$

where D is the number of features, superscript (1) denotes the first layer of the network and $j = 1,2, \dots, M$. The technical terms for $w_{ji}^{(1)}$ are weights and for $w_{j0}^{(1)}$ are biases. Also, the technical term for a_j is activation. As can be seen, activations are simply a linear combination of input data. Then, the activations are transformed into

$$z_j = h(a_j) \tag{3.13}$$

where activation function $h(\cdot)$ is assumed to be nonlinear and differentiable. The standard activation functions are sigmoid and tanh. In ANNs, the quantities z_j are called hidden units. Now, we combine z_js to create output unit activations (we assume there is one hidden layer)

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{3.14}$$

where $k = 1,2, \dots, K$ and there are K outputs. The superscripts (2) correspond to the second layer and similar to the first layer $w_{k0}^{(2)}$ are biases. In our setting as an expository example, where there is only a single hidden layer, the next step is to transform the output activations a_k to network outputs y_k. This transformation is carried out via the activation function. For instance, if we choose a linear activation function, we will reach to a simple linear regression because the composition of linear transformations will be linear at the end. These types of linear network are not of interest generally. If we have a binary output, we can choose the logistic sigmoid function and set

$$y_k = \sigma(a_k). \tag{3.15}$$

For categorical output we can choose a softmax activation function and so on. Therefore, for our Polish data set with binary outcomes we will have the following equation

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)}\, h\left(\sum_{i=1}^{D} w_{ji}^{(1)}\, x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{3.16}$$

Figure 3 represents the schematics of ANNs. As can be seen from the figure, the system looks like neurons and synapsis, thus the name. Note that for simplicity we used a network with one hidden layer only. However, all the concepts can be easily generalized for multiple layers by using a linear function of weights and nonlinear activation functions in each layer. Recall that we pointed out to the adaptivity of parameter estimation, which is achieved by another type of generalization with skip layers. These skip layers can be constructed by connecting the input layer to output layer directly in our example of one hidden layer.

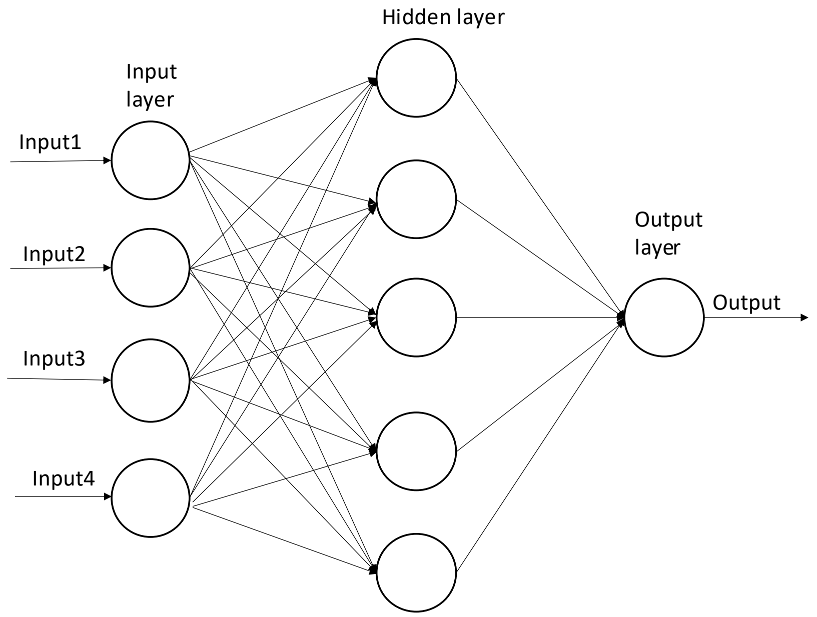

Figure 3: Artificial Neural Network

One crucial fact about the architecture of neural networks is that they must be feed-forward. That is, there should not be a closed directed cycle in the network because outputs should be a deterministic function of inputs and we cannot have cycles for estimation purposes. Finally, we notice that given a set of inputs and outputs we may have a lot of possible weights. That is, many different values for weights will result in the same transformation function and results. In other words, the choice of weights for a given input and output is not unique. Next, we will describe neural network model training which is a backbone of DL estimation methods.

3.2.2. Model Training

Let the data set be comprised of input vectors $\{\boldsymbol{x_n}\}$ for $n = 1,2,\dots,\ N$ and corresponding set of target vectors $\{\boldsymbol{t_n}\}$. Also let $X = \{\boldsymbol{x_1},\dots,\boldsymbol{x_n}\}$ and $\boldsymbol{t} = \{t_1,\dots,t_n\}$. The goal is to estimate $\boldsymbol{w}$ in the ANN. The first step is to give a Bayesian flavor to the data set to estimate precision as well. If the outcome is continuous, which is not the case in Polish data set, the probability distribution of the target t given $\boldsymbol{x}$ is assumed to be normally distributed with mean $y(\boldsymbol{x},\boldsymbol{w})$ and precision β, where precision is the inverse of variance. Therefore, the likelihood function is given by

$$p(\boldsymbol{t}|X,\boldsymbol{w},\beta) = \prod_{n=1}^{N} p(t_n|\boldsymbol{x_n},\boldsymbol{w},\beta) \tag{3.17}$$

where we assume that the N data points are independent. Using maximum likelihood estimation approach, one can easily show that the error function will become

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - t_n)^2 \tag{3.18}$$

where y_n is a shorthand notation for $y(\boldsymbol{x_n},\boldsymbol{w})$. This error function is a well-known formula. However, in Polish data set the outcome is binary. The standard approach to give a Bayesian flavor to construct the error function is to use Bernoulli random variable for the outcome. As noted earlier, we let $t = 1$ denote class C_1 and $t = 0$ and class C_2. Also, by using a logistic sigmoid activation function we have

$$y = \sigma(a) = \frac{1}{1 + exp(-a)} \tag{3.19}$$

where we can interpret $y(\boldsymbol{x}, \boldsymbol{w})$ as the conditional probability that the outcome belongs to class 1. The Bernoulli assumption implies that

$$p(t|\boldsymbol{x}, \boldsymbol{w}) = y^t(1-y)^{1-t} \tag{3.20}$$

If we assume that all the data point are independent, the error function, which is sometimes called cross-entropy error function, is the negative likelihood and is given by the following formula

$$E(\boldsymbol{w}) = -\sum_{n=1}^{N}\big(t_n y_n + (1-t_n)(1-y_n)\big) \tag{3.21}$$

This approach is flexible and can be used for more general outcomes like categorical outcomes and other activation functions like softmax. Now the crucial step is to find the parameter vector $\boldsymbol{w}$ to minimize the cross-entropy error function, which we will describe next.

One key step to minimize the error function is to find its gradient and equal it to zero since it is a continuous function and then check the Hessian to see if it is local or global minimum. In fact, the goal is to solve

$$\nabla E(\boldsymbol{w}) = 0 \tag{3.22}$$

However, there is no analytical solution for the equation above, unlike the regression, and we need to use numerical optimization methods to that end. There are several numerical techniques that are applied for numerically solving this equation for neural networks. For example, local quadratic approximation in which the Taylor approximation of the error function is used. However, this method needs the calculation of the Hessian matrix and could be computationally expensive. One approach that is discussed in literature and its advanced versions are proved useful in practice is gradient descent methods, which we explain next.

Gradient descent methods start with an initial guess about the minimizer, which we call $\boldsymbol{w^0}$. Let τ denote the iterations of the gradient descent method and η be the learning rate. The algorithm proceeds as follows from one step to the next

$$\boldsymbol{w}^{\boldsymbol{\tau+1}} = \boldsymbol{w}^{\tau} - \eta\nabla E(\boldsymbol{w}^{\tau}) \tag{3.23}$$

As can be seen, the important element in this formula is the gradient of the error function. There are several ways to estimate it, but the main approach is error backpropagation, which is a cornerstone of DL and we will describe it in detail. The learning rate is a tuning parameter and determines how fast we would like to traverse the direction of gradient. This method is continued until a notion of convergence is achieved. For example, if the norm between two consecutive iterations of the parameter vector is less than a given threshold. Some notes are in order.

First, gradient descent algorithm will not necessarily converge to a global optimal solution. One approach to find better solutions is to run the entire algorithm starting from different initial points and pick the best among them. Second, assuming independent data set one can write the error function as $E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w})$ which can facilitate the procedure. Third, there is an online version of gradient descent algorithm which has shown success in practice for neural networks. The idea is to update the weight vector based on the current data. That is, we use the following formula where there is a subscript n in the error function

$$\boldsymbol{w}^{\tau+1} = \boldsymbol{w}^{\tau} - \eta \nabla E_n(\boldsymbol{w}^{\tau}) \qquad 3.24$$

This online version which is sometimes called stochastic gradient descent method has several advantages over the standard version. For example, it can handle redundancy much better: suppose we duplicate the data set; the error function is going to be multiplied by 2 and the standard version is searching the enlarged space requiring a double computational time. But the online version will not be affected by doubling. In addition, they are less likely to be trapped in local optimal points. Next, we will describe error backpropagation.

3.2.3. Error Backpropagation

Error backpropagation is an essential technique in DL whose task is to find the derivative of the error function efficiently. In big picture, it is used for feed-forward NNs and can be thought of as a local message passing scheme where information is passed alternatively back and forth. We will describe this technique for a general neural network. Recall from the previous section that based on some natural assumptions the total error loss function can be represented by

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} E_n(\boldsymbol{w}) \tag{3.25}$$

So, if we calculate $\nabla E_n(\boldsymbol{w})$, the gradient of the loss function is retrieved. For a jumpstart, let us assume that we have a linear relationship between the outputs y_k and inputs x_i in the following sense

$$y_k = \sum_i w_{ki}\, x_i \tag{3.26}$$

with the error function

$$E_n = \frac{1}{2}\sum_k (y_{nk} - t_{nk})^2 \tag{3.27}$$

where $y_{nk} = y_k(\boldsymbol{x_n}, \boldsymbol{w})$. Therefore, it is clear that in this setting we have

$$\frac{\partial E_n}{\partial w_{ji}} = \left(y_{nj} - t_{nj}\right)x_{ni} \tag{3.28}$$

which has the following interpretations: Let w_{ji} be a link; x_{ni} can be thought of as the input of the link and $y_{nj} - t_{nj}$ can be thought of as the error signal of the output of the link. Although we presented this derivation for a linear function, similar terms will show up if we use logistic sigmoid activation functions or softmax. Interestingly, this idea extends to multilayer networks as follows.

Recall that each unit in the network is represented by

$$a_j = \sum_i w_{ji} z_i \tag{3.29}$$

where z_j acts as an activation unit that sends information to unit j. Then, the activation of unit j is a non-linear transformation of a_j via the activation function $h(\cdot)$ as follows

$$z_j = h(a_j) \tag{3.30}$$

By successive application of the above two formulae, we can calculate the activation of all hidden and output units starting from the input vectors. Thus, the name forward propagation. Now, the main step is to calculate the partial derivative of E_n with respect to w_{ji}. First, we observe that E_n depends on w_{ji} only through a_j. Therefore, using the chain rule we have

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \tag{3.31}$$

To ease notation, let $\delta_j = \frac{\partial E_n}{\partial a_j}$, which can be interpreted as errors. Due to the linear relationship between a_j and w_{ji} we simply have

$$z_i = \frac{\partial a_j}{\partial w_{ji}} \tag{3.32}$$

Finally, by substituting the above two formulae have

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{3.33}$$

Thus, we need to calculate the values of δ_js for all the hidden and output nodes. To that end, we first note that for the output nodes se simply have

$$\delta_k = y_k - t_k \tag{3.34}$$

which is easy to calculate. The next step is to calculate these δs for the hidden layers. We observe that change in a_j results in a change in the error function through changes in a_ks. Therefore, by using the chain rule we have

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j} \tag{3.35}$$

where summation above is over index k which points to the nodes in the network that send information to node j, a direction that is backward compared to the flow of information in the forward-feed neural networks. Thus, the name error backpropagation. Using the formulae stated above we conclude that

$$\delta_j = h'(a_j) \sum_k w_{kj}\, \delta_k \tag{3.36}$$

which is the backbone of this method. In particular, in order to calculate the value of error in a hidden node we can propagate the errors from the upper nodes in the network, starting from the error values at the output level, which is easy to calculate.

In summary, the error backpropagation works as follows: Starting from the input layers, calculate all the activations of all hidden and output nodes; calculate the error at the output level; backpropagate errors to find the errors in all hidden nodes of the network; use the formula to calculate desired derivatives.

Finally, we briefly describe regularization in NNs. Recall that the number of hidden layers in a neural network is arbitrary. However, there is a subtle tradeoff in choosing M. If we assume a fully connected network, and if M is large, the network can approximate any pattern in the training data set, but the danger of overfitting will become significant. If M is small, it will be difficult for the NN to have a small error in training. Therefore, there is a tradeoff between underfitting and overfitting. One approach to find a reasonable value for M is to train and validate the model for a variety of values for it and choose the best result. This approach is ad-hoc and data set dependent. There are also some general techniques such as consistent Gaussian priors and easy stopping that seek to systematically address this issue. In our implementation of the ANN for the Polish data set we choose one, two and three hidden layers. Furthermore, we also use convolutional neural networks to exploit the special structures and features of the Polish data set, which we will explain next.

3.3. Convolutional Neural Networks

Before we delve into the details of the construction of convolutional neural networks (CNNs) and how we applied them into the Polish data set, we will start with some motivation for such a design. Next, we will explain general structure for a convolutional neural network, and finally we describe the specific structures that we use for the CNNs for the application of Polish data set.

The main motivation for constructing CNNs is the concept of invariance, which implies that if the data is transformed in some ways, the output should not change. A major example is in image processing. Suppose that we have a handwritten digit as the input, like number 8. The same classification should be assigned (number 8) to the input if the position of the number in the image is changed, which is called translation invariance. Also, the same classification should be assigned if the size of the handwriting is changed, which is called scale invariance. Another example is in speech recognition in which small perturbation of nonlinear warping over time should not change the interpretation of the signal. This requirement poses a challenge to the traditional ANNs.

One way to address this challenge is to inject sufficiently large number of such transformed inputs into the data set. If there are a large number of them, one may expect that the ANN would learn these invariances and produce high quality solutions. However, this approach is impractical, and we need to develop other approaches for adaptive models that can exhibit the required invariances. There are three main approaches: 1- adding a regularization hyperparameter to the error function which penalizes the changes under the desired transformations, 2- adding a pre-processing stage where desired features for transformations are extracted, and 3- convolutional neural networks where the invariance properties are naturally built into the structure of the ANNs. Because CNNs have been successful in many applications in DL, we used several of its variants into the Polish data set. Before explaining each of them, we will explain the major components of a general CNN.

As mentioned above, CNNs build the invariance properties of the problem into the structure of the neural network itself. Since the main application area of CNNs is in image processing, we will explain CNNs in that context. Suppose we would like to recognize the digit written in a handwriting. The input is a set of pixel intensity values, and the output is a probability distribution over digits $0, 1, \dots, 9$. The inherit structure for transformation in this context is that the final result should be invariant under translations, scaling, rotation, and elastic deformation. In order to construct a CNN here, we note that the pixels near each other are more correlated than those distant pixels. We would like to exploit this property by extracting local features defined on small subregion of the input image. The information that we extract from each of these features can be combined in further layers of the network to detect higher-order features. If we continue this process, we would be able to get information about the whole image. This can help in image translation since local features that are extracted in one subregion can be used in other subregions.

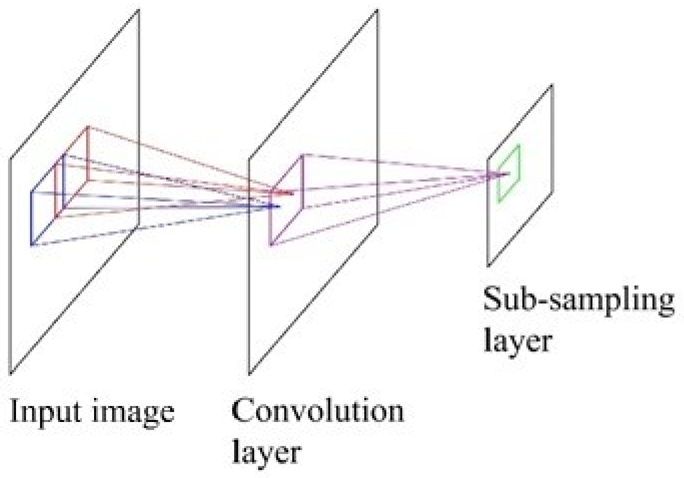

Figure 4: Convolutional Neural Networks

CNNs use three major mechanisms to extract the features: 1- local receptive fields, 2- weight sharing, and 3- subsampling. A schematic view of a CNN is depicted in Figure 4. As can be seen, we have an input layer, convolution layer, and subsampling layer. In the convolution layer units are organized in a feature map. Specifically, units in a feature map take inputs from their local receptive field, which is a small subregion of the input image. More importantly, the weights corresponding to units in each feature map is constrained to be the same. This property is important because it enables us to construct deep networks by restricting the number of weight parameters for estimation. Recall that traditional networks may suffer from gradient vanishing or explosion in the error backpropagation method due to excessive number of weights for estimation.

Let us use an example to clarify this construction. Suppose a feature map is a 10×10 grid having 100 units in the convolution layer, where each unit takes inputs from a 5×5 pixel patch of the input image. Therefore, the number of weight parameters will be 26: 25 for the actual weights for this feature map and one more for the bias term in the linear formula. Note that after the linear combination, we will transform it using a nonlinear function such as sigmoidal. One can interpret each unit as a feature detector, that is, all the units in a feature map can detect the same pattern. However, the location of the pattern can be anywhere in the original image. There is another interpretation for the fact that the weights are the same for a feature map: The evaluation of activations for these units is equivalent to applying a kernel to the intensities of the image pixels. Therefore, if we shift the image, the activations of the feature maps will also shift, which makes the model invariant to translations and distortion of the input. Because our goal is to hopefully detect more features, we will consider several feature maps in the convolution layer, where each of those feature maps has its own weight sharing.

Next, we will explain the sub-sampling layer which has some similarities and differences with the convolution layer. That is, the convolutional layer will play the role of the input for the units in the sub-sampling layer. Consider a feature map in the convolutional layer; we can construct planes in the sub-sampling layer where each unit there takes its input from a small receptive field in the convolutional layer. For our example, each sub-sampling unit may take its input from a 2×2 subregion of the corresponding feature map in the convolutional layer. For the corresponding operations in this stage, the map takes the average over the 2×2 units in the convolutional layer and multiplies it by a weight with a possibly bias term. Then, it uses a nonlinear function such as sigmoidal function for the final transformation. One difference between the convolutional layer and sub-sampling layer is that in the sub-sampling layer the receptive fields are non-overlapping while for the convolutional layer they are overlapping. We also make sure that the regions are contiguous for the sub-sampling layer. Thus, compared to the convolutional layer we will have a half number of rows and columns in the sub-sampling layer. This is critical because the model becomes less sensitive to shift perturbations in the corresponding region of the input image.

As noted, we described the ideas of CNNs for a single convolutional layer. However, in practice we can simply generalize this concept and consider network architects, where there are multiple layers of convolutional and sub-sampling layers. This makes the model more invariant with respect to aforementioned transformations. One might think that by an increase in the number of layers spatial resolution of the image will be deteriorated, but the increase in the number of features can compensate it. Finally, in the final layer of a CNN, the network is usually fully connected with a sigmoidal function for binary outcomes and softmax for categorical outcomes.

The process of training and validation for a CNN is similar to ANN. It usually uses error backpropagation. However, there is a subtle difference. In CNNs we need to ensure that the weights for a feature pattern remain the same and it needs a modification for the error backpropagation. In fact, as we mentioned earlier this is an appealing feature of CNN that needs a way smaller number of weights to train.

We applied the CNN technique to the Polish data set, and this is one of the main contributions of this thesis. The main idea is that the data set may have some invariant property because the scaling of values might not impact the result for bankruptcy. In addition, we might be interested in creating a model that is able to extract some features in this data set. To that end, we use three different CNN models: 1- CNN one-dimensional, 2- CNN two-dimensional, and 3- CNN_2D ResidualNetwork. The difference between one-dimensional and two-dimensional CNN is in the configuration of input. Because we have numerical feature variables in the Polish data set, we need to convert them to some sort of arrays to feed our model. More details are provided in chapter 5.

Another model that we use is a ResidualNetwork within a CNN. CNN_ 2D ResidualNetwork add the original input to the output of the convolution block. As a result, the vanishing gradient problem of deep network is solved. Also, the typical activation function in these models is ReLU.

3.4. Synthetic Minority Oversampling Technique (SMOTE)

One of the main techniques that we applied to the Polish data set for preprocessing is synthetic minority oversampling technique (SMOTE). First, we note that the Polish data set is very imbalanced. That is, the number of firms that do go bankrupt is way smaller than the number of firms that do not go bankrupt. Therefore, in our data set we have many rows with not bankrupt status and not many rows with bankrupt status. In general, imbalanced data poses significant challenges to ML models. In particular, their performance is typically poor for the minority class, which is the bankrupt status in our case. This is in fact bad news since these models are developed to have a good performance specially for the minority class.

To address this issue, SMOTE uses a simple and intuitive technique. It duplicates data in the minority class and injects them into the data set to make the number of data points for minority class larger. Note that these new data points are synthesized form the available data. This is a data augmentation approach and potentially can improve the performance of any ML model on minority data points. There are several mechanisms for the duplication, but a typical approach is to select data that are close in feature space, draw boundaries among the examples in the feature space, and then choose a data point on the boundary.

One other technique that we applied to the Polish data set is data normalization. The goal is to make the scale of all the features balanced. This will make the training more efficient. In fact, because the features of the input data set have very different scales and dimensions, we applied this technique to our data set. This is based on the fact that non-normalized data can produce low-quality solutions when applying ML models.

4. Experimental Settings

4.1. Data Description

The quality of the data set is crucial in ML/DL applications because the conclusion of the model depends on data quality. One major challenge in bankruptcy prediction models is that the data sets are usually not balanced in that the number of bankrupt companies is way less than the ones that did not go bankrupt. The data set we used for this thesis comes from Polish companies in the manufacturing sector because many companies went bankruptcy since 2004 in this sector in Poland. In fact, it does suffer from significant imbalanced data. In particular, the data set is from Emerging Markets Information Service (EMIS), which provides information on industry, news, macroeconomic statistics etc. In our case, the-still-operating companies were gathered from 2007 to 2013, and the bankrupt companies were assessed in the period of 2000-2012.

In this data set, there are formally two classifications for companies:

1) The companies which went bankrupt shown by 1 (Positive).
2) The companies which were not bankrupt shown by 0 (Negative).

The five individual data sets and sum of the whole years with some specific related numbers are shown in Table 1.

Table 1: Data sets used with their corresponding numbers

Year	**Bankruptcy after**	**Number of observations**	**Positive**	**Negative**	**Number of financial ratios**
1st	5 years	7027	271	6756	64
2nd	4 years	10173	400	9773	64
3rd	3 years	10503	495	10008	64
4th	2 years	9792	515	9277	64
5th	1 years	5910	410	5500	64
Total	NA	43405	2091	41314	64

we used the Weka software to combine all data together and calculated the accuracy of the entire five years data set since our data has an “arff” format.

In the first year, the data includes the financial rates from the first year for bankruptcy prediction and related bankruptcy labels showing if bankruptcy will occur after 5 years. The number of companies, in this data set, that were bankrupted and not bankrupted is 271 and 6756, respectively, after the forecasting period.

In the second year, the data includes the financial rates from the second year for bankruptcy prediction and related bankruptcy labels showing if bankruptcy will occur after 4 years. The number of companies, in this data set, that were bankrupted and not bankrupted is 400 and 9773, respectively, after the forecasting period.

In the third year, the data includes the financial rates from the third year for bankruptcy prediction and related bankruptcy labels showing if bankruptcy will occur after 3 years. The number of companies, in this data set, that were bankrupted and not bankrupted is 495 and 10008, respectively, after the forecasting period.

In the fourth year, the data includes the financial rates from the fourth year for bankruptcy prediction and related bankruptcy labels showing if bankruptcy will occur after 2 years. The number of companies, in this data set, that were bankrupted and not bankrupted is 515 and 9277, respectively, after the forecasting period.

In the fifth year, the data includes the financial rates from the fifth year for bankruptcy prediction and related bankruptcy labels showing if bankruptcy will occur after 1 years. The number of companies, in this data set, that were bankrupted and not bankrupted is 410 and 5500, respectively, after the forecasting period.

4.2. Definition of Input Variables

The Polish data sets described above, provided by Zięba et al. (2016) for the first time, includes numerous observations, many missing values and very unequal class distribution. By using 64 financial features that most of them are financial ratios, the financial situation of companies is predicted. In Table 2, we describe every indicator that used in this study(Zięba et al., 2016). These features are important to mention in order to understand the variables that are used for prediction.

Table 2: Features of Polish data set

Features	Description	Features	Description
X1	Net profit / total assets	X16	(Gross profit + depreciation) / total liabilities
X2	Total liabilities / total assets	X17	total assets / total liabilities
X3	Working capital / total assets	X18	gross profit / total assets
X4	Current assets / short-term liabilities	X19	Gross profit / sales
X5	[(Cash + short-term securities + receivables – short-term liabilities) / (operating expenses – depreciation)] * 365	X20	(Inventory ∗ 365) / sales
X6	Retained earning / total assets	X21	Sales (n) / sales (n-1)
X7	EBIT / total assets	X22	Profit on operating activities / total assets
X8	Book value of equity / total liabilities	X23	Net profit / sales
X9	Sales / total assets	X24	Gross profit (in 3 years) / total assets
X10	Equity / total assets	X25	(Equity - share capital) / total assets
X11	(Gross profit + extraordinary items + financial expenses) / total assets	X26	(Net profit + depreciation) / total liabilities
X12	Gross profit / short-term liabilities	X27	Profit on operating activities / financial expenses
X13	(Gross profit + depreciation) / sales	X28	Working capital / fixed assets
X14	(Gross profit + interest) / total assets	X29	Logarithm of total assets
X15	(Total liabilities ∗ 365) / (gross profit + depreciation)	X30	(Total liabilities - cash) / sales

X31	(Gross profit + interest) / sales	X48	EBITDA (profit on operating activities - depreciation) / total assets
X32	(Current liabilities ∗ 365) / cost of products sold	X49	EBITDA (profit on operating activities - depreciation) / sales
X33	Operating expenses / short-term liabilities	X50	Current assets / total liabilities
X34	Operating expenses / total liabilities	X51	Short-term liabilities / total assets
X35	Profit on sales / total assets	X52	(Short-term liabilities ∗ 365) / cost of products sold)
X36	Total sales / total assets	X53	Equity / fixed assets
X37	(Current assets - inventories) / long-term liabilities	X54	Constant capital / fixed assets
X38	Constant capital / total assets	X55	Working capital
X39	Profit on sales / sales	X56	(Sales - cost of products sold) / sales
X40	(Current assets - inventory - receivables) / short-term liabilities	X57	(Current assets - inventory - short-term liabilities) / (sales - gross profit - depreciation)
X41	Total liabilities / ((profit on operating activities + depreciation) ∗ (12/365))	X58	Total costs /total sales
X42	Profit on operating activities / sales	X59	Long-term liabilities / equity
X43	Rotation receivables + inventory turnover in days	X60	Sales / inventory
X44	(Receivables ∗ 365) / sales	X61	Sales / receivables
X45	Net profit / inventory	X62	(Short-term liabilities ∗365) / sales
X46	(Current assets - inventory) / short-term liabilities	X63	Sales / short-term liabilities
X47	(inventory ∗ 365) / cost of products sold	X64	Sales / fixed assets

4.3. Missing Values and Imbalanced Data set

We encountered some observations in the data set that one or more attributes have missing values. The identification and recording of observations with missing data for these data sets is crucial. With paying close attention to the data, we recognized that there are some missing values for each data point. Thus, we have handled the missing values in order to improve the prediction results by substituting them with mean values.

It is important to understand the imbalanced distribution of data set because it impacts the prediction performance of the proposed models. As we analyzed the data, we observed that the number of bankrupt firms are less than not bankrupt ones significantly in these data sets. For example, the data of the first year comprised 271 bankrupt (Positive) and 6756 non-bankrupt (negative) companies. Thus, we have implemented some methods to balance our data sets as follows.

4.4. Preprocessing

The preprocessing strategy depends mostly on the number of steps including normalization, imputing missing values and handling the imbalanced data.

The concept of "Re-scaling" data before training is one of the most significant topics in the field of data preparation. When the variables are on a smaller scale, e.g., in range [0,1] similar to our study, ML algorithms typically perform better. One scaling method is to use the Min-Max normalization function from Scikit-Learn library (https://scikit-learn.org/) in Python programming language. The formulation of the normalization is as follows:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad 4.1$$

In the second step, we have used Simpleimputer class to impute the missing values by the mean throughout every column. By doing this, we replaced the NaN values in our data to their corresponding new number.

As we mentioned in the first section, there is another important concern about this data set being imbalanced. For tackling this issue in classification problems, one popular approach that we also used is a technique called the SMOTE technique that relies on producing new samples of the minority class. This method increases the data points of the minority class in order to balance the imbalanced data set. It is important to know that these new data points are created by synthesizing the existing data(Fernandez et al., 2018).

By doing all above steps including normalizing, imputing missing values and solving the unbalanced data problem, the data is prepared to achieve good results (without bias) after applying the proposed DL models.

4.5. Training Data and Validation

We used 80 percentage of the data in order to train our models and the rest to assess whether or not the models will perform well on a new data set, which are respectively called training data and test data (Shmueli, 2017).

4.6. Performance Metrics

The main metrics for evaluation for our proposed models are based on three statistical indices including classification accuracy, sensitivity, and specificity. Accuracy is defined as the total proportion of individual companies that are correctly classified. The sensitivity of a model is its ability to determine bankrupt firms correctly, while the specificity of model shows its ability to correctly rule out non-bankrupt companies from the bankrupt ones. In particular, we have the following formulations

$$\text{Accuracy}=\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

$$\text{Sensitivity}=\frac{\text{TP}}{\text{TP+FN}}$$

$$\text{Specificity}=\frac{\text{TN}}{\text{TN+FP}}$$

where:

True positive (TP) = the number of cases correctly identified as the bankrupt companies.

False positive (FP) = the number of cases incorrectly identified as the bankrupt companies.

True negative (TN) = the number of cases correctly identified as the non-bankrupt companies.

False negative (FN) = the number of cases incorrectly identified as the non-bankrupt companies.

5. Computational Experiments

This section describes the details and outcomes of implementing several ML and DL methods, such as LR, ANN, CNN One-Dimensional, CNN Two-Dimensional and CNN ResidualNetwork with fine tuning on the Polish data sets. It also provides a comparison between the results of applied models. The results of all methods are executed in Google Colab platform (https://colab.research.google.com). The details of every step are described next.

5.1. Results of Logistic Regression

After preprocessing and cleaning the data (normalizing the data and imputing the missing values), we applied LR in two different situations: 1) by using SMOTE method, 2) without using SMOTE technique to see whether or not balancing the data affects the accuracy and sensitivity. As can be seen, the accuracy and the sensitivity metrics are reported in Table 3 and Table 4. These two are commonly used to assess the performance of a LR classifier, especially when we have imbalanced data sets.

Table 3: Results of applying LR (without SMOTE)

Year	Accuracy	Sensitivity
First year	0.9623	0.04
Second year	0.9626	0.0
Third year	0.9476	0.01
Fourth year	0.9443	0.02
Fifth year	0.9247	0.10
Total	0.9627	0.02

After analyzing the result of applying Logistic Regression, as we thought the sensitivity was not good enough as the data was so imbalanced. In fact, the number of bankrupted companies was only ~5% of the whole data set. So, we used SMOTE technique in order to solve that problem, and the results are shown in Table 4.

Table 4:Results of applying LR (after SMOTE)

Year	Accuracy	Sensitivity
First year	0.70	0.76
Second year	0.71	0.84
Third year	0.70	0.78
Fourth year	0.70	0.66
Fifth year	0.79	0.78
Total	0.59	0.79

As one can compare the results of Table 3 and Table 4, after applying the SMOTE technique, the sensitivity increased significantly. We can make a conclusion that by changing the imbalanced data set to a balanced one, we can considerably improve the prediction power of the number of bankrupted companies.

5.2. Results of Artificial Neural Networks

In the following tables, we can see the results of applying NN methods on the data tuned with different hyperparameters without Smote and with Smote technique after pre-processing the data. In this study, we explored and investigated both accuracy and sensitivity of ANN models. We used one, two and three fully connected layers with different units. To be consistent, we ran each model 100 times and reported the accuracy and sensitivity of each test. In addition, we explored different optimizers in Python. As can be seen, the following tables provide insight about each combination of (preprocessing, number of units in ANN, and optimizer) and how accuracy and sensitivity are changing within each tuple. As the loss function, for all these experiments, we used binary cross-entropy as there are 2 classes in output layers.

In the two following tables, one can see the results with these parameters:

One hidden layer, units=64, optimizer=Adam.

Table 5:Results of applying ANN (1, 64, Adam) (without SMOTE)

Year	**Accuracy**	**Sensitivity**
First year	0.97	0.43
Second year	0.965	0.098
Third year	0.953	0.196
Fourth year	0.95	0.07
Fifth year	0.93	0.41
Total	0.97	0.294

Table 6: Results of applying ANN (1, 64, Adam) (with SMOTE)

Year	**Accuracy**	**Sensitivity**
First year	0.95	0.65
Second year	0.93	0.51
Third year	0.923	0.514
Fourth year	0.9	0.54
Fifth year	0.94	0.67
Total	0.93	0.54

In the two next tables, one can see the results with these parameters:

Two hidden layers, units of every layer=64, optimizer=Adam.

Table 7: Results of applying ANN (2, 64, 64, Adam) (without SMOTE)

Year	**Accuracy**	**Sensitivity**
First year	0.975	0.65
Second year	0.97	0.295
Third year	0.96	0.34
Fourth year	0.95	0.27
Fifth year	0.94	0.566
Total	0.97	0.364

Table 8: Results of applying ANN (2, 64, 64, Adam) (with SMOTE)

Year	**Accuracy**	**Sensitivity**
First year	0.96	0.63
Second year	0.96	0.39
Third year	0.92	0.44
Fourth year	0.93	0.407
Fifth year	0.94	0.677
Total	0.95	0.61

In the two next tables, one can see the results with these parameters:

Two hidden layers, units of first layer=128, units of second layer=32, optimizer=RMSprop.

Table 9: Results of applying ANN (2, 128, 32, RMSprop) (without SMOTE)

Year	Accuracy	Sensitivity
First year	0.975	0.57
Second year	0.97	0.34
Third year	0.95	0.33
Fourth year	0.95	0.26
Fifth year	0.94	0.44
Total	0.975	0.4

Table 10: Results of applying ANN (2, 128, 32, RMSprop) (with SMOTE)

Year	Accuracy	Sensitivity
First year	0.96	0.57
Second year	0.96	0.58
Third year	0.92	0.44
Fourth year	0.94	0.46
Fifth year	0.94	0.61
Total	0.96	0.56

In the two next tables, one can see the results with these parameters:

Three hidden layers, first units=32, second units=32, third units=32, optimizer=Adam.

Table 11: Results of applying ANN (3, 32, 32, 32, Adam) (without SMOTE)

Year	**Accuracy**	**Sensitivity**
First year	0.98	0.61
Second year	0.97	0.35
Third year	0.96	0.36
Fourth year	0.93	0.23
Fifth year	0.93	0.54
Total	0.97	0.31

Table 12: Results of applying ANN (3, 32, 32, 32, Adam) (with SMOTE)

Year	**Accuracy**	**Sensitivity**
First year	0.97	0.61
Second year	0.96	0.34
Third year	0.93	0.44
Fourth year	0.93	0.31
Fifth year	0.92	0.61
Total	0.97	0.47

In the two next tables, one can see the results with these parameters:

Three hidden layers, first units=128, second units=128, third units=128, optimizer=Adam.

Table 13: Results of applying ANN (3, 128, 128, 128, Adam) (without SMOTE)

Year	Accuracy	Sensitivity
First year	0.973	0.61
Second year	0.96	0.32
Third year	0.952	0.34
Fourth year	0.94	0.35
Fifth year	0.92	0.59
Total	0.971	0.41

Table 14: Results of applying ANN (3, 128, 128, 128, Adam) (with SMOTE)

Year	Accuracy	Sensitivity
First year	0.96	0.59
Second year	0.96	0.45
Third year	0.92	0.3
Fourth year	0.94	0.32
Fifth year	0.93	0.54
Total	0.96	0.55

The next graph will show each data on the horizontal axis and the sensitivity on vertical axis after using smote technique.

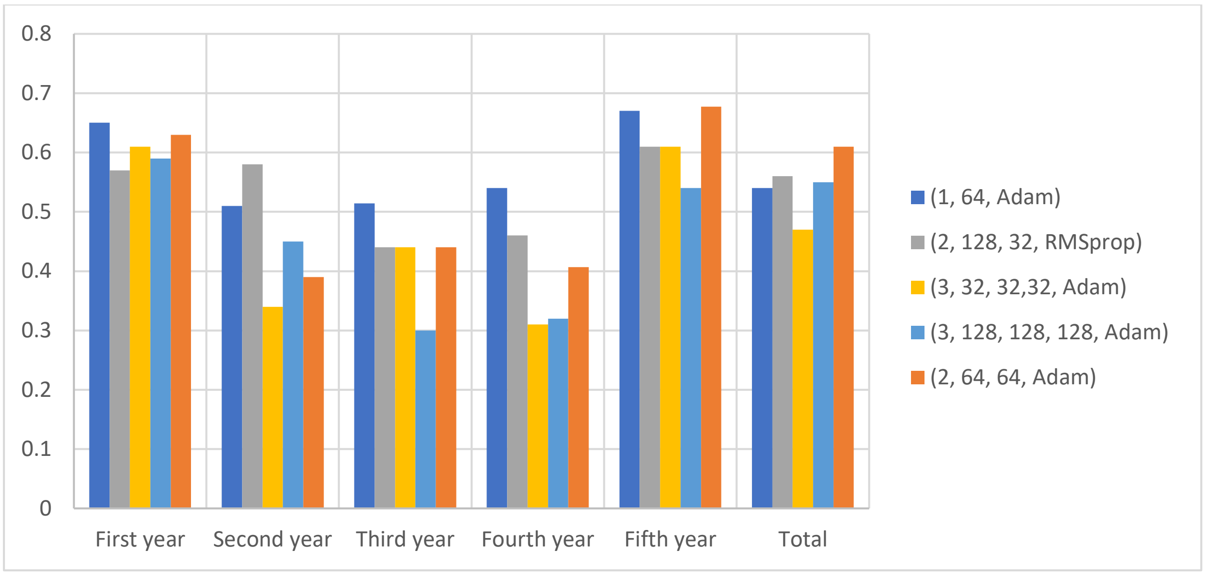

Figure 5: Sensitivity with different network architecture applied on different years

From the results of Table 5 to Table 14 and Figure 5, it can be concluded that using the SMOTE technique has a significant impact on improving the sensitivity, which is regarded as an important performance evaluation factor in the bankruptcy prediction. Also, we can conclude that whether we use the SMOTE technique or not, the accuracy is almost the same. Another important point that can be understood from these tables is that when we tuned the data with various hyperparameters, the best results belong to using 1 hidden layer with 64 units, as well as applying the Adam optimizer. Therefore, we used these results, best results for ANN, to compare with traditional ML techniques like LR and CNN methods, with and without implementing a data normalization technique.

5.3. Results of Convolutional Neural Network

In this study, we explored and investigated accuracy, sensitivity, and specificity of three CNN models: CNN_1D, CNN_2D and CNN_2D_ResidualNetwork tunning with different hyperparameters.

If we want to apply CNN models to this data, it is important that we change the data structure to based on the CNN model targets (1_D or 2_D). After reshaping the data set, for all three models, we used two fully connected layers at the end of the models' pipeline, followed by a sigmoid activation layer to predict whether the companies will bankrupt or not. To be consistent, we run each model with 100 epochs and 100 batch_size and reported the accuracy of test data. We also used RMSprop optimizer to run all models.

5.3.1. Convolutional Neural Network_1D

One of the main contributions of this thesis is the novel application of CNNs on the Polish data set. We have several approaches of applying CNNs. As the first approach, we trained a CNN_1D model on every input. To that end, we expanded the dimension of both the training and test data. Then, we had each sample with the shape of (64*1). To elaborate more, let us explain with an example with the first-year data set. The number of observations and features for this data set are 7027 and 64 respectively. For using CNN_1D, we need to add one more dimension as the channel. The data is reshaped from (7027, 64) to (7027, 64*1). Therefore, every sample of this data set reshaped in an array of dimension (64*1). Our network has two encoder blocks, and each block has 2 one-dimensional convolutional layers, followed by 2 fully connected layers. Rectified Linear Units (ReLU) is used as the activation function in the middle layer, and in the last layer we used Sigmoid activation function. By selecting batch size of 100, a patch of (100*64*1) is pushed to the model and the output is a list of 100 binary element which shows either the company was bankrupted or not. Figure 6 shows the schematic of this method for better understanding of the procedure.

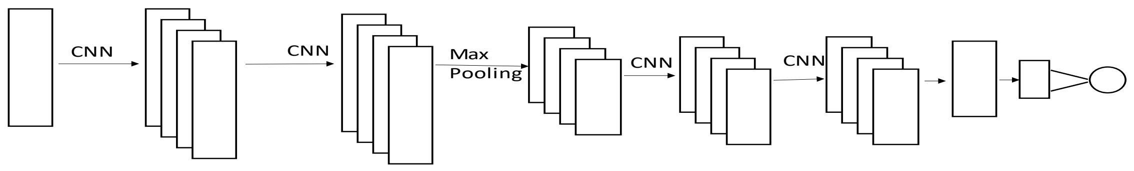

Figure 6: CNN_1D

The following tables report the results of applying CNN_1D on the data with four approaches of preprocessing the data: 1) with normalization, imputing missing values and SMOTE technique, 2) without normalization, 3) without normalization and SMOTE technique, 4) without SMOTE. The results of each combination provide insight about the importance of each element on the performance.

Table 15: Results of CNN_1D (with normalization, imputing missing values and SMOTE)

Year	**Accuracy**	**Sensitivity**	**Specificity**
First year	0.97	0.1	0.99
Second year	0.76	0.38	0.77
Third year	0.88	0.48	0.89
Fourth year	0.77	0.47	0.78
Fifth year	0.91	0.43	0.94
Total	0.92	0.43	0.93

Table 16: Results of CNN_1D (without normalization)

Year	**Accuracy**	**Sensitivity**	**Specificity**
First year	0.89	0.71	0.89
Second year	0.89	0.48	0.90
Third year	0.81	0.63	0.82
Fourth year	0.71	0.73	0.70
Fifth year	0.89	0.59	0.91
Total	0.94	0.40	0.96

Table 17: Results of CNN_1D (without normalization and SMOTE)

Year	Accuracy	Sensitivity	Specificity
First year	0.96	0	1
Second year	0.97	0.1	0.99
Third year	0.95	0	0.99
Fourth year	0.95	0.01	0.99
Fifth year	0.92	0.02	0.99
Total	0.96	0	1

Table 18: Results of CNN_1D (without SMOTE)

Year	Accuracy	Sensitivity	Specificity
First year	0.96	0	1
Second year	0.97	0	1
Third year	0.95	0	1
Fourth year	0.95	0	1
Fifth year	0.92	0.06	0.98
Total	0.96	0	1

5.3.2. Convolutional Neural Network_2D

In the second approach for applying CNN, we implemented the CNN_2D model. To prepare the data, each sample was reshaped from (64*1) to (8*8*1). For instance, for the first-year dataset mentioned in CNN_1D section, we need to reshape the data from (7027, 64) to (7027, 8*8*1) for CNN_2D models. Our network has two encoder blocks. Each block has two-dimensional convolutional layers, followed by 2 fully connected layers. In the middle layers and the last layer, we used ReLu and Sigmoid activation functions, respectively. The schematic of this approach is shown in Figure 7 in details in order to understand how it works.

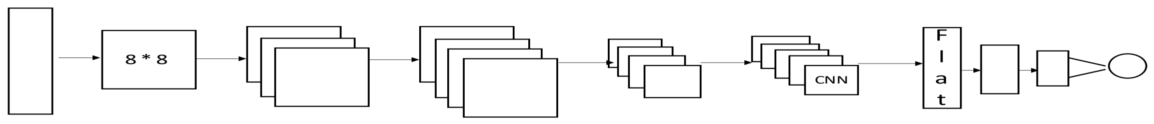

Figure 7: CNN_2D

Similar to CNN_1D we implemented several variations of CNN_2D on the data with some different ways in the next tables: 1) with normalization, imputing missing values and SMOTE technique, 2) without normalization, 3) without normalization and SMOTE technique, 4) without SMOTE. These results also shed light on the importance of each feature on the performance metrics.

Table 19: Results of CNN_2D (with normalization, imputing missing values and SMOTE)

Year	**Accuracy**	**Sensitivity**	**Specificity**
First year	0.44	0.78	0.44
Second year	0.154	0.90	0.13
Third year	0.856	0.27	0.887
Fourth year	0.746	0.61	0.754
Fifth year	0.906	0.31	0.955
Total	0.51	0.558	0.508

Table 20: Results of CNN_2D (without normalization)

Year	Accuracy	Sensitivity	Specificity
First year	0.91	0.57	0.92
Second year	0.93	0.32	0.94
Third year	0.85	0.40	0.87
Fourth year	0.87	0.38	0.89
Fifth year	0.86	0.49	0.89
Total	0.88	0.59	0.88

Table 21: Results of CNN_2D (without normalization and SMOTE)

Year	Accuracy	Sensitivity	Specificity
First year	0.96	0.22	0.99
Second year	0.96	0.14	0.99
Third year	0.95	0.23	0.98
Fourth year	0.93	0.22	0.96
Fifth year	0.93	0.23	0.98
Total	0.96	0.14	0.99

Table 22: Results of CNN_2D (without SMOTE)

Year	Accuracy	Sensitivity	Specificity
First year	0.96	0	1
Second year	0.97	0	1
Third year	0.95	0	1
Fourth year	0.95	0	1
Fifth year	0.92	0.01	0.99
Total	0.96	0	1

5.3.3. Convolutional Neural Network_2D_ResidualNetwork

ResidualNetwork (ResNet) is as a particular type of NN. Because of the vanishing gradient problem, deep networks are challenging to train. So, skip connections are used in the model to solve that issue. What skip connections do in CNN_2D_ResidualNetwork is adding original input to convolution block's output. By doing that, the more vanishing gradient problem decreases, the more the model learning speed increases.

In the proposed model of our study, additional layers are added to our CNN network that are effective in tackling complicated problems in order to improve accuracy and performance. The same as CNN_2D, we expanded the dimension of each sample, and we reshaped the matrix from (64*1) to (8*8*1). Our network has two encoder blocks, and each block has 2 two-dimensional convolutional layers. The output of second convolutional layer concatenates with the input of first convolutional layer as the residual connection. Then, the output of second block will be connected to 2 fully connected layers. In the middle layers, we used ReLU activation function.

The schematic of this method is shown in Figure 8 to better understand their structure and their work.

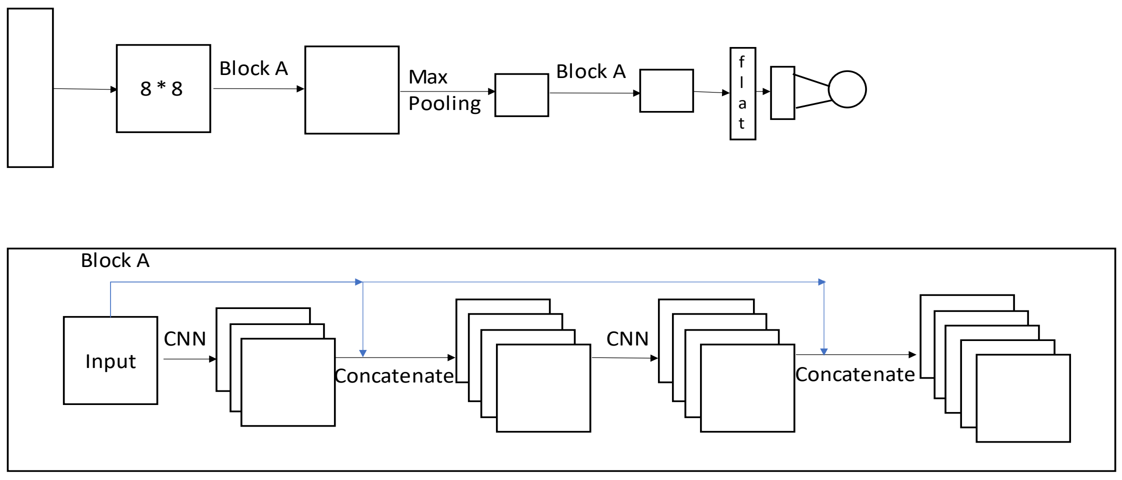

Figure 8: CNN_2D_ResidualNetwork

The next tables show the results of applying CNN_2D_ResidualNetwork on the data with four approaches like the previous CNN models: 1) with normalization, imputing missing values and SMOTE technique, 2) without normalization, 3) without normalization and SMOTE technique, 4) without SMOTE.

Table 23: Results of CNN_2D_ResidualNetwork (with normalize, imputing missing values and SMOTE)

Year	**Accuracy**	**Sensitivity**	**Specificity**
First year	0.62	0.08	1
Second year	0.97	0	1
Third year	0.93	0.07	0.97
Fourth year	0.65	0.22	0.91
Fifth year	0.73	0.77	0.72
Total	0.94	0.06	0.97

Table 24: Results of CNN_2D_ResidualNetwork (without normalize)

Year	**Accuracy**	**Sensitivity**	**Specificity**
First year	0.93	0.47	0.94
Second year	0.84	0.52	0.848
Third year	0.92	0.44	0.94
Fourth year	0.89	0.33	0.92
Fifth year	0.85	0.37	0.89
Total	0.88	0.41	0.90

Table 25: Results of CNN_2D_ResidualNetwork (without normalization and SMOTE)

Year	Accuracy	Sensitivity	Specificity
First year	0.97	0.04	1
Second year	0.96	0.13	0.99
Third year	0.95	0.07	0.99
Fourth year	0.95	0.04	0.99
Fifth year	0.93	0.34	0.98
Total	0.97	0.17	0.99

Table 26: Results of CNN_2D_ResidualNetwork (without SMOTE)

Year	Accuracy	Sensitivity	Specificity
First year	0.96	0	1
Second year	0.97	0	1
Third year	0.94	0	1
Fourth year	0.95	0	1
Fifth year	0.92	0.01	0.99
Total	0.96	0	1

After comparing theses results, one main insight that we can glean is that when we imputed missing values and applied the SMOTE technique (without normalization of data), we obtained the best results. So, in the next step, we wanted to check how the results may change with respect to different learning rates. To that end, we obtained the new results by implementing three different CNN models without data normalization approach (the best one among other approaches according to the results of previous steps) with 3 different Learning Rates (LR): 10^{-3}, 10^{-4}, and 10^{-5}. The following tables show the results after applying CNN_1D, CNN_2D and CNN_2D_ResidualNetwork with LR = 10^{-3}, 10^{-4}, 10^{-5}. These tables provide insight about the role that learning rate plays in deep learning applied to our data set.

Table 27: Results of CNN_1D (LR = 10^{-3})

Year	Accuracy	Sensitivity	Specificity
First year	0.86	0.75	0.86
Second year	0.13	0.94	0.09
Third year	0.11	0.98	0.05
Fourth year	0.11	0.99	0.06
Fifth year	0.08	1.00	0
Total	0.09	0.97	0.05
Average	0.23	0.938	0.185

Table 28: Results of CNN_1D (LR = 10^{-4})

Year	Accuracy	Sensitivity	Specificity
First year	0.89	0.71	0.89
Second year	0.89	0.48	0.90
Third year	0.81	0.63	0.82
Fourth year	0.71	0.73	0.70
Fifth year	0.89	0.59	0.91
Total	0.87	0.48	0.89
Average	0.84	0.60	0.85

Table 29: Results of CNN_1D (LR = 10^{-5})

Year	Accuracy	Sensitivity	Specificity
First year	0.89	0.61	0.897
Second year	0.84	0.42	0.85
Third year	0.85	0.48	0.87
Fourth year	0.77	0.42	0.78
Fifth year	0.70	0.74	0.702
Total	0.88	0.50	0.89
Average	0.82	0.528	0.83

Table 30: Results of CNN_2D (LR = 10^{-3})

Year	Accuracy	Sensitivity	Specificity
First year	0.93	0.607	0.94
Second year	0.918	0.394	0.93
Third year	0.898	0.43	0.923
Fourth year	0.88	0.417	0.90
Fifth year	0.918	0.36	0.96
Total	0.90	0.54	0.91
Average	0.907	0.458	0.927

Table 31: Results of CNN_2D (LR = 10^{-4})

Year	Accuracy	Sensitivity	Specificity
First year	0.91	0.57	0.92
Second year	0.93	0.32	0.94
Third year	0.85	0.40	0.87
Fourth year	0.87	0.38	0.89
Fifth year	0.86	0.49	0.89
Total	0.88	0.59	0.88
Average	0.878	0.4583	0.8983

Table 32: Results of CNN_2D (LR = 10^{-5})

Year	Accuracy	Sensitivity	Specificity
First year	0.937	0.53	0.95
Second year	0.95	0.24	0.97
Third year	0.696	0.691	0.696
Fourth year	0.686	0.514	0.696
Fifth year	0.898	0.622	0.92
Total	0.93	0.33	0.954
Average	0.8495	0.4878	0.8643

Table 33: Results of CNN_2D_ResidualNetwork (LR = 10^{-3})

Year	Accuracy	Sensitivity	Specificity
First year	0.93	0.372	0.947
Second year	0.89	0.41	0.91
Third year	0.87	0.49	0.889
Fourth year	0.679	0.65	0.681
Fifth year	0.889	0.58	0.914
Total	0.688	0.79	0.684
Average	0.827	0.539	0.841

Table 34: Results of CNN_2D_ ResidualNetwork (LR = 10^{-4})

Year	Accuracy	Sensitivity	Specificity
First year	0.93	0.47	0.94
Second year	0.84	0.52	0.848
Third year	0.92	0.44	0.94
Fourth year	0.89	0.33	0.92
Fifth year	0.85	0.37	0.89
Total	0.88	0.41	0.90
Average	0.906	0.395	0.92

Table 35: Results of CNN_2D_ResidualNetwork (LR = 10^{-5})

Year	Accuracy	Sensitivity	Specificity
First year	0.30	0.98	0.273
Second year	0.64	0.68	0.643
Third year	0.821	0.50	0.838
Fourth year	0.76	0.50	0.774
Fifth year	0.81	0.69	0.818
Total	0.85	0.56	0.86
Average	0.696	0.651	0.701

The results show the most effective learning rate for each CNN: i.e., learning rate 10^{-4} for CNN_1D, learning rate 10^{-3} for CNN_2D and learning rate 10^{-3} for CNN_2D_ResidualNetwork. As can be seen, the result of accuracy, sensitivity and specificity for the data of the first year is better than the data of other five data sets. One reason for this observation could be the ratio of bankrupted firms to non-bankrupted firms is minimal in data set one.

In the last step, we used 10_fold cross validation to evaluate our three DL models and LR model and calculate standard deviation and average of performance metrics. Note that because the system for implementing these approaches did not access to a GPU, the results of standard deviation and average of each CNN models are calculated for first year data only. This data set resulted the best outcomes among others. Table 36 provide a summary of 10-fold cross validation corresponding to the three DL models and LR model applied on the first year the Polish data set.

Table 36: Results for 10_fold cross validation

CNN	Average accuracy	Average sensitivity	Average specificity	Std_accuracy	Std_sensitivity	Std_specificity
CNN_2D_Residual Network	0.88	0.637	0.889	0.03	0.11	0.03
CNN_2D	0.919	0.45	0.938	0.01	0.13	0.01
CNN_1D	0.87	0.60	0.88	0.03	0.14	0.03
LR	0.595	0.399	0.59	0.05	0.09	0.05

As we can see in this table, the low standard deviation for accuracy shows low deviation from the mean for each fold which is an indicator of model robustness on this dataset. While, this rate is a little higher (~.1) for our sensitivity metric, knowing that dataset is heavily imbalanced and less than 3% of dataset are positive, this fluctuation is acceptable. We can conclude that the degree that our all ML/DL models performance change, when using test dataset versus training dataset, is minimal. By Comparing the result of 10-fold validation and the result observed from the train/test dataset, we also conclude that our test dataset in previous analysis represents a good distribution of the whole dataset and we can also rely on this result.

6. Discussion

In this thesis, we created a novel approach for bankruptcy prediction by the aid of DL models (CNN). Our analysis is based on the database from Polish companies in the 1st, 2nd, 3rd, 4th, and 5th year. This data set contains 43405 observations totally over 5 years. We designed several CNNs and trained and validated them by using this data set with different learning rates. We also conducted a 10_fold cross-validation for each model. Furthermore, we compare the results of our DL models including the proposed CNN models with the best learning rate, and ANN models with a standard ML technique (LR) in this section. We summarize the best results from every model in Table 37 and Table 38. In fact, Table 37 provides an overview of the performance of each model and highlights the best in terms of accuracy and sensitivity when the missing data was imputed, SMOTE technique was applied, and normalization was not used. Also, Table 38 gives us the best accuracy and sensitivity when the missing data was imputed, SMOTE technique was applied, and normalization was used. In fact, the important difference between the two tables is that we did not apply normalization technique for the first one, while we used normalization for the second one.

Table 37: Comparison the results of different models (without normalization, with imputing missing values, with SMOTE)

Year	**Accuracy**					**Sensitivity**				
	ANN	**CNN_1D**	**CNN_2D**	**CNN_2D_ Residual Network**	**LR**	**ANN**	**CNN_1D**	**CNN_2D**	**CNN_2D_ Residual Network**	**LR**
First year	0.96	0.89	0.93	0.93	0.65	0.61	0.71	0.61	0.37	0.69
Second year	0.96	0.89	0.918	0.89	0.58	0.44	0.48	0.394	0.41	0.69
Third year	0.92	0.81	0.898	0.87	0.64	0.46	0.63	0.43	0.49	0.75
Fourth year	0.92	0.71	0.88	0.679	0.735	0.45	0.73	0.42	0.65	0.66
Fifth year	0.94	0.89	0.918	0.889	0.79	0.59	0.59	0.36	0.58	0.83
Total	0.95	0.87	0.90	0.688	0.60	0.59	0.48	0.54	0.79	0.80

Table 38: Comparison the results of different models (with normalization, with imputing missing values, with SMOTE)

Year	**Accuracy**					**Sensitivity**				
	ANN	**CNN_1D**	**CNN_2D**	**CNN_2D_ Residual Network**	**LR**	**ANN**	**CNN_1D**	**CNN_2D**	**CNN_2D_ Residual Network**	**LR**
First year	0.95	0.97	0.44	0.62	0.70	0.65	0.1	0.78	0.08	0.76
Second year	0.93	0.76	0.154	0.97	0.71	0.51	0.38	0.90	0	0.84
Third year	0.92	0.88	0.856	0.93	0.70	0.51	0.48	0.27	0.07	0.78
Fourth year	0.90	0.77	0.746	0.65	0.7	0.54	0.47	0.61	0.22	0.66
Fifth year	0.94	0.91	0.906	0.73	0.79	0.67	0.43	0.31	0.77	0.78
Total	0.93	0.92	0.51	0.94	0.59	0.54	0.43	0.56	0.06	0.79

As shown in Table 37, the obtained accuracy of ANN models is higher than other four methods. Then, CNN_1D method achieved the best accuracy after ANN. In general, we observe that the accuracy for both models is almost the same for all data sets. However, the lowest accuracy is observed for the LR model. In terms of sensitivity, the CNN_1D achieved the highest sensitivity for the 1st year and 4th year in comparison to other methods, and the LR model produced the highest sensitivity for other data sets. In fact, the CNN_1D model is competitive with the LR method according to the sensitivity criterion. Also, it is evident that the accuracy and sensitivity of the LR model are in the same range. Furthermore, regarding sensitivity, the LR, ANN, CNN_1D, CNN_2D, CNN_2D_ResidualNetwork models have the sensitivity numbers ranging between 66% and 83%, 44% to 61%, 48% to 73%, 36% to 61%, 37% to 79%, respectively.

As it is clear from Table 38, the ANN model achieved the highest accuracy in comparison to other methods. Our results show that when we used normalization, the accuracy of all the models increased except the CNN_2D model. As for the sensitivity, the LR model achieved the largest one except for 1st and 2nd years, where the best results is obtained by CNN_2D model. In other words, the CNN_2D model holds the second place. In contrast, overall CNN_2D_ResidualNetwork has the smallest sensitivity for most data sets. This poor performance of CNN_2D_ResidualNetwork with normalized data may hint that concatenation of normalized data in our application might be suboptimal. Additionally, with respect to sensitivity, the LR, ANN, CNN_1D, CNN_2D, and CNN_2D_ResidualNetwork models have the sensitivity numbers ranging between 66% to 84%, 51% to 67%, 1% to 48%, 27% to 90%, and 0% to 77%, respectively. We can make a conclusion that the LR model achieved the best performance for the range of sensitivity compared to other methods, which is the main metric in this data set. Nevertheless, some of the developed DL models are quite competitive.

In the last step, we compared our best results with those obtained in other studies that considered the same datasets.

Table 39: Comparison of the obtained results in this study with the results of previous studies (accuracy)

Study	**features**	**classifier**	**performance**
Zięba et al	64	EXGB	1st year: 0.95 2nd year: 0.94 3rd year: 0.94 4th year: 0.94 5th year: 0.95
Lahmiri et al	64	AdaBoost	1st year: 0.946
Xiaomao et al	64	XGBG XGBG XGBSC XGBS XGBG	1st year: 0.95 2nd year: 0.93 3rd year: 0.93 4th year: 0.93 5th year: 0.95
Marso and El Merouani	64	BPNN ABCNN	3rd year: 81.05 5th year: 0.92
Keya et al	12	Bagging	1st year: 0.97
Marso and El Merouani	64	CSFNN	3rd year: 82.79 5th year: 90.30

Smiti and Soui	64	BSM-SAES	1st year: 0.96 2nd year: 0.96 3rd year: 0.950 4th year: 0.980 5th year: 0.96
This study	64	CNN_1D CNN_2D_ResidualNetwork CNN_2D_ResidualNetwork ANN ANN ANN	1st year: 0.97 2nd year: 0.97 3rd year: 0.93 4th year: 0.92 5th year: 0.94 Total: 0.95

As you see when we applied these different DL models, the accuracy is higher than the results of the previous studies in most data sets. So, the results of our proposed models are compatible with the previous contributions in the literature.

Table 40: Comparison of the obtained results in this study with the results of previous studies (sensitivity)

Study	**features**	**classifier**	**performance** (Sensitivity)
Aljawazneh et al	64	DBN (With SMOTE_ENN)	2nd year: 0.85
Aljawazneh et al	64	DBN (With SMOTE)	2nd year: 0.83
This study	64	CNN_2D	2nd year: 0.90

According to Table 40, we can compare our result with this paper in terms of sensitivity. In the first paper, the researcher used the SMOTE_ENN technique to balance the data, while we used SMOTE method. However, they achieved the very same results.

7. Conclusion

In this study, we created five Polish data sets, each for one year, in addition to the entire whole data set for bankruptcy prediction based on their financial ratios. This data sets include financial data for bankrupt companies from 2007 to 2013 and for non-bankrupt ones from 2010 to 2012 with 64 financial ratios. After preprocessing the data and using the SMOTE algorithm, we developed and applied the new DL models, convolutional neural network (CNN), including CNN_1D, CNN_2D, CNN_2D_ResidualNetwork, and ANN models on these six data sets. We compared the results of our proposed DL methods with the results of applying conventional ML methods such as logistic regression (LR) and the results of the previous work that used the same data sets.

By analyzing and comparing the extensive numerical results presented in the previous sections, we can make a conclusion that the plain LR method is not an efficient model for this data set mainly because it should be trained on an unbalanced data set for classification purposes. In fact, the inconsistent distribution of the bankrupted and non-bankrupted samples in the data set makes a substantial adverse effect on the model prediction and creates bias. It is necessary that we balance the data with some technique like SMOTE in the first step, and then apply the LR model. By doing this, the sensitivity of the model will increase significantly while its accuracy also will become reasonable.

In terms of developed DL models, as can be seen in the previous sections, we applied the ANN model and tuned the model with different hyperparameters. We also used three different CNN models including CNN_1D, CNN_2D and CNN_2D_ResidualNetwork with four various data preprocessing approaches and calculated accuracy, sensitivity, and specificity of each model. After conducting extensive experiments, we can conclude that by imputing the missing values and applying the SMOTE technique, without data normalization, the CNN models achieve better performance metrics. Afterwards, by examining three learning rates, we showed that the CNN models work better when the data is balanced and missing values are handled, and learning rate equals to 10^{-4} for CNN_1D, 10^{-3} for CNN_2D_ResidualNetwork and 10^{-3} for CNN_2D. To further assess the performance of these CNN models, we used a 10_fold cross validation approach for the data set, which produced the best results for the 1st year data. Furthermore, we calculated the standard deviation and average of performance metrics over all these 10-fold cross validations.

Finally, we summarize the novel findings of this thesis. In terms of the accuracy criterion, DL methods are dominant in almost all the data sets. Therefore, they may be the model of choice for this type of data sets. In terms of the sensitivity criterion, DL methods are competitive with traditional methods. In particular, we observed that normalizing the data set adversely impacted the sensitivity of deep learning methods while it slightly improved that of the traditional methods. One may argue that since the data set contains features that reflect financial ratios for which the magnitude of the feature itself might be informative, normalization may hurt feature selection, which is the backbone of DL methods. That is, because feature selection is mainly designed for the detection of invariance properties in the data set and financial data may lack such properties, normalizing the data might negatively impact the sensitivity. Also, although DL is typically a more suitable for image and voice processing, the results of this thesis show that applying CNN_1D method with proper settings can provide a significant improvement in terms of model sensitivity, which is a key performance metric, in analyzing the bankruptcy prediction.

There are also some limitations for the deep learning methods we used in this work, which may make them not applicable in other data sets and sectors. For example, the response variable that we have is binary, i.e., the firm is bankrupt or not. However, the response variable may be categorical for other data sets and our proposed methods do not directly apply. Also, the independent variables in our data sets are all numerical. If they become categorical or non-numerical, our proposed methods may not directly apply either.

Current study can be extended in the following directions. First of all, a major property of our data set is that it is imbalanced, and we used the SMOTE technique to make it balanced. We have several results indicating that applying this technique is crucial for improved results. However, there are many other techniques to make the data set balanced such as class weight technique. Therefore, a natural question is: What would be the results of other balancing techniques on the results? Secondly, we designed and applied three CNN architectures to our data set. However, one can design more sophisticated deep learning architectures by considering different convolutions and apply them to this data set. Nonetheless, our results show that a simple CNN_1D method outperforms more complex architectures. Third, and more importantly, it is interesting to consider an ensemble model, where we create many simple CNN models and report some measure, such as the average, over all these simple models. This is similar to a random forest approach where the results of simple decision trees are combined for the final reports. We expect that the ensemble approach will improve validation results and address potential overfitting. Fourth, we have analyzed the results of logistic regression as the benchmark. One may use other benchmarks such as support vector machines or random forest for classification purposes. Note that although some of these benchmarks have already been applied to this data set, a combination of data normalization, balancing techniques and so on will create new benchmarks. Fifth, the data set does not keep track of firms over time. If such a data becomes available, that is, we have the features of a firm over time and the final result of bankruptcy, we can apply time series methods which can potentially provide novel insights to this problem where the static model cannot.

References

Acharjya, D.P., Rathi, R., 2021. An extensive study of statistical, rough, and hybridized rough computing in bankruptcy prediction. Multimedia Tools and Applications 80, 35387–35413.

Alfaro, E., García, N., Gámez, M., Elizondo, D., 2008. Bankruptcy forecasting: An empirical comparison of AdaBoost and neural networks. Decision Support Systems 45, 110–122.

Aljawazneh, H., Mora, A.M., García-Sánchez, P., Castillo-Valdivieso, P.A., 2021. Comparing the performance of deep learning methods to predict companies' financial failure. IEEE Access 9, 97010–97038.

Altman, E.I., 1968. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. The journal of finance 23, 589–609.

Altman, E.I., Hotchkiss, E., 2010. Corporate Financial Distress and Bankruptcy: Predict and Avoid Bankruptcy, Analyze and Invest in Distressed Debt. John Wiley & Sons.

Atiya, A.F., 2001. Bankruptcy prediction for credit risk using neural networks: A survey and new results. IEEE Transactions on neural networks 12, 929–935.

Barboza, F., Kimura, H., Altman, E., 2017. Machine learning models and bankruptcy prediction. Expert Systems with Applications 83, 405–417.

Beaver, W.H., 1966. Financial ratios as predictors of failure. Journal of accounting research 71–111.

Bellovary, J.L., Giacomino, D.E., Akers, M.D., 2007. A review of bankruptcy prediction studies: 1930 to present. Journal of Financial education 1–42.

Bishop, C.M., Nasrabadi, N.M., 2006. Pattern recognition and machine learning. Springer.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., 2015. Xgboost: extreme gradient boosting. R package version 0.4-2 1, 1–4.

Ding, X., Zhang, Y., Liu, T., Duan, J., 2015. Deep learning for event-driven stock prediction, in: Twenty-Fourth International Joint Conference on Artificial Intelligence.

Djamal, E.C., Ramdlan, S.N., Saputra, J., 2013. Recognition of Handwriting Based on Signature and Digit of Character Using Multiple of Artificial Neural Networks in Personality Identification. ISICO 2013 2013.

Fedorova, E., Gilenko, E., Dovzhenko, S., 2013. Bankruptcy prediction for Russian companies: Application of combined classifiers. Expert systems with applications 40, 7285–7293.

Fernandez, A., Garcia, S., Herrera, F., Chawla, N.V., 2018. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. Journal of Artificial Intelligence Research 61, 863–905. https://doi.org/10.1613/jair.1.11192

García, V., Marqués, A.I., Sánchez, J.S., 2019. Exploring the synergetic effects of sample types on the performance of ensembles for credit risk and corporate bankruptcy prediction. Information Fusion 47, 88–101.

Jain, P., Tiwari, A.K., Som, T., 2021. Improving financial bankruptcy prediction using oversampling followed by fuzzy rough feature selection via evolutionary search, in: Computational Management. Springer, pp. 455–471.

Keya, M.S., Akter, H., Rahman, M.A., Rahman, M.M., Emon, M.U., Zulfiker, M.S., 2021. Comparison of Different Machine Learning Algorithms for Detecting Bankruptcy, in: 2021 6th International Conference on Inventive Computation Technologies (ICICT). IEEE, pp. 705–712.

Khoshgoftaar, T.M., Allen, E.B., 1999. Logistic regression modeling of software quality. Int. J. Rel. Qual. Saf. Eng. 06, 303–317. https://doi.org/10.1142/S0218539399000292

Lahmiri, S., Bekiros, S., Giakoumelou, A., Bezzina, F., 2020. Performance assessment of ensemble learning systems in financial data classification. Intelligent Systems in Accounting, Finance and Management 27, 3–9.

Liang, D., Lu, C.-C., Tsai, C.-F., Shih, G.-A., 2016. Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. European Journal of Operational Research 252, 561–572.

Liu, Z., Peng, C., Xiang, W., Tian, D., Deng, X., Zhao, M., 2010. Application of artificial neural networks in global climate change and ecological research: An overview. Chin. Sci. Bull. 55, 3853–3863. https://doi.org/10.1007/s11434-010-4183-3

Mahapatra, U., Nayak, S.M., Rout, M., 2020. A Systematic Approach to Enhance the Forecasting of Bankruptcy Data, in: Progress in Computing, Analytics and Networking. Springer, pp. 641–650.

Mai, F., Tian, S., Lee, C., Ma, L., 2019. Deep learning models for bankruptcy prediction using textual disclosures. European journal of operational research 274, 743–758.

Marso, S., El Merouani, M., 2020. Predicting financial distress using hybrid feedforward neural network with cuckoo search algorithm. Procedia Computer Science 170, 1134–1140.

Marso, S., EL Merouani, M., 2020. Bankruptcy Prediction using Hybrid Neural Networks with Artificial Bee Colony. Engineering Letters 28.

Matin, R., Hansen, Casper, Hansen, Christian, Mølgaard, P., 2019. Predicting distresses using deep learning of text segments in annual reports. Expert Systems with Applications 132, 199–208.

McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133. https://doi.org/10.1007/BF02478259

Merwin, C.L., 1942. Financing small corporations in five manufacturing industries, 1926-1936. National Bureau of Economic Research, New York.

Min, J.H., Lee, Y.-C., 2005. Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. Expert systems with applications 28, 603–614.

Ohlson, J.A., 1980. Financial ratios and the probabilistic prediction of bankruptcy. Journal of accounting research 109–131.

Pisula, T., 2020. An ensemble classifier-based scoring model for predicting bankruptcy of polish companies in the Podkarpackie Voivodeship. Journal of Risk and Financial Management 13, 37.

Pituch, K.A., Stevens, J.P., 2015. Applied Multivariate Statistics for the Social Sciences: Analyses with SAS and IBM‘s SPSS, 6th ed. Routledge, New York. https://doi.org/10.4324/9781315814919

Quynh, T.D., Phuong, T.T.L., 2020. Improving the bankruptcy prediction by combining some classification models, in: 2020 12th International Conference on Knowledge and Systems Engineering (KSE). IEEE, pp. 263–268.

Ren, Y., Weiss, G.M., 2021. A Comparison of Important Features for Predicting Polish and Chinese Corporate Bankruptcies, in: Advances in Data Science and Information Engineering. Springer, pp. 187–198.

Rosenblatt, F., 1961. PRINCIPLES OF NEURODYNAMICS. PERCEPTRONS AND THE THEORY OF BRAIN MECHANISMS. CORNELL AERONAUTICAL LAB INC BUFFALO NY.

Sangwan, K.S., Saxena, S., Kant, G., 2015. Optimization of Machining Parameters to Minimize Surface Roughness using Integrated ANN-GA Approach. Procedia CIRP, The 22nd CIRP Conference on Life Cycle Engineering 29, 305–310. https://doi.org/10.1016/j.procir.2015.02.002

Shah, J.R., Murtaza, M.B., 2000. A neural network based clustering procedure for bankruptcy prediction. American Business Review 18, 80.

Shahee, S.A., Ananthakumar, U., 2021. An overlap sensitive neural network for class imbalanced data. Data Mining and Knowledge Discovery 35, 1654–1687.

Shin, K.-S., Lee, T.S., Kim, H., 2005. An application of support vector machines in bankruptcy prediction model. Expert systems with applications 28, 127–135.

Shmueli, 2017. Data Mining for Business Analytics Concepts, Techniques, and Applications with XLMiner.

Singh, N.S., Hariharan, S., Gupta, M., 2020. Facial Recognition Using Deep Learning, in: Jain, V., Chaudhary, G., Taplamacioglu, M.C., Agarwal, M.S. (Eds.), Advances in Data Sciences, Security and Applications, Lecture Notes in Electrical Engineering. Springer, Singapore, pp. 375–382. https://doi.org/10.1007/978-981-15-0372-6_30

Smiti, S., Soui, M., 2020. Bankruptcy prediction using deep learning approach based on borderline SMOTE. Information Systems Frontiers 22, 1067–1083.

Sordo, M., n.d. Introduction to Neural Networks in Healthcare 18.

Soui, M., Smiti, S., Mkaouer, M.W., Ejbali, R., 2020. Bankruptcy prediction using stacked auto-encoders. Applied Artificial Intelligence 34, 80–100.

Strzelecka, A., Kurdyś-Kujawska, A., Zawadzka, D., 2020. Application of logistic regression models to assess household financial decisions regarding debt. Procedia Computer Science, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 24th International Conference KES2020 176, 3418–3427. https://doi.org/10.1016/j.procs.2020.09.055

Thorgren, S., Williams, T.A., 2020. Staying alive during an unfolding crisis: How SMEs ward off impending disaster. Journal of Business Venturing Insights 14, e00187. https://doi.org/10.1016/j.jbvi.2020.e00187

Tsai, C.-F., Wu, J.-W., 2008. Using neural network ensembles for bankruptcy prediction and credit scoring. Expert systems with applications 34, 2639–2649.

Tu, J.V., 1996. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. Journal of Clinical Epidemiology 49, 1225–1231. https://doi.org/10.1016/S0895-4356(96)00002-9

Vui, C.S., Soon, G.K., On, C.K., Alfred, R., Anthony, P., 2013. A review of stock market prediction with Artificial neural network (ANN), in: 2013 IEEE International Conference on Control System, Computing and Engineering. Presented at the 2013 IEEE International Conference on Control System, Computing and Engineering, pp. 477–482. https://doi.org/10.1109/ICCSCE.2013.6720012

Wong, T.C., Chan, H.K., Lacka, E., 2017. An ANN-based approach of interpreting user-generated comments from social media. Applied Soft Computing 52, 1169–1180. https://doi.org/10.1016/j.asoc.2016.09.011

Wood, E.H., 2006. The internal predictors of business performance in small firms: A logistic regression analysis. Journal of Small Business and Enterprise Development 13, 441–453. https://doi.org/10.1108/14626000610680299

Xiaomao, X., Xudong, Z., Yuanfang, W., 2019. A comparison of feature selection methodology for solving classification problems in finance, in: Journal of Physics: Conference Series. IOP Publishing, p. 012026.

Zhang, G., Hu, M.Y., Patuwo, B.E., Indro, D.C., 1999. Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. European journal of operational research 116, 16–32.

Zhang, M., Li, H., Xia, G., Zhao, W., Ren, S., Wang, C., 2018. Research on the Application of Deep Learning Target Detection of Engineering Vehicles in the Patrol and Inspection for Military Optical Cable Lines by UAV, in: 2018 11th International Symposium on Computational Intelligence and Design (ISCID). Presented at the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), pp. 97–101. https://doi.org/10.1109/ISCID.2018.00029

Zhang, W., Yang, D., Zhang, S., Ablanedo-Rosas, J.H., Wu, X., Lou, Y., 2021. A novel multi-stage ensemble model with enhanced outlier adaptation for credit scoring. Expert Systems with Applications 165, 113872.

Zhao, D., Huang, C., Wei, Y., Yu, F., Wang, M., Chen, H., 2017. An effective computational model for bankruptcy prediction using kernel extreme learning machine approach. Computational Economics 49, 325–341.

Zięba, M., Tomczak, S.K., Tomczak, J.M., 2016. Ensemble boosted trees with synthetic features generation in application to bankruptcy prediction. Expert systems with applications 58, 93–101.