**FACULTY OF SCIENCE, ENGINEERING AND COMPUTING**

**School of Computer Science and Mathematics**

**MSc in Data Science**

**Author**

Darpan Sanjay Agarwal - K2279745

**Project Title**

Company Bankruptcy Predictions

**Supervisor**

Prof. Nebel, Jean-Christophe A

**Kingston University London**

(Jan 2024)

**WARRANTY STATEMENT**

This is a student project. Therefore, neither the student nor Kingston University makes any warranty, express or implied, as to the accuracy of the data or conclusion of the work performed in the project and will not be held responsible for any consequences arising out of any inaccuracies or omissions therein.

# ABSTRACT

In the dynamic landscape of today's economic climate, accurately predicting an organisation's financial health is a critical concern for its survival and prosperity. Financial forecasting, particularly in the context of bankruptcy prediction, has become indispensable for decision-makers aiming to assess a company's solvency and its ability to meet financial obligations. In this research endeavour, we focus on leveraging cutting-edge machine learning techniques, specifically Deep Learning (DL) methods, to enhance the accuracy and reliability of bankruptcy predictions.

Our study is centred around the comparison of three widely utilized Deep Learning techniques: the Multilayer Perceptron model with six layers (MLP-6L), Long-Short Term Memory (LSTM), and the Deep Belief Network (DBN). These models have demonstrated efficacy in various classification tasks, making them suitable candidates for predicting financial distress.

Recognizing the challenge posed by imbalanced datasets, we intend to employ appropriate balancing techniques during our analysis to ensure more robust predictions. Imbalances in the dataset can skew the model's performance, leading to biased outcomes. By addressing this concern, we aim to improve the generalisation capability of our models and provide more reliable insights into a company's financial future.

To further enhance the robustness of our predictions, we plan to integrate three ensemble classifier techniques: K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Random Forest (RF). Ensemble methods combine the strengths of multiple models to achieve superior performance and mitigate individual model weaknesses. This approach adds an extra layer of sophistication to our analysis, allowing us to capture a more comprehensive understanding of financial dynamics.

As the field of machine learning continuously evolves, we are keen on exploring new methods, particularly those in the realm of Artificial Neural Networks (ANNs). Given their capacity to model complex relationships within data, ANNs offer a promising avenue for improving predictive accuracy in financial applications. We aim to scrutinize the performance of various ANN approaches and identify novel architectures that may outperform traditional methods in bankruptcy prediction.

One of the intriguing areas we delve into is hybrid algorithms, which amalgamate the strengths of different techniques to achieve enhanced predictive performance. Among these, the relatively newer MOA-PSO method has captured our attention. The MOA-PSO method combines the Metaheuristic Optimization Algorithm (MOA) with Particle Swarm Optimization (PSO) to offer a unique hybrid approach. We intend to implement and compare the MOA-PSO method against our established deep-learning techniques to assess their respective performances in the context of bankruptcy prediction.

Our research contributes not only to the refinement of financial forecasting methodologies but also to the growing body of knowledge on the application of machine learning in the finance domain. By conducting a comprehensive comparative analysis of established Deep Learning techniques and innovative hybrid algorithms, we aim to provide valuable insights that can guide decision-makers in assessing and mitigating financial risks effectively. As the financial landscape continues to evolve, the findings from this research will inform the development of more robust and adaptive models for anticipating the financial future of organizations.

Keywords: Bankruptcy Prediction, Deep Learning (DL), Machine Learning (ML), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest (RF), Multilayer Perceptron model with six layers (MLP-6L), Long-Short Term Memory (LSTM), Deep Belief Network (DBN), and Particle Swarm Optimization (PSO).

Contents

[ABSTRACT 2](#_Toc155694999)

[LIST OF FIGURES 5](#_Toc155695000)

[LIST OF TABLES 6](#_Toc155695001)

[ACKNOWLEDGEMENT 7](#_Toc155695002)

[CHAPTER 1 - INTRODUCTION 8](#_Toc155695003)

[1.1 BACKGROUND 8](#_Toc155695004)

[1.2 AIMS AND OBJECTIVE 9](#_Toc155695005)

[1.3 SUMMARY OF CONTRIBUTIONS 10](#_Toc155695006)

[1.4 THESIS OUTLINE 10](#_Toc155695007)

[1.5 ETHICS, AND DATA PROTECTION 11](#_Toc155695008)

[CHAPTER 2 – LITERATURE REVIEW 12](#_Toc155695009)

[2.1 RESEARCH QUESTIONS 12](#_Toc155695010)

[2.2 SCREENING PROCESS 12](#_Toc155695011)

[2.3 LITERATURE ANALYSIS 13](#_Toc155695012)

[2.4 Classification Algorithms Compared 16](#_Toc155695013)

[2.4.1. Deep Belief Network (DBN) 16](#_Toc155695014)

[2.4.2. Long-Short Term Memory (LSTM) 17](#_Toc155695015)

[2.4.3. Multilayer Perceptron With 6 Layers (MLP-6L) 17](#_Toc155695016)

[2.4.4. Random Forest (RF) 18](#_Toc155695017)

[2.4.5. Support Vector Machine (SVM) 19](#_Toc155695018)

[2.4.6. K-Nearest Neighbor (KNN) 20](#_Toc155695019)

[2.4.7. Magnetic Optimization Algorithm - Particle Swarm Optimization (MOA-PSO) 20](#_Toc155695020)

[2.4.8. Adaptive Boost (AdaBoost) 21](#_Toc155695021)

[CHAPTER 3 – CONTRIBUTION 22](#_Toc155695022)

[3.1 Dataset 22](#_Toc155695023)

[3.2. Experimental Setup 22](#_Toc155695024)

[3.3. Evaluation metrics 23](#_Toc155695025)

[3.4. Hyperparameters Considered 24](#_Toc155695026)

[3.5. Implementation 25](#_Toc155695027)

[3.5.1. MLP-6L (Benchmark model) 25](#_Toc155695028)

[3.5.2. K-Nearest Neighbor (KNN) 25](#_Toc155695029)

[3.5.3. Random Forest 28](#_Toc155695030)

[3.5.4 Simple Vector Mechanism 30](#_Toc155695031)

[3.5.5. Deep Belief Network 32](#_Toc155695032)

[3.5.6. Long-Short Term Memory (LSTM) 42](#_Toc155695033)

[3.5.7. Multilayer Perceptron With 6 Layers (MLP-6L) 46](#_Toc155695034)

[3.5.8. Particle Swarm Optimizer 49](#_Toc155695035)

[3.5.9. Overall Model Comparison. 51](#_Toc155695036)

[CHAPTER 4 – CONCLUSION 53](#_Toc155695037)

[4.1 Challenges Faced 53](#_Toc155695038)

[4.2 Conclusion 53](#_Toc155695039)

[4.3 Future Work 53](#_Toc155695040)

[REFERENCES 55](#_Toc155695041)

[APPENDIX 61](#_Toc155695042)

# LIST OF FIGURES

[Figure 1. Three Hidden Layer DBN 16](#_Toc155694496)

[Figure 2. LSTM Memory Cell 17](#_Toc155694497)

[Figure 3. Three Layer MLP 18](#_Toc155694498)

[Figure 4. Structure of Random Forest 19](#_Toc155694499)

[Figure 5. Company Bankruptcy Disparity 22](#_Toc155694500)

[Figure 6. Confusion Matrix 24](#_Toc155694501)

[Figure 7. F1-Score Vs No. of Neighbours. 26](#_Toc155694502)

[Figure 8. Random Forest Validation Dataset Confusion Matrix 29](#_Toc155694503)

[Figure 9. Random Forest Testing Dataset Confusion Matrix 30](#_Toc155694504)

[Figure 10. SVM RBF Validation Dataset Confusion Matrix 32](#_Toc155694505)

[Figure 11. SVM RBF Testing Dataset Confusion Matrix 32](#_Toc155694506)

[Figure 12. DBN Validation Dataset Confusion Matrix 42](#_Toc155694507)

[Figure 13. DBN Testing Dataset Confusion Matrix 42](#_Toc155694508)

[Figure 14. LSTM Validation Dataset Confusion Matrix 45](#_Toc155694509)

[Figure 15. DBN Testing Dataset Confusion Matrix 45](#_Toc155694510)

[Figure 16. MLP-6L Validation Dataset Confusion Matrix 48](#_Toc155694511)

[Figure 17. MLP-6L Testing Dataset Confusion Matrix 48](#_Toc155694512)

[Figure 18. PSO-NN Validation Dataset Confusion Matrix 50](#_Toc155694513)

[Figure 19. PSO-NN Testing Dataset Matrix 50](#_Toc155694514)

# LIST OF TABLES

[Table 1. Bench Mark Values (MLP-6L) 25](#_Toc155694527)

[Table 2. KNN Classifier Metrics 27](#_Toc155694528)

[Table 3. KNN Classifier Final Model Metrics 27](#_Toc155694529)

[Table 4. Random Forest Classifier Metrics 29](#_Toc155694530)

[Table 5. Random Forest Classifier Final Model Metrics 29](#_Toc155694531)

[Table 6. Simple Vector Machine Metrics 31](#_Toc155694532)

[Table 7. Simple Vector Machine Final Model Metrics 32](#_Toc155694533)

[Table 8. DBN\_RBM\_SMOTE Metrics 34](#_Toc155694534)

[Table 9. DBN\_RBM\_SMOTE final model Metrics 34](#_Toc155694535)

[Table 10. DBN\_RBM\_SMOTE\_ADABOOST Metrics 36](#_Toc155694536)

[Table 11. DBN\_RBM\_SMOTE\_ADABOOST Final Model Metrics 36](#_Toc155694537)

[Table 12. DBN\_RBM\_SMOTE\_ADABOOST\_RF Metrics 38](#_Toc155694538)

[Table 13. DBN\_RBM\_SMOTE\_ADABOOST\_RF Final Model Metrics 39](#_Toc155694539)

[Table 14. DBN\_KFOLD Metrics 40](#_Toc155694540)

[Table 15. DBN\_KFOLD Final model Metrics 41](#_Toc155694541)

[Table 16. DBN Best Model Metrics 42](#_Toc155694542)

[Table 17. LSTM Metrics 43](#_Toc155694543)

[Table 18. LSTM Final Model Metrics 45](#_Toc155694544)

[Table 19. MLP-6L Metrics 47](#_Toc155694545)

[Table 20. MLP-6L Final Model Metrics 48](#_Toc155694546)

[Table 21. PSO with Neural Network Metrics 50](#_Toc155694547)

[Table 22. All Best Model Metrics vs Bench Mark 51](#_Toc155694548)

# ACKNOWLEDGEMENT

I would like to express my profound gratitude to Prof. Jean-Christophe A Nebel for his immense knowledge, understanding, and patience, which greatly enhanced my experience during my graduate studies. His readiness to share his time generously has been deeply valued. I am immensely thankful for his exceptional guidance, enthusiastic support, and for creating an engaging research environment that nurtured my intellectual and professional growth.

My thanks also go to the faculty members of the School of Computer Science and Mathematics at Kingston University London. Their insightful feedback, encouragement, and thought-provoking questions encouraged me to broaden my research from multiple perspectives. Their knowledge and insights played a crucial role in guiding my research direction and in developing my problem-solving abilities.

I would also like to recognize the contributions of the technical and administrative staff in the Faculty of Science, Engineering, and Computing. Their assistance made my research journey smoother and more feasible.

Lastly, I extend my thanks to everyone who played a role, directly or indirectly, in the successful completion of this project. Even though I may not have mentioned all by name, their support and influence have been essential and greatly appreciated.

# CHAPTER 1 - INTRODUCTION

## 1.1 BACKGROUND

The issue of forecasting bankruptcy has garnered attention from researchers since the 1929 stock market crash (Bellovary, Giacomino and Akers, 2007). The repercussions of bankruptcy for a company carry significant weight, impacting a wide range of stakeholders, including employees, creditors, suppliers, and even entire nations. In the financial industry, researchers have been increasingly drawn to the field of technology such as machine learning (ML) and more particularly newer advancements in Deep Learning (DL). Increasingly, organizations are keen on acquiring this crucial analytical information. Nonetheless, the data concerning a company's financial health inherently exhibits an imbalance, as real-life bankruptcies are relatively rare as discussed by (Veganzones and Séverin, 2018). Numerous types of research have concentrated on rectifying the lack of patterns within smaller dataset classes, such as companies being bankrupt in our scenario, which significantly impairs classifier performance and reliability. This is because these methods tend to prioritize modeling the majority class. Consequently, a variety of techniques have been used to tackle this issue, each employing its own criteria for data balance. We have carefully considered the most suitable and relevant techniques and implemented them in our analysis of financial data.

This paper seeks to take a more advanced approach by utilizing sophisticated classification methods to enhance prior outcomes. To achieve this goal, we have explored Deep Learning (DL) techniques by (Schmidhuber, 2015) and upcoming artificial neural network models. (Hassani et al., 2024) describes DL, a subset of Machine Learning (ML), has demonstrated remarkable success in numerous applications, particularly those involving big data, surpassing the performance of traditional ML algorithms. We have opted for these advanced classification methods due to their established track record of delivering superior results when applied to financial data.

Hence, this research has examined several Deep Learning (DL) techniques, that includes Long-Short Term Memory (LSTM) described in brief by (Hochreiter and Schmidhuber, 1997), a Multilayer Perceptron model with 6 layers (MLP-6L) by authors (Ahmadpour Kasgari et al., 2012), and (Hinton, Osindero and Teh, 2006) describing Deep Belief Network (DBN) in very comprehensive detail, for the purpose of predicting corporate financial distress. Additionally, we have implemented three ensemble methods based on a technique called “Bagging” which is briefed by (Breiman, 1996), namely Random Forest (RF) (Breiman, 2001), K-Nearest Neighbor (KNN) by (Cover and Hart, 1967) and (Cortes and Vapnik, 1995) describe Support Vector Machine (SVM).

Hence, each of the chosen DL classifiers falls into a distinct category of neural networks, with the intention of comprehensively exploring in diverse manners. Specifically, MLP-6L represents a neural network that works in a feed-forward way, (Bagheri and Cheung, 2018) describes LSTM is categorized as a recurrent neural network, and Deep Believe Network is characterized as a neural network that learns greedily, consisting of directed and undirected layers. Conversely, to enhance the performance of SVM and KNN, we treat them as ensemble models employing the bagging technique. Meanwhile, RF functions as an ensemble of decision trees based on bagging.

Typically, balancing of data can be achieved through one of the following approaches: Oversampling, Clustering-based techniques, and Hybrid Oversampling-Undersampling. Consequently, the methods chosen for this research encompass all three of these data preparation strategies to assess their impact on the performance of classifiers. The objective is to ultimately determine the most effective 'DL/Data balancing' combination to tackle this financial challenge.

Furthermore, building upon research initiated by (Aljawazneh et al., 2021) regarding company bankruptcy prediction, we compare the effectiveness of different classification methods in predicting the financial status of Taiwanese companies. In this research, we utilize a classification approach called Metaheuristic Optimization-based Artificial Neural Network (MOAANN). This method is built upon the principles of the Particle Swarm Optimizer (PSO) which is widely discussed by (Ruba Abu Khurma et al., 2019)*.* Furthermore, author (Cheng and Jin, 2015) gives us an understanding of the technique known as Competitive Swarm Optimiser (CSO) which will help us to investigate cost sensitivity.

In the end, the effectiveness of the proposed methods cannot be adequately assessed solely by relying on the standard accuracy measure. This is especially true when dealing with extremely imbalanced data, as accuracy may not provide a reliable indication. In such cases, the minority class might consistently be misclassified, resulting in a very high accuracy score. Therefore, in addition to accuracy, we have incorporated other metrics such as precision, specificity, recall, to comprehensively evaluate the execution of all classification methods.

## 1.2 AIMS AND OBJECTIVE

The comprehensive aims and objectives of our project, which focuses on developing a predictive model to identify and forecast the likelihood of bankruptcy in companies and organizations, are as follows:

Primary Objective:

* Develop an Efficient Predictive Model:
  + Goal: Create a model that can accurately predict the probability of bankruptcy in companies and organizations.
  + Impact: Provide valuable insights to stakeholders such as investors and regulatory bodies, enabling them to make informed decisions and mitigate financial risks associated with bankruptcy.

Overview:

* Data Collection:
  + Data Sourcing: Collect relevant financial data from diverse sources.
  + Data Preparation: Cleanse and preprocess the data to ensure its quality and suitability for analysis.
  + Handling Imbalance: Use undersampling and oversampling techniques to balance the dataset and enhance its analytical stability.
* Feature Selection:
  + Identification of Key Variables: Determine crucial variables and features that strongly correlate with a company's financial health and bankruptcy risk.
* Model Development:
  + Selection of Techniques: Employ deep learning and statistical methods including Deep Belief Network, Multilayer Perceptron with six layers, and Long-Short Term Memory (LSTM).
  + Classifier Ensemble: (Hassanien, Chang and Mincong, 2021) Utilize K-Nearest Neighbor, Support Vector Machine (SVM), and Random Forest methods.
  + Innovative Approach: Develop the MOA-PSO method within the Artificial Neural Network framework to explore additional possibilities.
* Evaluation:
  + Model Assessment: Compare the performance of the developed models with previous related works using similar techniques.
  + Performance Metrics: Employ machine learning evaluation metrics such as F1-score, precision, accuracy, and recall to assess the models' efficacy.

Expected Outcomes:

* A robust predictive model that offers high accuracy in forecasting bankruptcy risks.
* Enhanced decision-making tools for stakeholders to proactively manage financial risks.
* Contribution to the field through innovative use of deep learning and statistical techniques in bankruptcy prediction.

This project is significant as it aims to leverage advanced data analytics and machine learning techniques to tackle a critical financial challenge. By accurately predicting bankruptcy risks, the project seeks to safeguard investments and guide regulatory practices, thereby contributing to overall economic stability and growth.

## 1.3 SUMMARY OF CONTRIBUTIONS

* This thesis provides a very comprehensive literature review regarding machine learning models applied to the “Taiwan Company Financial” Dataset.
* The thesis works to develop a series of predictive models using ensemble methods and deep-learning methods on our datasets such as KNN, SVM, RF, Deep Belief Network, MLP-6l and LSTM.
* The thesis also implements different practices such as data collection, cleaning and pre-processing practices. This ensures that a high-quality dataset is used for developing different machine-learning models.
* The study compares the model’s performance with the existing works, contributing to the body of knowledge on bankruptcy prediction.
* The thesis employs robust evaluation metrics which are F1 Score, precision, accuracy, recall, roc-auc curve and confusion matrices to ensure a comprehensive assessment of the model’s effectiveness.

## 1.4 THESIS OUTLINE

* Chapter 1 – Introduction
  + This section of our thesis provides a summary of our project and background of our topic.
  + It also helps to understand the problem statement, the methods for intended use and overall contributions.
* Chapter 2 – Literature Review
  + This section gives us the complete past and present contributions of different authors regarding the financial bankruptcy problem.
  + It helps us to understand the methods used by them and the limitations faced by them which led to different development of models.
* Chapter 3 – Contributions
  + This section represents the development of an advanced predictive model using deep learning and statistical techniques like Deep Belief Network, Multilayer Perceptron, LSTM, and MOA-PSO methods. This represents a significant advancement in the methodology of financial risk assessment.
  + Addressing data imbalance through under-sampling and oversampling techniques, contributing to the development of more robust and accurate predictive models in skewed datasets.
  + Identifying and utilizing key variables that correlate with bankruptcy risk. This approach enhances the model's relevance and accuracy in predicting financial distress.
  + Providing valuable insights to investors and regulatory bodies. The model aids in making informed decisions and in the proactive management of financial risks, thereby potentially reducing the incidence and impact of bankruptcies.
  + Comparing the model's performance with existing works, contributing to the body of knowledge on bankruptcy prediction.
  + The project's approach and methodologies have implications beyond financial risk assessment, potentially informing practices in other areas where predictive modelling is applicable.
* Chapter 4 – Conclusion
  + This section gives a complete overview of the outcomes of all the models developed, the comparisons of the results, the significant findings and suggestions for future research.

## 1.5 ETHICS, AND DATA PROTECTION

Our research utilizes the publicly available Taiwan Company dataset, a resource that has been extensively explored in various academic studies employing diverse machine learning techniques. The choice of this dataset is strategic, as it not only provides a rich source of information for our analysis but also ensures compliance with the stringent General Data Protection Regulation (GDPR) laws. By using a dataset that is freely available and has already been the subject of numerous studies, we align our work with global data privacy standards, reinforcing the ethical integrity of our research.

Central to our study is the application of this dataset in the realm of financial forecasting and bankruptcy prediction. We employ a range of machine learning methods, including advanced deep learning models, to analyse the data. This approach allows us to explore the potential of these techniques in providing accurate predictions of financial distress, a crucial aspect of the stability and growth of businesses.

Our research is grounded in ethical considerations. We are dedicated to using the data responsibly, to contribute positively to society. The insights gained from our analysis are intended to aid in better understanding and predicting financial trends, particularly in the context of bankruptcy. This could have significant implications for businesses and the broader economy, providing valuable guidance in financial decision-making processes.

In summary, our research, leveraging the Taiwan Company dataset, is focused on applying advanced machine learning techniques to predict financial distress. We ensure GDPR compliance, data security, and ethical use of information. Our findings, stored securely and temporarily on One-Drive, aim to contribute meaningfully to academic and practical knowledge in financial forecasting.

# CHAPTER 2 – LITERATURE REVIEW

## 2.1 RESEARCH QUESTIONS

To effectively guide this literature review, the following questions were considered

1. What are the current methods and techniques in Bankruptcy Prediction?
   1. Understanding the existing methodologies, models and techniques used in bankruptcy prediction which included both traditional models and modern machine learning approaches.
2. What Data Sources and variables have been commonly used?
   1. Investigated the types of financial data and specific variables that previous studies have utilized and summarised what data has been effective in the past models.
3. How have recent advances in AI and machine learning impacted the field?
   1. Explored how the integration of advanced AI and machine learning techniques, like deep learning, has influenced bankruptcy prediction models.
4. What are the common challenges and limitations faced in Bankruptcy Prediction Models?
   1. Identifies the typical challenges faced in developing accurate bankruptcy prediction models, such as data imbalance, feature selection complexities and model overfitting.
5. How are Model performance and effectiveness typically evaluated?
   1. Understand the standard metrics and methods used to evaluate the performance of bankruptcy prediction models. This includes accuracy, precision, recall, f1-score and other metrics.

## 2.2 SCREENING PROCESS

After carefully considering all the aspects I needed to cover for my literature review, I can provide a comprehensive summary of the steps I took and the key outcomes at each stage.

1. Establishing Inclusion and Exclusion Criteria
   1. My first step was to clear inclusion and exclusion criteria. I chose to include recent studies focusing on AI and Machine learning applications in bankruptcy prediction, particularly those discussing model development, feature selection and evaluation metrics.
2. Database and Keyword Selection
   1. I selected databases renowned for their quality research in finance and technology, such as IEEE Xplore, JSTOR, ScienceDirect, and Google Scholar.
   2. My keyword strategy was comprehensive, including terms like “bankruptcy prediction models”, “machine learning in financial risk”, and “predictive analytics in finance”. This ensured a focused yet extensive search.
3. Initial Screening
   1. I conducted an initial screening based on titles and abstracts. This step was crucial for quickly identifying the most relevant studies while filtering out those that didn't meet the basic criteria related to my project's aims.
4. Detailed Review
   1. For the papers that passed the initial screening, I carried out a more in-depth review. This involved scrutinizing their methodologies, data sources, model performances, and conclusions.
5. Assessment
   1. I assessed the quality of the selected studies, considering factors like the impact factor of the journals, the authors' expertise, and the robustness of their methodologies. This step ensured that the information I gathered was credible and scientifically sound.
6. Organizing and synthesizing information
   1. I organized the findings thematically to align with my project objectives, such as data handling techniques, model development, and performance evaluation. This thematic organization helped me create a coherent narrative for my literature review.
7. Documenting and updating the process
   1. Throughout the process, I kept a comprehensive record of all the sources I reviewed, the reasons for their inclusion or exclusion, and the insights they provided. I also established a procedure for regularly updating the literature review to incorporate new findings and trends in the field.

## 2.3 LITERATURE ANALYSIS

The prediction of financial failure is a matter of utmost importance and has been a focal point for numerous researchers. An incorrect assessment of a company's financial health can lead to significant financial losses. Traditionally, predicting a company's financial status has been accomplished through statistics. Many techniques such as Linear Discriminant Analysis (LDA) have been used repeatedly to research this topic. Another technique is called Multi-Discriminant Analysis (MDA). The most common technique is Logistic Regression (LR or Logit). Alternatively, in recent times Machine Learning (ML) has also been employed as shown in (Devi and Radhika, 2018). The authors used a meta-heuristic approach in which they combined SVM with PSO to achieve a high accuracy score (95%). The authors also suggested the use of optimization techniques to even further the accuracy. In the 1960s, (Altman, 1968) utilized MDA to forecast the financial health of an organization or a company based on their financial reports. They were able to predict the bankruptcy with 94% accuracy, however, the issue they faced was that all the data was for publicly held companies and they suggested that including smaller-sized companies and unlisted companies would increase the percentage of accuracy. Subsequently, (Ohlson, 1980) adopted the Logit model to predict corporate financial distress. He found that the prediction of the model is available on all the information in the market and suggested that additional parameters are required to enhance the model. (Brozyna, Mentel and Pisula, 2016) applied Linear Discriminant Analysis or LDA and Logistic regression or LR to forecast the financial condition of companies from Poland and Slovak. They found that although the method produces high results, the constant legal and economic conditions make the model redundant over time. (Jones and Hensher, 2004) introduced a variation of the Logistic regression model and compared it with a model that was standard for predicting financial anomaly, demonstrating that the new varied Logistic regression model yielded superior conclusions. More recently, several researchers have conducted comparative studies between statistical techniques and ML methods for predicting corporate financial disasters. For example, (Pompe and Feelders, 1997) made a comparison between neural networks and classification trees with the results of Linear Discriminant Analysis (LDA), concluding that neural networks outperformed other methods. The author however suggested that better hyperparameter tuning can be done to achieve higher results.(MIN and LEE, 2005)assessed the effectiveness of Simple Vector Machine and MDA with Simple vector machine delivering the most satisfactory results. They summarised that SVM was able to transform complex issues into smaller ones that can effectively identify linear functions. (Do Prado et al., 2019) talk Logistic regression, and neural networks in bankruptcy forecasting,

Furthermore, some researchers have integrated multiple ML algorithms to enhance the effectiveness of forecasting financial failure in companies. For example, (Fedorova, Gilenko and Dovzhenko, 2013) experimented with various methods one being a combination of Radial Basis Function networks with the technique known as MLP to predict the bankruptcy of companies based in Russia, employing a dataset that was balanced in nature containing 2906 entries chosen from the dataset complied. The authors were able to achieve 89% overall accuracy over the classical methods which gave 82% accuracy. The authors however suggested that since the laws keep changing the effect of those on bankruptcy also changes and should be considered in the models at regular intervals. (López Iturriaga and Sanz, 2015) used another method they merged Self-Organized Maps with MLP to forecast the failures of US banks up to 3 years in advance. Although their model gave a high accuracy result of 96.15%, they suggested that the dataset was built on banks which were caught in the mortgage bubble and a much-distributed dataset is required for building a reliable model. Similarly, (Lanbouri and Achchab, 2015) introduced a newer model that consisted of a Deep Belief Network and a Simple Vector Machine, the model was hybrid in nature and it was used to predict financial distress in French companies, using a dataset with 966 entries which was classified as balanced. The authors were able to achieve a precision of 70.1% Nevertheless, it's worth noting that these studies assessed the performance of their algorithm combinations using just a relatively small dataset.

Bankruptcy prediction datasets typically exhibit an imbalanced distribution, reflecting the fact that a very small percentage of any company/organization experiences bankruptcy in real-world scenarios. Consequently, it becomes essential to employ techniques that help us to make the dataset more balanced. SMOTE is one such technique and its variations have been widely utilized in various studies. For example, (Kim, Kang and Kim, 2015) incorporated SMOTE in conjunction with a boosting method known as Geometric Mean-based Boosting (GMBoost) algorithm, yielding highly favourable outcomes. However, the authors suggested that the method is really sensitive to outliers and it has a tendency to be less interpretable as compared to simpler models. Authors (Islam et al., 2019) used the SMOTE technique to rebalance a highly imbalanced dataset during preprocessing, resulting in performance improvements across 13 classification and regression algorithms. The authors suggested that since SMOTE is very well known for resampling minority classes, Generative Adversarial Networks should also be looked into for creating newer synthetic data. In another study by author (Zhou, 2013), SMOTE was combined with various traditional classification methods. The author examined the optimal scenarios for using 6 different techniques that used different balancing scenarios and highlighted the benefits of taking into consideration diverse datasets, such as companies from either America or Japan, to gain insights into how these methods perform in such contexts. The author concluded that the under sampling technique yielded far better results than the oversampling technique. The author however suggests that model selection should be based on the nature of the dataset.

Different variations of SMOTE have also been subject to comparison. In the study by (Le et al., 2018), the effect of various balancing techniques, including several SMOTE variants, was explored in the context of predicting the bankruptcy of Korean companies. Using (Le et al., 2018) Four classification models which are Random Forest, Decision tree, Multi-layer perceptron, and Simple vector machine) were employed to forecast financial status. The dataset in question exhibited extreme imbalance, prompting the evaluation of balancing techniques: the SMOTE and the ADASYN techniques. Additionally, the classifiers were tested on both the original and balanced datasets. Random Forest (RF) consistently performed superiorly to different models in both scenarios and yielded a score of 84.2%, with the best results achieved when RF was paired with SMOTE. However, the authors suggested that additional newly developed techniques should also be considered to solve the data imbalance issue. Consequently, given RF's superior performance in that study, we have also incorporated RF into our framework for predicting corporate financial failure, alongside other Deep Learning methods and different ensemble techniques, as mentioned earlier.

One of the driving factors behind our research is the utilization of DL algorithms as potent tools for forecasting corporate financial failure. There is a scarcity of studies that employ DL techniques with actual company data to predict financial distress.

(Jang et al., 2019) conducted a comparative analysis involving LSTM, Feed-forward neural network, and SVM for the prediction of business failure among listed US construction contractors. In a subsequent work by (Jang, Jeong and Cho, 2020), the same authors introduced a newer method that was based on LSTM to estimate the probability of company/organization failure within a specific timeframe, utilizing accounting data, market information, and other minor/macro-economic variables. Notably, both studies incorporated the SMOTE-Tomek balancing technique during data preprocessing, which yielded superior results compared to using only accounting variables.

Taking a unique approach, certain researchers have leveraged financial data in the form of graphical representations. For instance, (Yeh, Wang and Tsai, 2015) predicted the financial status of companies by employing Deep Belief Networks (DBN). They transformed the stock market returns of solvent and bankrupt companies into binary images, using these as training data for their models. Their findings demonstrated that the DBN method had an accuracy of 70% and outperformed the traditional SVM classification method which gave an output of 53%. However, the authors suggested more comprehensive techniques should be used to better the model performances. In a similar vein, (Hosaka, 2019) introduced a Convolutional Neural Networks (CNN) model to forecast company failure using grayscale representations. The data from the companies in Japan was used for this particular method. Remarkably, this novel approach yielded superior outcomes as compared to the other older methods and other Deep Learning techniques. However, the author suggested that it is difficult for the proposed model to accurately identify which financial ratio has a significant impact on bankruptcy and has admitted that the model is not entirely suitable for predicting bankruptcy.

With the advancements in computational power and data availability, researchers have turned to new machine-learning techniques. Artificial Neural Networks (ANNs) were among the early methods applied to bankruptcy prediction due to their ability to learn complex patterns from the data provided. Subsequent studies incorporated decision trees, support vector machines, and random forests, all of which demonstrated improved predictive performances compared to traditional methods. These methods have demonstrated better performance than gradient-based algorithms as shown in (Ansari et al., 2020). Additionally, the effects of MOA on imbalanced datasets were discussed by authors (Al-Badarneh et al., 2020), where a PSO algorithm was used as an optimizer for predicting bankruptcy in a neural network architecture. The authors were able to develop a hybrid MOA-PSO model which succeded in improving the results of MOA and PSO algorithms. However, they suggested that a newly developed method such as Functional Sized Population (FSMOA) could also be looked into for predictions.

We have examined a dataset comprising Taiwanese companies, encompassing 6819 entries collected over a decade from 1999 to 2009. Among these entries, 6599 companies (approximately 97%) are non-bankrupt, while 220 companies (roughly 3%) are labelled as bankrupt. The dataset exhibits significant class imbalance. It includes 95 financial health indicators and a single-class label indicating the bankruptcy status of each company. Detailed information about our dataset can be found in (Liang et al., 2016).

Bankruptcy prediction remains a critical task in financial analysis and decision-making. This literature review highlights the different bankruptcy prediction models and their approaches. The incorporation of non-financial data and the utilization of ensemble techniques have further advanced the predictive capabilities of these models. However, despite this significant progress, challenges persist, including imbalanced data and the interpretability of complex models. Our research efforts are focused on addressing these challenges to create more robust and practical bankruptcy prediction models that can better serve owners, investors, creditors, and stakeholders in the financial industry.

## 2.4 Classification Algorithms Compared

In this segment, we introduce three distinct advanced deep learning algorithms and three bagging ensemble methods that have demonstrated their efficacy in addressing classification tasks.

As previously detailed, we have chosen deep learning algorithms (namely, DBN, MLP-6L, and LSTM) to represent various neural network types. Meanwhile, the ensemble methods (RF, SVM, and KNN) have exhibited strong performance in classifying problems within existing literature.

### 2.4.1. Deep Belief Network (DBN)

Deep Belief Network, introduced by (Hinton, Osindero and Teh, 2006), is a probabilistic deep learning technique comprising multiple layers of Restricted Boltzmann Machines (RBMs). (Ruslan Salakhutdinov and Murray, 2008) describes Restricted Boltzmann Machines is a generative model that consists of two layers: visible and hidden. These layers have fully bidirectional connections with symmetric weights connecting them. In Figure, you can observe that a DBN is constructed by stacking several RBMs. In this arrangement, the layer that is hidden in the lower RBM serves as the layer that is visible in the upper RBM. The connections between these two layers lack directionality, while the connections between the layers that remain are directed. Moreover, Deep Belief Network adopts a greedy training approach, wherein each RBM undergoes unsupervised training sequentially. The result of every RBM which serves as the input for the RBM on the top is carefully tuned using Supervised Learning.

(image source - (Aljawazneh et al., 2021))

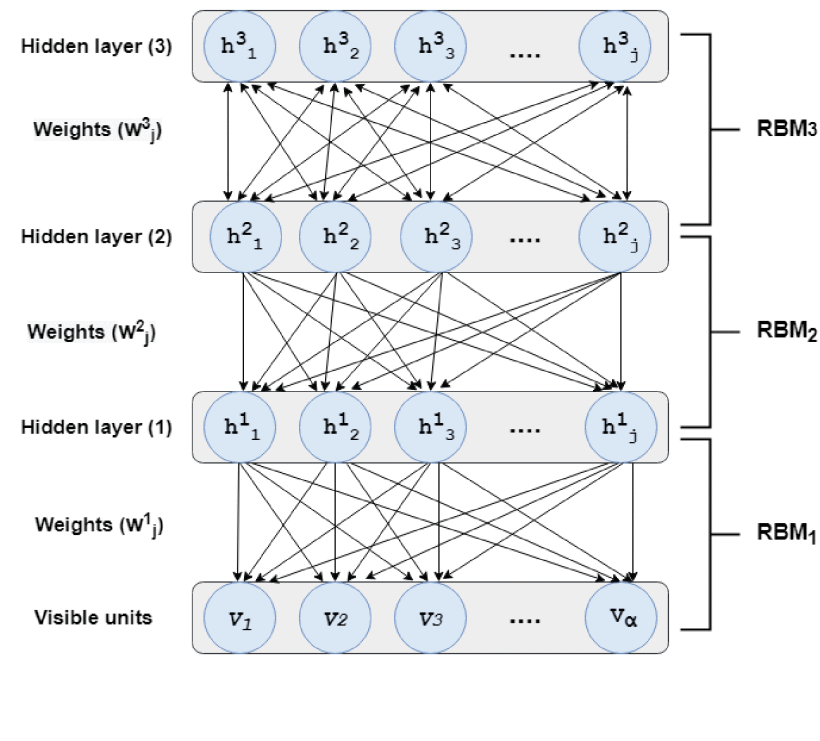


Figure 1. Three Hidden Layer DBN

Deep Belief Networks (DBNs) are proficient in feature learning and unsupervised learning and useful in complex data analysis like image and speech recognition. They benefit from layer-wise pre-training, enhancing learning efficiency. However, they are computationally intensive and challenging to train, sensitive to hyperparameter settings, and prone to overfitting. DBNs face limitations in scalability, and interpretability, and have been somewhat superseded by newer models like CNNs and RNNs. They're not well-suited for sequential data analysis, a drawback in fields like natural language processing.

### 2.4.2. Long-Short Term Memory (LSTM)

LSTM stands for Long Short-Term Memory, a distinctive variant within the family of Recurrent Neural Networks (RNNs). It was originally introduced by (Hochreiter and Schmidhuber, 1997) in their work. The fundamental building block of LSTM is a cell, which takes the place of the hidden layer neurons found in traditional RNNs. Each LSTM cell is primarily characterized by three crucial components or gates: the input, the output, and the forget gate described in (Paralic et al., 2023), illustrated in the provided diagram.

(image source - (Aljawazneh et al., 2021))

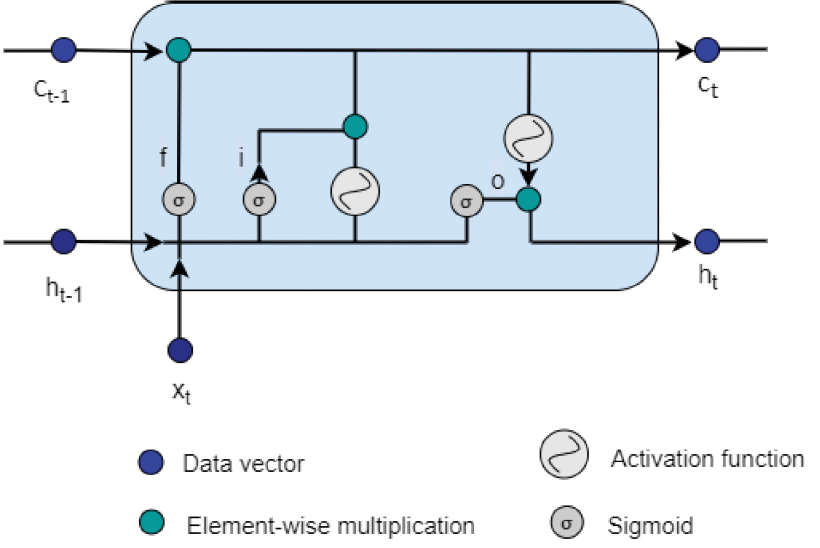


Figure 2. LSTM Memory Cell

(Pal, 2023) describes Long Short-Term Memory (LSTMs) networks are specialized in processing sequential data, making them ideal for tasks in natural language processing, speech recognition, and time series forecasting. (Przystalski and Thanki, 2024) address the vanishing gradient problem of traditional RNNs through their unique gating mechanism, enabling learning of long-term dependencies. However, LSTMs are computationally intensive, can be slow to train, require careful hyperparameter tuning, are prone to overfitting on smaller datasets, and have complex architectures that are challenging to interpret.

### 2.4.3. Multilayer Perceptron With 6 Layers (MLP-6L)

MLP, short for Multi-Layer Perceptron, is a method that is based on a feed-forward neural network, it is typically employed in supervised learning scenarios. It relies on back-propagation learning techniques, as outlined in (Ahmadpour Kasgari et al., 2012). (Aljawazneh, 2022) states that an MLP comprises a neural network structure with input and output layers, as well as one or more parallel layers that are hidden. The structure of the MLP is characterized by complete interconnections among its layers. However, when the number of layers that are hidden is increased, the MLP transitions from being a conventional learning approach to a deep learning method, as discussed in (Hatcher and Yu, 2018). In our research, we have utilized an MLP model featuring four hidden layers, resulting in a total of six layers in the network. Consequently, we refer to this configuration as MLP-6L.

(image source - (Aljawazneh et al., 2021))

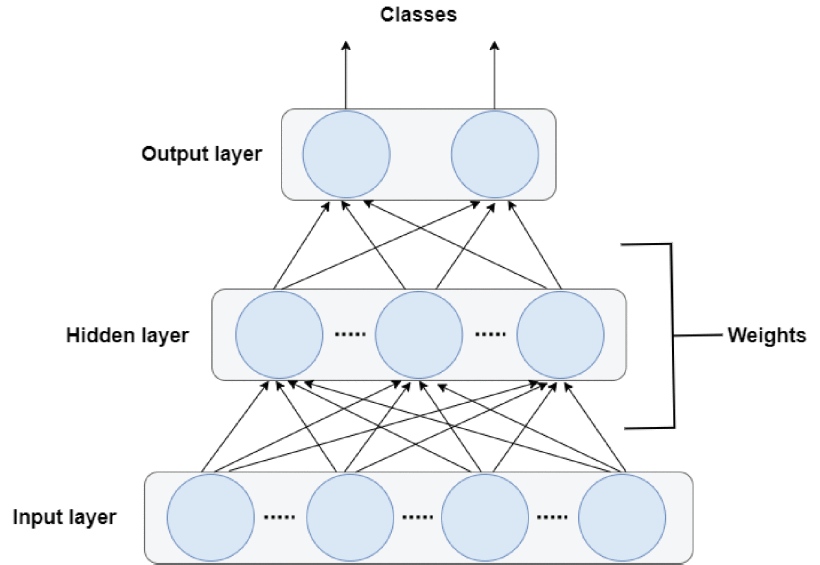


Figure 3. Three Layer MLP

In each layer, every processing unit establishes connections with all the units in the subsequent layer through weighted links, as described in (TSAI and WU, 2008). The input values serve as representations of the data that is passed forward through the network. The information after being processed within the units relies on both the input data and the weight assigned to each connection between input and hidden units. Likewise, the data generated by the output units is contingent on the values present in the units that are hidden and the weight assigned to each connection between hidden and output units, as detailed in (TSAI and WU, 2008)*.*

A Multilayer Perceptron with 6 Layers (MLP-6L) is a neural network known for its ability to recognize complex patterns, thanks to its deep, six-layer architecture (Aljawazneh, 2022). This makes it versatile and effective for various applications, particularly in structured data tasks like image and speech recognition. However, its training can be computationally intensive, particularly with large datasets, and it's sensitive to hyperparameter settings, requiring careful tuning to avoid issues like overfitting, especially on smaller datasets. While powerful in many contexts, MLP-6L has limitations: it's not inherently suited for sequential data analysis like time series, maybe outperformed by more specialized networks in certain tasks, and, like many deep learning models, it faces challenges in interpretability, making it difficult to understand its internal decision-making processes.

### 2.4.4. Random Forest (RF)

Random forest (RF), is a classification technique introduced by (Breiman, 2001), that operates by generating multiple decision trees from the initial dataset. Typically, these datasets are constructed using the bootstrapping method, and the individual trees are built employing the C4.5 algorithm, a well-established decision tree approach primarily rooted in concepts of entropy and information gain. The ultimate classification outcome produced by RF is determined through a majority vote among these constituent subtrees.

(image source - (Aljawazneh et al., 2021))

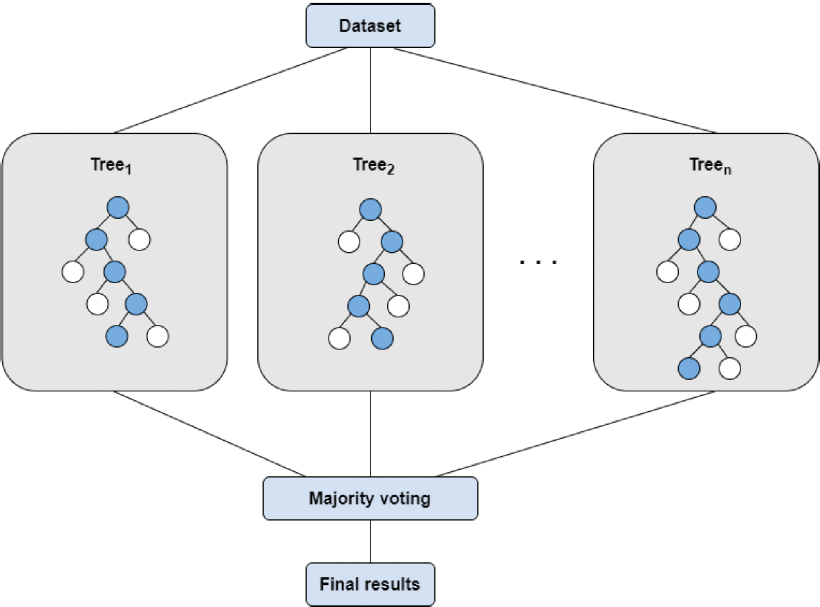


Figure 4. Structure of Random Forest

Random Forest is a robust machine-learning algorithm known for its high accuracy, versatility, and ease of use. (https://vegibit.com/, n.d.) describes it as an ensemble method, it combines multiple decision trees to produce more accurate and stable predictions, effectively reducing the risk of overfitting which is common in individual decision trees. Its strengths lie in its ability to handle both classification and regression tasks, deal with large datasets and high dimensional spaces, and its inherent feature selection capability, which improves model performance and interpretability. However, Random Forests are not without weaknesses and limitations. They can be computationally intensive, particularly with a large number of trees and deep tree structures, making them slower to train and predict compared to simpler models. Their model complexity can also lead to difficulties in interpretation, as understanding the collective decision-making process of numerous trees can be challenging. Moreover, Random Forests are less effective on very sparse data and don't perform well with very large datasets, where training can become prohibitively time-consuming.

### 2.4.5. Support Vector Machine (SVM)

SVM, short for Support Vector Machine, is a widely adopted supervised machine learning algorithm primarily designed for tackling binary classification and regression challenges. It was initially introduced by (Cortes and Vapnik, 1995). In essence, SVM seeks to identify an optimal separating hyperplane in the feature space that maximizes the margin between the two classes. Importantly, the data doesn't necessarily need to be linearly separable. (Altamirano-Flores et al., 2023)

To simplify complex calculations in such cases, SVM employs kernel functions (including Linear, Gaussian Radial Basis function (RBF), Sigmoid, and Polynomial (Altamirano-Flores et al., 2023)) as hyperparameters to determine the positioning of separating hyperplanes, as discussed in (Patle and Chouhan, 2013)*.*

Furthermore, in our study, we leverage SVM as an ensemble model and apply bagging (Breiman, 1996) to enhance its classification performance. This involves utilizing the bootstrapping technique to generate multiple subsets from the original dataset and independently implementing the method multiple times as shown in (Aljawazneh, 2022). The outcomes of the ensemble model are aggregated through majority voting.

Support Vector Machines (SVM) are a powerful classification method known for their accuracy and efficiency, particularly in high-dimensional spaces. SVMs excel in binary classification tasks and are effective in cases where the number of dimensions exceeds the number of samples. One of their key strengths is the use of kernel functions, allowing them to handle non-linear relationships by mapping data to higher-dimensional spaces. However, SVMs have their weaknesses and limitations. They are less effective with large datasets, as their training time can become prohibitively long. Additionally, they require careful tuning of parameters, such as the choice of kernel and regularization parameters, which can significantly impact their performance. (Pei et al., 2009) describes that SVMs also struggle with multi-class classification tasks and can be less intuitive to interpret compared to simpler models. Their performance heavily relies on the quality of the data; they are sensitive to noise and outliers, and may not perform well on datasets with overlapping classes.

### 2.4.6. K-Nearest Neighbor (KNN)

K-Nearest Neighbor (KNN), is another frequently employed non-parametric machine learning algorithm originally introduced by (Cover and Hart, 1967). Its principle involves determining the class label of each sample based on its similarity to its nearest neighbours. (Chomboon et al., 2015)state that a variety of metrics which includes Mahalanobis, Euclidian or Hamming can be used to find resemblance in the samples.

K-Nearest Neighbors (KNN) is a simple, yet effective machine-learning algorithm known for its ease of implementation and interpretability as described in (Ding and Eng, 2023). It works well for both classification and regression tasks, particularly in scenarios where the decision boundary is irregular. KNN is a non-parametric method, meaning it makes no underlying assumptions about the distribution of data, which adds to its flexibility. However, KNN has several weaknesses and limitations. (Pei et al., 2009) state that performance deteriorates with high-dimensional data due to the curse of dimensionality, as distances in higher-dimensional spaces become less meaningful. The algorithm is computationally intensive during the prediction phase because it involves calculating distances to all training samples. KNN also requires careful selection of the 'k' value (number of nearest neighbours), which can greatly impact its effectiveness. Moreover, (Azadeh Mokari, Guo and Bocklitz, 2023) state that it is sensitive to the scale of the data and irrelevant features, requiring thorough preprocessing like feature scaling and selection. KNN's simplicity can be a drawback in complex problems where more sophisticated models might be more effective.

### 2.4.7. Magnetic Optimization Algorithm - Particle Swarm Optimization (MOA-PSO)

The Magnetic Optimization Algorithm (MOA) is a relatively recent heuristic optimization technique developed by (Tayarani-N and Akbarzadeh-T, 2008). Research has demonstrated its effectiveness in tackling optimization problems, primarily those involving continuous real search spaces described in (Mirjalili and Sadiq, 2011). On the other hand, Particle Swarm Optimization (PSO) is a widely recognized metaheuristic optimization approach that has been extensively explored for optimizing a wide range of problems across various domains as discussed in (Nakisa et al., 2014). Initially proposed by (Eberhart and Kennedy, 1995), PSO has undergone several modifications over the years to enhance its performance.

In this study, we will work on the development and implementation of a neural network algorithm, named MOA-PSO. (Brenes, Johannssen and Chukhrova, 2022) describes Artificial Neural Networks (ANNs) in the context of bankruptcy prediction. (Ansari et al., 2020) states how MOA-PSO leverages the local search capabilities of MOA and the social thinking capabilities of PSO. Our research aims to demonstrate the capability of ANNs to the other DL and ensemble methods being used in our research.

The Magnetic Optimization Algorithm combined with Particle Swarm Optimization (MOA-PSO) is a powerful hybrid technique that excels in solving complex, multi-dimensional optimization problems. It effectively balances exploration and exploitation, leading to efficient convergence to optimal solutions. However, its performance heavily relies on precise parameter tuning, making it less accessible for non-experts. While generally robust, MOA-PSO can struggle with extremely complex or irregular objective functions and may be computationally demanding for very large-scale problems. Despite these limitations, MOA-PSO is a versatile tool for a wide range of optimization challenges.

### 2.4.8. Adaptive Boost (AdaBoost)

AdaBoost, short for Adaptive Boosting, is a transformative technique in machine learning, renowned for its ability to enhance the performance of algorithms. Developed by (Freund and Schapire, 1996), it primarily addresses classification problems by combining multiple weak learners to create a strong, accurate model. This technique operates on an iterative approach, adjusting the significance of different parts of the data based on the performance of successive models.

At the core of AdaBoost is the concept of weak and strong learners. As described by authors (Freund and Schapire, 1996) it is essentially a classifier that performs marginally better than random guessing. In AdaBoost, these are often simple decision trees, also known as decision stumps. The algorithm seeks to convert a collection of these weak learners into a strong learner – a classifier with significantly improved accuracy. This transformation is achieved through a process of iterative learning and weight adjustment. Initially, all training samples are assigned equal weights. However, as the algorithm iterates, it modifies these weights: increasing for samples that were misclassified and decreasing for those that were accurately predicted. This iterative refinement ensures that the algorithm increasingly focuses on the more challenging aspects of the dataset, leading to a model that effectively learns from its errors as proposed in (Kim and Kang, 2010).

Adaptive Boosting (AdaBoost) excels in improving the accuracy of weak classifiers, especially in binary classification, by focusing on misclassified instances in sequential training rounds. It is user-friendly with few parameters to tune and offers good generalization. However, AdaBoost is sensitive to noisy data and outliers, potentially leading to overfitting, and is less efficient due to its sequential nature. While effective for binary classification, it may struggle with regression or multi-class tasks without adaptations.

# CHAPTER 3 – CONTRIBUTION

## 3.1 Dataset

This dataset, covering Taiwanese companies from 1999 to 2009, consists of 6,819 entries, providing a detailed snapshot of the financial landscape over a significant period. This era is particularly notable for its varied economic conditions, making it a rich source for financial analysis.

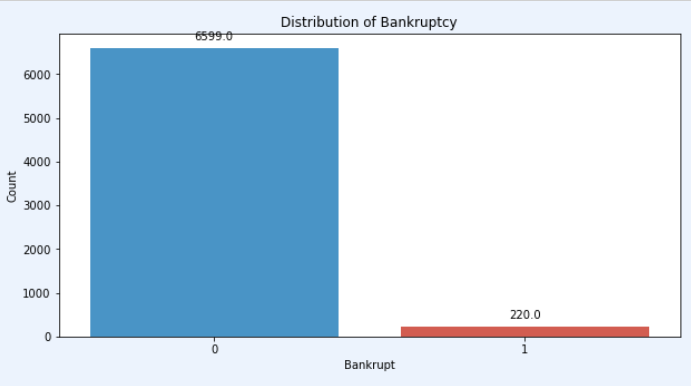


Figure 5. Company Bankruptcy Disparity

A striking feature of this dataset is the disparity in bankruptcy status among the companies. Out of the total, 97% (6,599) are non-bankrupt, while a mere 3% (220 companies) are bankrupt. This discrepancy points to a significant class imbalance, a crucial aspect for analytical approaches, particularly in predictive modelling where balance between classes is essential for accuracy.

The dataset is further enhanced by the inclusion of 95 distinct financial health indicators for each company. These indicators encompass a wide array of financial metrics, including profitability ratios, which measure a company's earnings about its revenues, expenses, and equity; leverage ratios, which assess the extent of a company's debt in comparison to its equity; liquidity ratios, indicating the company's ability to meet short-term obligations; and efficiency ratios, evaluating how effectively a company uses its assets and manages liabilities.

a comprehensive description is available in the scholarly work of *Liang, Lu, Tsai & Shih (2016)*., the rationale behind selecting financial indicators, and the impact of class imbalance.

## 3.2. Experimental Setup

We are exploring the prediction of corporate financial distress by framing it as a classification issue. To determine the most effective classifier for this challenge, we are assessing the efficacy of three deep learning (DL) algorithms – Deep Belief Networks (DBN), a six-layer Multi-Layer Perceptron (MLP-6L), and Long Short-Term Memory networks (LSTM) – against five established and highly regarded ensemble classification techniques, namely Random Forest (RF), Support Vector Machines (SVM), k-Nearest Neighbors (KNN), AdaBoost, and Extreme Gradient Boosting (XGBoost). Our goal is to identify which of these advanced computational methods demonstrates superior performance in accurately classifying companies at risk of financial failure.

In the experimental setup for our analysis, we utilized Python programming language in a Jupyter Notebook environment, leveraging several key libraries for data processing and machine learning (Abdelaziz Testas, 2023). Pandas and NumPy were indispensable for data manipulation and numerical calculations. Pandas, with its powerful data structures, enabled efficient handling of our dataset, facilitating tasks such as data cleaning, transformation, and aggregation. NumPy complemented this by providing support for complex mathematical operations, especially those involving multi-dimensional arrays.

For visual exploration of the data, we employed Seaborn and Matplotlib. Seaborn, known for its aesthetically pleasing and informative statistical graphics, was used to create insightful plots that helped in understanding underlying patterns and relationships in the data. Matplotlib offered further customization and flexibility in visualizing data, crucial for detailed analysis.

Our experimental setup included diverse machine-learning algorithms. We utilized RandomForestClassifier for decision tree-based ensemble learning, SVC for implementing support vector machines, and KNeighborsClassifier for neighbor-based classification. These methods were chosen for their efficacy in handling complex datasets and providing robust classification capabilities in varied scenarios.

In our study, we employed a range of advanced machine-learning techniques and tools. We used train\_test\_split for dataset partitioning, AdaBoostClassifier for ensemble learning, and BernoulliRBM within a neural network context. SMOTE addressed class imbalance, while TensorFlow and Keras enabled deep learning with LSTM layers. StandardScaler and performance metrics like ROC-AUC were crucial for preprocessing and evaluation.

The core of our experimental setup involved using scikit-learn, a robust machine-learning library. It provided tools for model building, validation, and evaluation. We utilized its functions to create classification models, applying them to our dataset. (Abdelaziz Testas, 2023) using the research from the author we will be using key metrics such as accuracy, precision, recall, and F1-score, calculated using Scikit-learn's metrics module, were crucial in assessing model performance. The confusion matrix, also generated through scikit-learn, provided a clear visualization of the model's performance, highlighting its ability to correctly or incorrectly classified instances. This comprehensive setup ensured a thorough and efficient analysis, allowing us to derive meaningful insights from our dataset.

We can find the libraries in Appendix 1.

## 3.3. Evaluation metrics

In evaluating the performance of machine learning or deep learning models, especially in the context of our imbalanced dataset, we prioritize metrics beyond mere accuracy due to its potential bias towards the majority class. Accuracy, while intuitive, is the ratio of correctly predicted observations (true positives and true negatives) to the total observations. It's simple but can be misleading for imbalanced datasets.

We focus on several other key metrics:

**Recall**:

* (Abdelaziz Testas, 2023) measures the model's ability to correctly identify all relevant instances (true positives) in the dataset. It is the ratio of true positives to the sum of true positives and false negatives, crucial in scenarios where missing a positive instance (false negative) carries a high cost.

**Precision**:

* Precision assesses the accuracy of positive predictions. (Ahmad et al., 2023) states that it is the ratio of true positives to all instances predicted as positive (true positives plus false positives). This metric is vital in situations where false positives are more undesirable than false negatives.

**F1-score**:

* (Ahmad et al., 2023) also describes that F1-score is the harmonic mean of precision and recall, considering both false positives and false negatives. It is useful for finding a balance between precision and recall, particularly valuable in the context of uneven class distribution.

**Confusion Matrix**: This is a tabular representation showing the model's performance, detailing the number of correct and incorrect predictions by each class. It includes true positives, true negatives, false positives, and false negatives, offering a comprehensive view of model accuracy.

(image source - (Chollet, 2020))

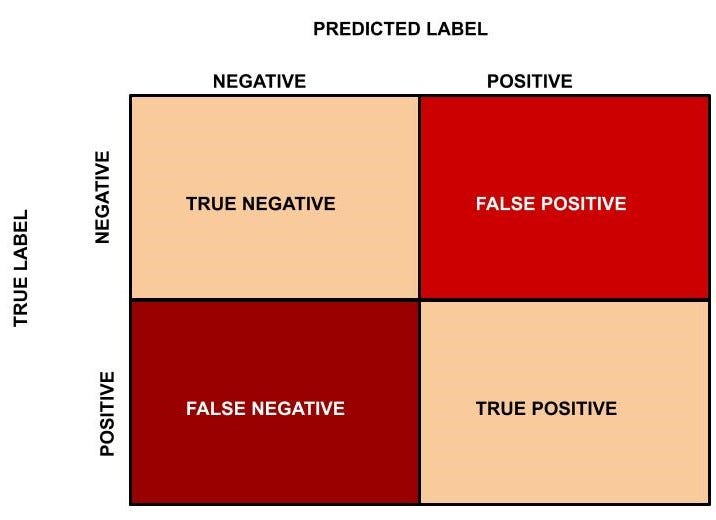


Figure 6. Confusion Matrix

## 3.4. Hyperparameters Considered

In our thesis on deep learning and ensemble methods, the performance critically hinges on the selection of appropriate hyperparameters, which vary depending on the specific problem. These hyperparameters are fine-tuned through experimentation. Key hyperparameters include:

* **Learning Rate:** Crucial in deep learning, it determines the model's adjustment in response to the error each time the weights are updated.
* **Epoch:** A full pass of the training data through the model.
* **Batch Size:** The number of data samples processed before updating the model's internal parameters.
* **Activation Functions:** Sigmoid (mapping input values between 0.0 and 1.0), Softmax (converting the output to a probability distribution), and ReLU (outputting the input directly if positive, else zero).
* **Dropout:** Reduces overfitting by randomly dropping out neurons.
* **Loss Function:** Categorical Cross-Entropy measures the difference between predicted probabilities and actual classes in classification (Asemi, 2023).
* **Optimizers:** Adam Optimizer adjusts both learning rate and model weights during training.
* **Kernel Function in SVMs:** Transforms data in classifiers, with types like Linear, Polynomial, Sigmoid, and RBF.
* **Ensemble Parameters:** Number of Estimators, Samples, and Features determine the composition and training of ensemble models.
* **K in KNN:** The count of nearest neighbours considered in KNN methods.
* **Entropy Criterion:** Measures data impurity in decision trees to guide splitting.

These hyperparameters, each with a specific role in the model's learning process, are chosen based on the data and task specifics. Not all hyperparameters are used for all models; their combinations vary as required by different models.

## 3.5. Implementation

For this thesis, as we have mentioned above, we will be looking into different ensemble (KNN, SVM, Random Forest) and Deep Learning (DBN, MLP-6L, LSTM) models in detail and how their design and implementation and how they perform on our dataset.

### 3.5.1. MLP-6L (Benchmark model)

(Aljawazneh et al., 2021) stated that the MLP-6L model gave them the best results. We will use this as our benchmark to make a comparison between our model performance and this one.

|  |  |
| --- | --- |
| **Metrics** | **Score** |
| Accuracy | 0.9963 |
| Recall | 1 |
| Precision | 0.9974 |

Table 1. Bench Mark Values (MLP-6L)

Accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined. An accuracy of 0.9963 is exceptionally high, suggesting that the MLP-6l model is very effective at classifying instances correctly.

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positives that are correctly identified by the model. A recall score of 1 is perfect, indicating that the model identified all positive cases without any false negatives.

Precision is the proportion of correct identifications. The precision score of 0.9974 is also outstandingly high, showing that when the MLP-6l model predicts a positive case, it is correct nearly all the time.

### 3.5.2. K-Nearest Neighbor (KNN)

**Design And Implementation**

We divided our dataset into training, testing and validation datasets with ratios of 70%, 15% and 15% respectively.

# dividing our dataset into a training dataset

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# dividing our dataset into testing and validation   
X\_val, X\_test, y\_val, y\_test = train\_test\_split(X\_temp, y\_temp, test\_size=0.5, random\_state=42)

To determine the optimal value of k, we developed a code using the F1-score parameters.

# Iterate through different values of K  
for k in k\_values:  
 knn\_classifier = KNeighborsClassifier(n\_neighbors=k)  
 knn\_classifier.fit(X\_resampled\_smote, y\_resampled\_smote)  
 y\_val\_pred\_smote = knn\_classifier.predict(X\_val\_smote)  
 f1\_value = metrics.f1\_score(y\_val\_smote, y\_val\_pred\_smote)  
 f1\_scores.append(f1\_value)

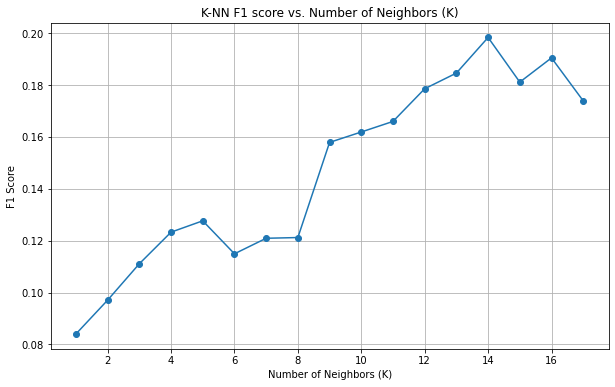


Figure 7. F1-Score Vs No. of Neighbours.

as we can observe 14 gives us the best f1-score and using this method we developed our model for the KNN method.

knn\_classifier = KNeighborsClassifier(n\_neighbors=k)

# Step 4: Train the classifier on the training data  
knn\_classifier.fit(X\_train, y\_train)

This was the base model we created. After this, we implemented a balancing technique known as SMOTE on our model. We tried 3 different sampling\_ratio which were 1, 0.5 and 0.25.

(Brownlee, 2020)

smote = SMOTE(sampling\_strategy=1, random\_state=42)

We also tried the ADASYN balancing technique on our KNN classification method.

(He et al., n.d.)

adasyn = ADASYN(sampling\_strategy='minority', random\_state=42)

After this we used another method called Bagging from (Classifier, n.d.) with smote to check its performance on the KNN classifier.

bagging\_knn = BaggingClassifier(base\_estimator=knn, n\_estimators=40, random\_state=42)

bagging\_knn.fit(X\_resampled\_smote, y\_resampled\_smote)

BaggingClassifier(base\_estimator=KNeighborsClassifier(n\_neighbors=25),n\_estimators=40, random\_state=42).

**Evaluation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Accuracy | Precision | Recall | F1-Score |
| Regular Model | Training | 0.94 | 0.12 | 0.08 | 0.1 |
| Testing | 0.94 | 0.12 | 0.08 | 0.09 |
| Smote = 1 | Training | 0.81 | 0.12 | 0.62 | 0.2 |
| Testing | 0.80 | 0.09 | 0.44 | 0.14 |
| Smote = 0.5 | Training | 0.86 | 0.14 | 0.51 | 0.22 |
| Testing | 0.86 | 0.09 | 0.31 | 0.14 |
| Smote = 0.25 | Training | 0.92 | 0.15 | 0.26 | 0.19 |
| Testing | 0.92 | 0.07 | 0.1 | 0.09 |
| Adasyn | Training | 0.92 | 0.15 | 0.26 | 0.09 |
| Testing | 0.92 | 0.07 | 0.1 | 0.19 |
| Bagging method with smote = 0.25 | Training | 0.91 | 0.13 | 0.23 | 0.17 |
| Testing | 0.92 | 0.08 | 0.1 | 0.09 |

Table 2. KNN Classifier Metrics

1. Regular Model: Shows high accuracy (0.94) but very low precision (0.12), recall (0.08-0.09), and F1-score (0.09-0.1). This indicates the model is correctly predicting the majority class but failing significantly at predicting the minority class, a common issue in imbalanced datasets.

2. SMOTE (with different ratios): As the SMOTE ratio decreases (from 1 to 0.25), there's a general trend of increased accuracy but reduced recall.

* Smote = 1: Balanced accuracy and recall, but low precision and moderate F1-score.
* Smote = 0.5: Better accuracy, slightly improved precision, reduced recall, and the best F1-score among the models.
* Smote = 0.25: Further improved accuracy, best precision among SMOTE models, but lower recall and F1-score.

3. ADASYN: Similar to the regular model in terms of metrics, suggesting that ADASYN didn't significantly change the model's ability to predict the minority class.

4. Bagging with Smote = 0.25: This method seems to be a compromise between accuracy and the ability to predict the minority class (recall). It has decent accuracy, but like most other methods here, it struggles with precision and F1-score.

**CONCLUSION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Accuracy | Precision | Recall | F1-Score |
| Smote = 0.5 | Training | 0.86 | 0.14 | 0.51 | 0.22 |
| Testing | 0.86 | 0.09 | 0.31 | 0.14 |

Table 3. KNN Classifier Final Model Metrics

Considering these factors, the SMOTE with a ratio of 0.5 appears to be the most balanced approach. It offers a reasonable compromise between accuracy and recall, with the highest F1-score, suggesting a better balance between precision and recall compared to other methods.

### 3.5.3. Random Forest

**Design and Implementation**

We split our dataset into training, testing and validation into 80%, 10% and 10% respectively.

After splitting our dataset, we use the pre-trained model of the Random Forest classifier to see the performance of our dataset.

(Scikit-learn-Random-Forest, 2018)

clf = RandomForestClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

After our model, we employ the technique of SMOTE balancing with the conjecture of the Random Forest classifier to check its performance.

We use the sampling ratio of 1, 0.5 and 0.25 and evaluate its performance.

After this, we tried using the AdaBoost Classifier with the Random Forest classifier.

clf = RandomForestClassifier(n\_estimators=100,criterion = "entropy", max\_depth= 8, random\_state=42)

(Adaboost, n.d.)

ada\_classifier = AdaBoostClassifier(base\_estimator=clf,

n\_estimators=50, learning\_rate=1, random\_state=42)

ada\_classifier.fit(X\_train\_resampled, y\_train\_resampled)

#we fit our trained model into our evaluation metrics.

(Huilgol, 2020)

val\_accuracy = accuracy\_score(y\_val, y\_val\_pred)  
precision = precision\_score(y\_val, y\_val\_pred)  
recall = recall\_score(y\_val, y\_val\_pred)  
f1\_value = f1\_score(y\_val, y\_val\_pred)

**Evaluation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Accuracy | Precision | Recall | F1-Score |
| Regular Model | Training | 0.97 | 0.67 | 0.1 | 0.18 |
| Testing | 0.96 | 0.875 | 0.23 | 0.35 |
| Smote = 1 | Training | 0.96 | 0.375 | 0.6 | 0.46 |
| Testing | 0.95 | 0.43 | 0.52 | 0.48 |
| Smote = 0.5 | Training | 0.96 | 0.4 | 0.55 | 0.47 |
| Testing | 0.95 | 0.49 | 0.51 | 0.5 |
| Smote = 0.25 | Training | 0.98 | 0.58 | 0.55 | 0.56 |
| Testing | 0.95 | 0.5 | 0.39 | 0.44 |
| Adaboost + Smote = 1 | Training | 0.97 | 0.46 | 0.7 | 0.55 |
| Testing | 0.95 | 0.47 | 0.48 | 0.48 |
| Adaboost + Smote = 0.5 | Training | 0.97 | 0.46 | 0.7 | 0.55 |
| Testing | 0.95 | 0.49 | 0.52 | 0.5 |
| Adaboost + Smote = 0.25 | Training | 0.98 | 0.55 | 0.6 | 0.58 |
| Testing | 0.96 | 0.56 | 0.49 | 0.52 |

Table 4. Random Forest Classifier Metrics

1. Regular Model (No SMOTE, No Adaboost):

- Training: High accuracy (0.97) but low precision (0.67), very low recall (0.1), and low F1-score (0.18). Indicates overfitting to training data with poor generalization to the minority class.

- Testing: Similar pattern as training but with slightly better recall and F1-score.

2. SMOTE = 1 (100% Over-sampling of Minority Class):

- Training: Slight drop in accuracy, significantly improved recall (0.6), resulting in a better F1-score (0.46). Shows better handling of the minority class.

- Testing: Similar trend as training with a balanced recall and precision, leading to an improved F1-score (0.48) compared to the regular model.

3. SMOTE = 0.5 (50% Over-sampling):

- Training: Similar accuracy to SMOTE=1, balanced precision and recall, leading to a slightly improved F1-score (0.47).

- Testing: Further improvement in both precision and recall compared to SMOTE=1, yielding the highest F1-score (0.5) among the SMOTE variations.

4. SMOTE = 0.25 (25% Over-sampling):

- Training: Highest accuracy (0.98) among SMOTE models, decent balance between precision and recall, with a good F1-score (0.56).

- Testing: Drop in recall compared to SMOTE=0.5, but better precision, resulting in a slightly lower F1-score (0.44).

5. Adaboost + SMOTE (Boosting Technique Combined with SMOTE):

- For all SMOTE ratios (1, 0.5, 0.25), Adaboost improves the recall in training significantly (around 0.7) while maintaining similar accuracy and precision levels, leading to higher F1-scores (0.55, 0.55, 0.58).

- In testing, the Adaboost+SMOTE models show a consistent pattern of improved balance between precision and recall compared to the regular model, resulting in better F1 scores.

**CONCLUSION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Accuracy | Precision | Recall | F1-Score |
| Adaboost + Smote = 0.5 | Training | 0.97 | 0.46 | 0.7 | 0.55 |
| Testing | 0.95 | 0.49 | 0.52 | 0.5 |

Table 5. Random Forest Classifier Final Model Metrics

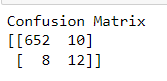


Figure 8. Random Forest Validation Dataset Confusion Matrix

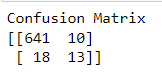


Figure 9. Random Forest Testing Dataset Confusion Matrix

Considering both the detailed performance metrics and the comparison with the benchmark, the best model among those tested appears to be the combination of Adaboost with SMOTE at a ratio of 0.5. This model achieves a good balance between precision and recall, both in the training and testing phases, with the highest F1-score observed in testing (0.5). While it still falls short of the near-perfect benchmark model in terms of precision and recall, it represents the most effective approach among the tested models for balancing the trade-off between identifying positive instances (recall) and avoiding false positives (precision). Further improvements should focus on enhancing both precision and recall to better approximate the benchmark's performance.

### 3.5.4 Simple Vector Mechanism

**Design and Implementation**

We split our dataset into training, testing and validation into 80%, 10% and 10% respectively.

After splitting our dataset, we use the pre-trained model of the Simple Vector Machine to see the performance of our dataset.

We tried different kernel methods which are polynomial, Radial Basis Function (RBF), Sigmoid and Linear (scikit learn SVM, 2018).

#kernel = Polynomial

svm\_classifier\_poly = SVC(kernel='poly', random\_state=42)

svm\_classifier\_poly.fit(X\_train, y\_train)

#kernel = Radial Basis Function

svm\_classifier\_rbf = SVC(kernel='rbf', random\_state=42)

svm\_classifier\_rbf.fit(X\_train, y\_train)

#kernel = Sigmoid

svm\_classifier\_sigmoid = SVC(kernel='sigmoid', random\_state=42)

svm\_classifier\_sigmoid.fit(X\_train, y\_train)

#kernel = Linear

svm\_classifier\_linear = SVC(kernel='linear', random\_state=42)

Our Linear kernel unfortunately did not yield any results but Poly, RBF and Sigmoid gave us results.

After this, I applied the smote balancing technique to our SVM models and evaluated our performance.

**Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kernel |  |  | Accuracy | Precision | Recall | F1-Score |
| Polynomial | Regular Model | Training | 0.97 | 0 | 0 | 0 |
| Testing | 0.95 | 0 | 0 | 0 |
| Smote = 1 | Training | 0.63 | 0.05 | 0.7 | 0.1 |
| Testing | 0.65 | 0.07 | 0.59 | 0.13 |
| Smote = 0.5 | Training | 0.86 | 0.06 | 0.3 | 0.11 |
| Testing | 0.87 | 0.12 | 0.05 | 0.17 |
| Smote = 0.25 | Training | 0.96 | 0.14 | 0 | 0.07 |
| Testing | 0.95 | 0 | 0 | 0 |
| Radial Basis Function (RBF) | Regular Model | Training | 0.97 | 0 | 0 | 0 |
| Testing | 0.95 | 0 | 0 | 0 |
| Smote = 1 | Training | 0.76 | 0.08 | 0.7 | 0.15 |
| Testing | 0.75 | 0.08 | 0.45 | 0.15 |
| Smote = 0.5 | Training | 0.87 | 0.12 | 0.6 | 0.2 |
| Testing | 0.87 | 0.1 | 0.25 | 0.15 |
| Smote = 0.25 | Training | 0.97 | 0 | 0 | 0 |
| Testing | 0.95 | 0 | 0 | 0 |
| Sigmoid | Regular Model | Training | 0.97 | 0 | 0 | 0 |
| Testing | 0.98 | 0 | 0 | 0 |
| Smote = 1 | Training | 0.40 | 0.02 | 0.45 | 0.04 |
| Testing | 0.42 | 0.03 | 0.38 | 0.05 |
| Smote = 0.5 | Training | 0.59 | 0.02 | 0.3 | 0.04 |
| Testing | 0.62 | 0.04 | 0.32 | 0.07 |
| Smote = 0.25 | Training | 0.93 | 0 | 0 | 0 |
| Testing | 0.94 | 0 | 0 | 0 |
| Linear | The model did not run | | | | |  |

Table 6. Simple Vector Machine Metrics

1. Polynomial Kernel:

- Regular Model: High accuracy (0.97 training, 0.95 testing) but zero precision, recall, and F1-score, indicating it's not predicting the minority class at all.

- Smote = 1: Moderate accuracy (0.63 training, 0.65 testing) with some improvement in precision, recall, and F1-score, but still low.

- Smote = 0.5: Better balance between accuracy (0.86 training, 0.87 testing) and F1-score (0.11 training, 0.17 testing).

- Smote = 0.25: High accuracy but low F1-score.

2. Radial Basis Function (RBF) Kernel

- Similar trends to the Polynomial kernel. The Smote = 0.5 setting shows a relatively better balance with an F1-score of 0.2 in training and 0.15 in testing.

3. Sigmoid Kernel:

- Regular Model: Very high accuracy but zero precision, recall, and F1-score.

- Smote = 1: Low accuracy and very low F1-score.

- Smote = 0.5: Moderate accuracy and F1-score.

- Smote = 0.25: High accuracy but zero precision, recall, and F1-score.

4. Linear Kernel: Did not run, so cannot be evaluated.

**Conclusion**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Kernel |  |  | Accuracy | Precision | Recall | F1-Score |
| Radial Basis Function (RBF) | Smote = 0.5 | Training | 0.87 | 0.12 | 0.6 | 0.2 |
| Testing | 0.87 | 0.1 | 0.25 | 0.15 |

Table 7. Simple Vector Machine Final Model Metrics

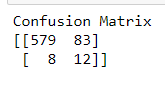


Figure 10. SVM RBF Validation Dataset Confusion Matrix

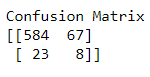


Figure 11. SVM RBF Testing Dataset Confusion Matrix

Considering the balance across all metrics, the RBF kernel with Smote = 0.5 seems to be the best model among the ones you tested. It offers a good compromise between accuracy (0.87 for both training and testing) and F1-score (0.2 in training, 0.15 in testing), indicating a more balanced performance across both classes compared to other models. However, there's still room for improvement, especially in precision and recall.

### 3.5.5. Deep Belief Network

To build our Deep learning model of the Deep Belief Network, we have tried a multiple combination to get the best optimal results, they are DBN + RBM + SMOTE, DBN + RBM + SMOTE + ADABOOST, DBN + RBM + SMOTE + ADABOOST + RANDOM FOREST, DBN + KFOLD. We will discuss briefly about them

**3.5.5.1. DBN + RBM + SMOTE**

**Design and Implementation**

We split our dataset into training, testing and validation into 80%, 10% and 10% respectively. We used the smote balancing ratio technique with our DBN model with varying sampling ratios of 1, 0.5 and 0.25.

We tested the models using different epochs which range from 5 to 50 while maintaining a consistent batch size of 32 and a number of components at 95.

We transformed our data into Scaler to ensure it works perfectly in our Deep learning model. Then we built our RBM model and created our deep belief network. I selected 'adam' as the optimizer and 'binary\_crossentropy' for the loss function due to their efficiency in handling binary classification tasks. I monitored 'accuracy' as a performance metric to evaluate the model's predictions. The best model which gave us the most balanced results is as follows.

(Brownlee, 2020)

smote = SMOTE(sampling\_strategy=0.5, random\_state=42)  
X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)

# Standardize the data (Scikit-Learn- StandardScaler, 2019)  
scaler = StandardScaler()  
X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled)  
X\_val = scaler.transform(X\_val)  
X\_test = scaler.transform(X\_test)

(Analytics Vidhya, 2022)

rbm = BernoulliRBM(n\_components=95, n\_iter=10, random\_state=42)  
rbm.fit(X\_train\_resampled)

BernoulliRBM(n\_components=95, random\_state=42)

# Transform the data using the trained RBM   
X\_train\_transformed = rbm.transform(X\_train\_resampled)  
X\_val\_transformed = rbm.transform(X\_val)  
X\_test\_transformed = rbm.transform(X\_test)

(Chollet, 2020)

model = tf.keras.Sequential([  
 tf.keras.layers.Input(shape=(X\_train\_transformed.shape[1],)),  
 tf.keras.layers.Dense(64, activation='relu'),  
 tf.keras.layers.Dropout(0.2),  
 tf.keras.layers.Dense(32, activation='relu'),  
 tf.keras.layers.Dropout(0.2),  
 tf.keras.layers.Dense(1, activation='sigmoid')   
])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.fit(X\_train\_transformed, y\_train\_resampled, epochs=25, batch\_size=32, verbose=2)

**Evaluation and Conclusion**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Model | Parameters | |  | Accuracy | Precision | Recall | F1-Score |
| Regular model | Epoch = 5  batch\_size = 32  n\_components = 95  n\_iters = 10 | Optimizer = adam  Loss = binary cross entropy  metrics = accuracy | Validation | 0.97 | 0 | 0 | 0 |
| Testing | 0.95 | 0 | 0 | 0 |
| Smote = 1 | Epoch = 50  batch\_size = 32  n\_components = 95  n\_iters = 10 | Validation | 0.8 | 0.125 | 0.9 | 0.22 |
| Testing | 0.8 | 0.18 | 0.9 | 0.3 |
| Smote = 0.5 | Epoch = 25  batch\_size = 32  n\_components = 95  n\_iters = 10 | Validation | 0.85 | 0.16 | 0.9 | 0.27 |
| Testing | 0.85 | 0.23 | 0.97 | 0.36 |
| Smote = 0.25 | Epoch = 5  batch\_size = 32  n\_components = 95  n\_iters = 5 | Validation | 0.97 | 0 | 0 | 0 |
| Testing | 0.95 | 0 | 0 | 0 |

Table 8. DBN\_RBM\_SMOTE Metrics

1. Accuracy: High accuracy (0.95-0.97) in both validation and testing without SMOTE and with SMOTE = 0.25. However, with SMOTE = 1 and 0.5, accuracy drops significantly (0.8-0.85), indicating potential overfitting or the model's struggle to generalize with heavily synthesized data.

2. Precision and Recall: Without SMOTE and with SMOTE = 0.25, the model has zero precision and recall, suggesting it fails to correctly identify positive cases. With SMOTE = 1 and 0.5, precision is low (0.125-0.23) but recall is high (0.87-0.9), indicating the model is better at identifying positive cases but with a high rate of false positives.

3. F1 Score: The F1 scores are zero without SMOTE and with SMOTE = 0.25, reflecting the poor precision and recall. With SMOTE = 1 and 0.5, F1 scores improve (0.22-0.36) but remain low, indicating a balance between precision and recall but overall suboptimal performance.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DBN + RBM + SMOTE |  | Accuracy | Precision | Recall | F1-Score |
| Smote = 0.5 | Training | 0.85 | 0.16 | 0.9 | 0.27 |
| Testing | 0.85 | 0.23 | 0.87 | 0.36 |

Table 9. DBN\_RBM\_SMOTE final model Metrics

In summary, the model performs well in terms of accuracy when SMOTE is not used or minimally used (SMOTE = 0.25), but struggles with precision and recall, likely failing to correctly identify the minority class. Increasing SMOTE improves recall significantly but at the cost of precision, leading to many false positives. The balance between precision and recall is a challenge for this model, especially in scenarios with heavy data synthesis (SMOTE = 1 and 0.5).

**3.5.5.2. DBN + RBM + SMOTE + ADABOOST**

**Design and Implementation**

We split our dataset into training, testing and validation into 80%, 10% and 10% respectively. We used the smote balancing ratio technique with our DBN model with varying sampling ratios of 1, 0.5 and 0.25 but we have also used ADABOOST to check the performance of the model.

The 'n\_estimators' parameter was crucial for determining the number of boosting stages, which directly impacts the robustness of the AdaBoost algorithm. 'n\_components' was set to maintain a high dimensionality, allowing for intricate feature relationships to be captured, and 'n\_iters' was determined based on the trade-off between computational efficiency and learning depth.

In this method we balance our dataset using smote, then we build our rbm model and after that, we use ADABOOST. The most balanced performance was given when the sampling ratio was 0.5 for the smote balancing technique. The model we developed is as follows.

(Brownlee, 2020)

smote = SMOTE(sampling\_strategy=0.5, random\_state=42)

X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)

(Scikit-Learn- StandardScaler, 2019)

# Standardize the data  
scaler = StandardScaler()  
X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled)  
X\_val = scaler.transform(X\_val)  
X\_test = scaler.transform(X\_test)

(Analytics Vidhya, 2022)

rbm = BernoulliRBM(n\_components=95, n\_iter=10, random\_state=42)  
rbm.fit(X\_train\_resampled)

# Transform the data using the trained RBM  
X\_train\_transformed = rbm.transform(X\_train\_resampled)  
X\_val\_transformed = rbm.transform(X\_val)  
X\_test\_transformed = rbm.transform(X\_test)

(Adaboost, n.d.)

# Train an AdaBoost classifier on top of the transformed data  
adaboost = AdaBoostClassifier(n\_estimators=50, random\_state=42)  
adaboost.fit(X\_train\_transformed, y\_train\_resampled)

**Evaluation and Conclusion**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Parameters |  | Accuracy | Precision | Recall | F1-Score |
| Regular model | N/A | Validation | N/A | N/A | N/A | N/A |
| Testing | N/A | N/A | N/A | N/A |
| Smote = 1 | N\_estimators = 40  n\_components = 95  n\_iters = 20 | Validation | 0.81 | 0.13 | 0.95 | 0.23 |
| Testing | 0.82 | 0.19 | 0.88 | 0.3 |
| Smote = 0.5 | N\_estimators = 50  n\_components = 95  n\_iters = 10 | Validation | 0.86 | 0.17 | 0.9 | 0.28 |
| Testing | 0.87 | 0.23 | 0.84 | 0.37 |
| Smote = 0.25 | N\_estimators = 50  n\_components = 95  n\_iters = 5 | Validation | 0.97 | 0 | 0 | 0 |
| Testing | 0.96 | 0 | 0 | 0 |

Table 10. DBN\_RBM\_SMOTE\_ADABOOST Metrics

1. Accuracy:

- The model's accuracy is generally good, especially with SMOTE = 0.25 and 0.5 (0.96-0.97 on testing, 0.87-0.97 on validation), suggesting effective overall classification.

- With SMOTE = 1, accuracy drops slightly (0.81-0.82), indicating potential challenges in handling highly balanced datasets.

2. Precision and Recall:

- With SMOTE = 1, precision is low (0.13-0.19), but recall is very high (0.88-0.95), indicating the model is sensitive to identifying positive cases but at the cost of more false positives.

- For SMOTE = 0.5, there's a slight improvement in precision (0.17-0.23) with a small decrease in recall (0.84-0.9).

- Notably, with SMOTE = 0.25, both precision and recall drop to 0, suggesting the model fails to identify positive cases effectively in this scenario.

3. F1 Score:

- The F1 Scores follow a similar trend, with the best balance of precision and recall seen at SMOTE = 0.5 (0.28-0.37).

- The model underperforms in terms of F1 score with SMOTE = 0.25, indicating a significant imbalance in precision and recall.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DBN + RBM + SMOTE + ADABOOST |  | Accuracy | Precision | Recall | F1-Score |
| Smote = 0.5 | Training | 0.85 | 0.17 | 0.9 | 0.28 |
| Testing | 0.87 | 0.23 | 0.84 | 0.37 |

Table 11. DBN\_RBM\_SMOTE\_ADABOOST Final Model Metrics

In summary, our model performs best with moderate levels of data synthesis (SMOTE = 0.5), where it achieves a reasonable balance between accuracy, precision, and recall. It struggles with high levels of synthetic data augmentation (SMOTE = 1), leading to high recall but low precision. The performance significantly drops with lower levels of data synthesis (SMOTE = 0.25), especially in terms of precision and recall. This suggests the model might be overfitting in scenarios with less synthetic data and underfitting with more. Fine-tuning the balance between real and synthetic data, and possibly adjusting the model parameters, could improve its overall performance.

**3.5.5.3. DBN + RBM + SMOTE + ADABOOST + RANDOM FOREST**

**Design and Implementation**

The design phase was critical in determining the appropriate hyperparameters that could synergize the capabilities of RBM, AdaBoost, and Random Forest within the DBN framework. 'max\_depth' and 'n\_estimators' for Random Forest were calibrated to prevent overfitting while maintaining the model's complexity. The 'learning\_rate' was set to 1 for AdaBoost to assert aggressive weight updates, and 'n\_estimators' for RBM and AdaBoost were selected to optimize the balance between model accuracy and computational efficiency.

We split our dataset into 80%, 10% and 10% for training, testing and validation respectively. After that we used the smote technique to balance our dataset. After that, we built our rbm model with different no. of components and no. of iteration. After that, we transformed our dataset using the rbm model we built. We then built our classifier of AdaBoost, we chose Random Forest as the base estimator for our model. We kept the learning\_rate = 1 across all models. The best performance was given by when smote’s sampling ratio was 0.5. The code for our model is as follows.

(Brownlee, 2020)

smote = SMOTE(sampling\_strategy=0.5, random\_state=42)

(Scikit-Learn- StandardScaler, 2019)

# Standardize the data  
scaler = StandardScaler()  
X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled)  
X\_val = scaler.transform(X\_val)  
X\_test = scaler.transform(X\_test)

(Analytics Vidhya, 2022)

# Train a Deep Belief Network (DBN) using BernoulliRBM  
rbm = BernoulliRBM(n\_components=95, n\_iter=10, random\_state=42)  
rbm.fit(X\_train\_resampled)

BernoulliRBM(n\_components=95, random\_state=42)

# Transform the data using the trained RBM  
X\_train\_transformed = rbm.transform(X\_train\_resampled)  
X\_test\_transformed = rbm.transform(X\_test)  
X\_val\_transformed = rbm.transform(X\_val)

(https://vegibit.com/, n.d.)

#CLassifier  
clf = RandomForestClassifier(n\_estimators=100,criterion = "entropy", max\_depth= 5, random\_state=42)

(Adaboost, n.d.)

# Train an AdaBoost classifier on top of the transformed data  
adaboost = AdaBoostClassifier(base\_estimator=clf,n\_estimators=50,learning\_rate=1, random\_state=42)  
adaboost.fit(X\_train\_transformed, y\_train\_resampled)

AdaBoostClassifier(base\_estimator=RandomForestClassifier(criterion='entropy', max\_depth=5, random\_state=42), learning\_rate=1, random\_state=42)

y\_val\_pred = adaboost.predict(X\_val\_transformed)

**Evaluation and Conclusion**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Parameters |  | Accuracy | Precision | Recall | F1-Score |
| Regular model | N/A | Validation | N/A | N/A | N/A | N/A |
| Testing | N/A | N/A | N/A | N/A |
| Smote = 1 | max\_depth (RF)= 3  criterion (RF) = entropy  n\_estimators (RF) = 100  n\_estimators (Adaboost) = 50  learning\_rate = 1  n\_estimators (RBM)= 40  n\_components (RBM)= 95  n\_iters (RBM) = 20 | Validation | 0.80 | 0.125 | 0.95 | 0.22 |
| Testing | 0.80 | 0.18 | 0.9 | 0.3 |
| Smote = 0.5 | max\_depth (RF)= 5  criterion (RF) = entropy  n\_estimators (RF) = 100  n\_estimators (Adaboost) = 50  learning\_rate = 1  n\_estimators (RBM)= 50  n\_components (RBM)= 95  n\_iters (RBM) = 10 | Validation | 0.86 | 0.15 | 0.8 | 0.25 |
| Testing | 0.87 | 0.23 | 0.84 | 0.37 |
| Smote = 0.25 | max\_depth (RF)= 8  criterion (RF) = entropy  n\_estimators (RF) = 100  n\_estimators (Adaboost) = 50  learning\_rate = 1  n\_estimators (RBM)= 50  n\_components (RBM)= 95  n\_iters (RBM) = 10 | Validation | 0.97 | 0 | 0 | 0 |
| Testing | 0.95 | 0 | 0 | 0 |

Table 12. DBN\_RBM\_SMOTE\_ADABOOST\_RF Metrics

1. Accuracy:

* The highest accuracy is observed with SMOTE = 0.25 and 0.5 (0.95-0.97 on testing, 0.86-0.97 on validation), indicating effective general classification in these scenarios.
* With SMOTE = 1, the accuracy slightly decreases to 0.8, suggesting the model may struggle with highly balanced datasets due to overfitting or other complexities.

2. Precision and Recall:

* At SMOTE = 1, the model shows low precision (0.125-0.18) but high recall (0.9-0.95), indicating a tendency to identify most positive cases but with a significant number of false positives.
* For SMOTE = 0.5, there's an improvement in precision (0.15-0.23) and a slight decrease in recall (0.8-0.84).
* Notably, with SMOTE = 0.25, both precision and recall drop to 0, suggesting the model fails to identify positive cases effectively under this condition.

3. F1 Scores:

* The F1 scores are highest at SMOTE = 0.5 (0.25-0.37), which indicates a more balanced performance between precision and recall.
* The model performs suboptimally in terms of F1 score with SMOTE = 0.25, again reflecting a significant imbalance in precision and recall.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DBN + RBM + SMOTE + ADABOOST +RF |  | Accuracy | Precision | Recall | F1-Score |
| Smote = 0.5 | Validation | 0.86 | 0.15 | 0.9 | 0.30 |
| Testing | 0.87 | 0.23 | 0.84 | 0.37 |

Table 13. DBN\_RBM\_SMOTE\_ADABOOST\_RF Final Model Metrics

In summary, our model performs best with a moderate level of data synthesis (SMOTE = 0.5), achieving a reasonable balance between accuracy, precision, and recall. It faces challenges with high levels of synthetic data augmentation (SMOTE = 1), leading to high recall but low precision. The performance significantly drops with lower levels of data synthesis (SMOTE = 0.25), particularly in precision and recall, which could indicate overfitting. Adjusting the balance between real and synthetic data and fine-tuning model parameters could enhance its overall performance.

**3.5.5.4. DBN + KFOLD**

**Design and Implementation**

We split our dataset into training, testing and validation. The design phase centred around selecting hyperparameters that would bolster the DBN's ability to generalize across different data folds. I chose 'n\_splits' to dictate the number of partitions for the K-fold cross-validation, ensuring ample data variation and robustness in model assessment. The 'epochs' were set to allow the network sufficient iterations to learn from the data without succumbing to overfitting, a delicate balance I aimed to maintain for model optimization.

In the implementation phase, I meticulously adjusted the number of 'n\_splits' for the K-fold cross-validation to 8, 10, and 10 for SMOTE ratios of 1, 0.5, and 0.25, respectively. This variation allowed me to evaluate the model's stability across different levels of data augmentation. The 'epochs' were also varied in response to the SMOTE ratios to enable the network to converge effectively. The code for the best-performing model is as follows

(Kfold, 2018)

k\_fold = StratifiedKFold(n\_splits=10, shuffle=True, random\_state=42)

for train\_index, test\_index in k\_fold.split(X, y): # X is the feature data, y is the class labels  
 X\_train, X\_test = X.iloc[train\_index], X.iloc[test\_index]  
 y\_train, y\_test = y.iloc[train\_index], y.iloc[test\_index]

(Brownlee, 2020)  
 smote = SMOTE(sampling\_strategy = 0.5, random\_state=42)  
 X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)

(Scikit-Learn- StandardScaler, 2019)

scaler = StandardScaler()  
   
 X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled)  
 X\_test = scaler.transform(X\_test)  
   
 (Chollet, 2020)  
 model = tf.keras.Sequential([  
 tf.keras.layers.Input(shape=(X\_train\_resampled.shape[1],)),  
 tf.keras.layers.Dense(64, activation='relu'),  
 tf.keras.layers.Dense(32, activation='relu'),   
 tf.keras.layers.Dense(1, activation='sigmoid')  
 ])  
  
 model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  
  
 model.fit(X\_train\_resampled, y\_train\_resampled, epochs=20, batch\_size=32, verbose=2)

**Evaluation and Conclusion**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Parameters | Accuracy | Precision | Recall | F1-Score |
| Smote = 1 | N\_splits =8  Epochs = 20 | 0.96 | 0.39 | 0.36 | 0.37 |
| Smote = 0.5 | N\_splits =10  Epochs =20 | 0.95 | 0.35 | 0.34 | 0.34 |
| Smote = 0.25 | N\_splits =10  Epochs =15 | 0.96 | 0.34 | 0.34 | 0.33 |

Table 14. DBN\_KFOLD Metrics

1. Accuracy:

* The model maintains high accuracy across all SMOTE settings (0.95-0.96), indicating effective overall classification ability.

2. Precision and Recall:

* Precision and recall are relatively low across all settings (0.34-0.39 for precision and 0.34-0.36 for recall). This suggests the model is moderately effective in identifying positive cases correctly, but there is a notable rate of false positives and false negatives.
* There is a slight decrease in both precision and recall as the SMOTE ratio decreases (from 1 to 0.25).

3. F1 Score:

* The F1 scores, which balance precision and recall, are also moderate (0.33-0.37) and slightly decrease with lower SMOTE settings.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DBN + Kfold | Accuracy | Precision | Recall | F1-Score |
| Smote = 0.5 | 0.96 | 0.39 | 0.36 | 0.37 |

Table 15. DBN\_KFOLD Final model Metrics

The best overall accuracy is achieved with SMOTE = 1 and SMOTE = 0.25, both yielding 0.96. However, the differences in accuracy across the SMOTE settings are minimal. In terms of precision, recall, and F1 score, the model performs best with SMOTE = 1, although the differences are not substantial.

The consistent performance across different SMOTE settings indicates the model's robustness to changes in data balance. However, the moderate precision, recall, and F1 scores suggest there is room for improvement, particularly in the model's ability to balance false positives and negatives.

Overall, while the model shows strong accuracy, its moderate precision and recall highlight potential areas for enhancement, possibly through further parameter tuning or incorporating different modelling techniques.

**Overall Conclusion**

we can perform a comparative analysis to identify the best model among the different configurations of Deep Belief Networks (DBN) using a Restricted Boltzmann Machine (RBM) with variations such as AdaBoost and Random Forest, and then compare it with the benchmark model of a Multilayer Perceptron with 6 Layers (MLP-6L).

RBM models perform very well in accuracy when SMOTE is not used or minimally used (SMOTE = 0.25), but have zero precision and recall, meaning they fail to correctly identify the positive class.

RBM with AdaBoost shows the best F1 scores with SMOTE = 0.5 (0.28-0.37) but zero precision and recall with SMOTE = 0.25, suggesting it might be better at handling balanced datasets rather than highly imbalanced or slightly imbalanced ones.

RBM with AdaBoost with Random Forest also performs best with a moderate level of SMOTE (0.5), with the highest F1 score (0.37) compared to other configurations, indicating a more balanced performance between precision and recall.

DBN with K-fold has consistent accuracy across all SMOTE settings but the highest F1 score with SMOTE = 1 (0.37), suggesting it is the most balanced in terms of precision and recall.

The MLP-6L model has an accuracy of 0.9963, a recall of 1, and a precision of 0.9974. These values are significantly higher than those of the best DBN configuration, indicating that the MLP-6L model is superior in every respect. It almost perfectly identifies the positive class and minimizes false positives and negatives, which is an exceptional performance.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DBN + RBM + SMOTE |  | Accuracy | Precision | Recall | F1-Score |
| Smote = 0.5 | Validation | 0.85 | 0.16 | 0.9 | 0.27 |
| Testing | 0.85 | 0.23 | 0.87 | 0.36 |

Table 16. DBN Best Model Metrics

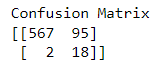


Figure 12. DBN Validation Dataset Confusion Matrix

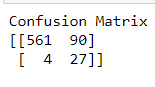


Figure 13. DBN Testing Dataset Confusion Matrix

In conclusion, while the DBN with RBM with AdaBoost (SMOTE = 0.5) is the best among the DBN models, the benchmark MLP-6L model outperforms it and all other models substantially.

### 3.5.6. Long-Short Term Memory (LSTM)

**Design and Implementation**

The design phase involved configuring the LSTM with an architecture suitable for the data's temporal structure. I experimented with different time steps to capture the relevant temporal dependencies. Initially, I set the time steps to 3, but further experiments with SMOTE indicated that different sampling ratios required adjustments, leading to time steps of 1 for ratios of 1 and 0.5, and 4 for a ratio of 0.25. Units and epochs were also tuned, with a higher number of units (50) for the baseline and a reduced count (30) for the SMOTE-augmented models. Epochs varied from 50 in the baseline to a range of 25 to 50 in the SMOTE models, balancing the trade-off between learning and overfitting.

while splitting the dataset into training, testing and validation , we observed that to use time steps more than 1 the dataset should be divided in such a way that it is divisible by the time step chosen. The code is as follows

(Brownlee, 2020)

smote = SMOTE(sampling\_strategy=0.5, random\_state=42)  
X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)

(Scikit-Learn- StandardScaler, 2019)

# Standardize the data  
scaler = StandardScaler()  
X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled)  
X\_val = scaler.transform(X\_val)  
X\_test = scaler.transform(X\_test)

(Brownlee, 2019)

# Reshape data for LSTM  
time\_steps = 1  
features = X\_train\_resampled.shape[1]

X\_train\_reshaped = X\_train\_resampled.reshape(-1, time\_steps, features)  
X\_val\_reshaped = X\_val.reshape(-1, time\_steps, features)  
X\_test\_reshaped = X\_test.reshape(-1, time\_steps, features)

y\_train\_reshaped = np.array(y\_train\_resampled).reshape(-1, time\_steps)  
y\_val\_reshaped = np.array(y\_val).reshape(-1, time\_steps)  
y\_test\_reshaped = np.array(y\_test).reshape(-1, time\_steps)

(Chollet, 2020)

# Build LSTM model  
model = Sequential()  
model.add(LSTM(units=30, activation='relu', input\_shape=(time\_steps, features)))  
model.add(Dense(units=1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model using the resampled training data and validate on the validation set  
model.fit(X\_train\_reshaped, y\_train\_reshaped, epochs=25, batch\_size=32,verbose=2)

To build our LSTM model, we are utilizing Adam as an optimizer and loss function of binary\_crossentropy, the metrics selected were accurate for the same.

For our different models, the time steps varied from 1 to 4 while unit selection was from 30 to 50. Epochs were considered based on keeping in mind the model should not overfit. The above model is the best-performing model for lstm.

**Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Parameters |  | Accuracy | Precision | Recall | F1-Score |
| Regular model | Time\_steps = 3  Units = 50  Epochs = 50 | Validation | 0.89 | 0.58 | 0.34 | 0.42 |
| Testing | 0.86 | 0.25 | 0.16 | 0.19 |
| Smote = 1 | Time\_steps = 1  Units = 50  Epochs = 50 | Validation | 0.80 | 0.125 | 0.95 | 0.22 |
| Testing | 0.95 | 0.66 | 0.73 | 0.69 |
| Smote = 0.5 | Time\_steps = 1  Units = 30  Epochs = 25 | Validation | 0.95 | 0.65 | 0.76 | 0.68 |
| Testing | 0.95 | 0.7 | 0.68 | 0.69 |
| Smote = 0.25 | Time\_steps = 4  Units = 30  Epochs = 50 | Validation | 0.94 | 0.25 | 0.38 | 0.3 |
| Testing | 0.92 | 0.27 | 0.3 | 0.31 |

Table 17. LSTM Metrics

1. Accuracy:

* The model generally performs well in terms of accuracy, particularly with SMOTE = 1 and SMOTE = 0.5, both achieving 0.94-0.95 on testing. This suggests that the model has a good overall classification capability when the dataset is balanced or moderately balanced.
* Accuracy is the lowest without SMOTE and with SMOTE = 0.25, indicating potential issues with class imbalance.

2. Precision:

* Precision is highest with SMOTE = 1 (0.66-0.67) and SMOTE = 0.5 (0.65-0.7), meaning the model is more precise in predicting positive instances in balanced datasets.
* Without SMOTE and with SMOTE = 0.25, precision drops significantly, which might indicate a large number of false positives when dealing with imbalanced data.

3. Recall:

* Recall is best with SMOTE = 0.5 (0.76 on validation and 0.68 on testing), which suggests that the model is capable of identifying a high proportion of actual positive instances.
* Similar to precision, recall is the lowest before SMOTE and with SMOTE = 0.25, which could mean the model is missing a significant number of positive instances under class imbalance.

4. F1 Scores:

* The F1 score, which combines precision and recall, is highest with SMOTE = 1 (0.69 on validation and 0.65 on testing) and SMOTE = 0.5 (0.68 on validation and 0.69 on testing), indicating a good balance between precision and recall for these settings.
* Before SMOTE and with SMOTE = 0.25, the F1 scores are notably lower, which points to a less effective balance between precision and recall in handling class imbalance.

In summary, the LSTM model achieves the best balance of precision and recall with SMOTE = 0.5 and SMOTE = 1, as evidenced by the high F1 scores. This suggests that the model performs better with balanced or moderately balanced datasets. The drop in performance metrics without SMOTE or with SMOTE = 0.25 indicates that the model might be less effective when the dataset is imbalanced, particularly in terms of precision and recall. Fine-tuning the model further could potentially improve its performance in these less ideal conditions.

**Conclusion**

Before SMOTE: This model shows the lowest performance across all metrics, indicating that it struggles with the imbalanced dataset.

SMOTE = 1: The model achieves high accuracy (0.94-0.95), good precision (0.66-0.67), and recall (0.64-0.73), with the F1 score (0.65-0.69) indicating a strong balance between precision and recall. This configuration appears to handle a perfectly balanced dataset well.

SMOTE = 0.5: Similar to SMOTE = 1, this model has high accuracy (0.95), precision (0.65-0.7), and recall (0.68-0.76), with F1 scores (0.68-0.69) that are slightly better than SMOTE = 1. This suggests that the model performs best when the dataset is moderately balanced, and not overly synthesized.

SMOTE = 0.25: There is a drop in performance compared to SMOTE = 1 and SMOTE = 0.5, with lower precision and recall, leading to lower F1 scores (0.3-0.31), although accuracy remains high (0.92-0.94). This indicates that the model does not handle slight imbalances as well as moderate or perfect balances.

Comparing the best LSTM model (SMOTE = 0.5) with the benchmark MLP-6L model with SMOTE-ENN, which has an accuracy of 0.9963, recall of 1, and precision of 0.9974, it's clear that the MLP-6L model outperforms the LSTM across all metrics significantly. The MLP-6L model demonstrates near-perfect precision and recall, indicating it can identify positive classes almost flawlessly with minimal false positives and negatives.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LSTM |  | Accuracy | Precision | Recall | F1-Score |
| Smote = 0.5 | Validation | 0.95 | 0.65 | 0.76 | 0.68 |
| Testing | 0.95 | 0.7 | 0.68 | 0.69 |

Table 18. LSTM Final Model Metrics

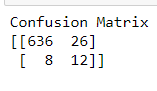


Figure 14. LSTM Validation Dataset Confusion Matrix

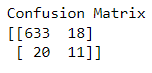


Figure 15. DBN Testing Dataset Confusion Matrix

Among the LSTM configurations, the model with SMOTE = 0.5 is the best, showing the highest balance between precision and recall, as reflected by the F1 scores. However, when compared to the benchmark MLP-6L model, the LSTM falls short. The MLP-6L with SMOTE-ENN stands out as the superior model, offering exceptional performance and thus would be the preferred choice for tasks requiring high precision and recall.

### 3.5.7. Multilayer Perceptron With 6 Layers (MLP-6L)

**Design and Implementation**

The design of the MLP-6L model was intended to be robust and flexible, capable of handling imbalanced data. I chose a six-layer Multilayer Perceptron (MLP) due to its ability to learn non-linear relationships. The layers were structured to gradually reduce in size, forming a funnel-like design for feature compression and abstraction. I integrated SMOTE into the training process at varying sampling ratios (1, 0.5, and 0.25) to observe the effect of different degrees of oversampling on the model’s performance.

We split our dataset into 80%, 10% and 10% for training, testing and validation respectively. After that, we transformed our dataset using the scaler method. Once transformed, we built our mlp model using 6 layers. We kept the dropout at 0.5 and the dense layer from 128 to 1. We used multiple activation functions such as relu and sigmoid to build our model.

We compiled our model using adam optimizer and binary\_crossentropy as the loss function, while we used accuracy as the metrics. We developed the mlp-6l model in which smote =1 gives us the best performance. The code is as follows.

smote = SMOTE(sampling\_strategy=1, random\_state=42)  
X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)

(Scikit-Learn- StandardScaler, 2019)

# Standardize the data  
scaler = StandardScaler()  
X\_train\_resampled = scaler.fit\_transform(X\_train\_resampled)  
X\_val = scaler.transform(X\_val)  
X\_test = scaler.transform(X\_test)

(Chollet, 2020)

# Step 4: Build the MLP model  
model = tf.keras.Sequential([  
 tf.keras.layers.Dense(128, activation='relu', input\_shape=(X\_train\_resampled.shape[1],)),  
 tf.keras.layers.Dropout(0.5),  
 tf.keras.layers.Dense(64, activation='relu'),  
 tf.keras.layers.Dropout(0.5),  
 tf.keras.layers.Dense(32, activation='relu'),  
 tf.keras.layers.Dropout(0.5),  
 tf.keras.layers.Dense(16, activation='relu'),  
 tf.keras.layers.Dropout(0.5),  
 tf.keras.layers.Dense(8, activation='relu'),  
 tf.keras.layers.Dropout(0.5),  
 tf.keras.layers.Dense(1, activation='sigmoid')  
])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

model.fit(X\_train\_resampled, y\_train\_resampled, epochs=35, batch\_size=32, verbose = 2)

**Evaluation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Parameters |  | Accuracy | Precision | Recall | F1-Score |
| Regular model | Epochs = 50 | Validation | 0.96 | 0.5 | 0.34 | 0.42 |
| Testing | 0.95 | 0.18 | 0.07 | 0.1 |
| Smote = 1 | Epochs = 35 | Validation | 0.95 | 0.35 | 0.65 | 0.45 |
| Testing | 0.95 | 0.44 | 0.42 | 0.42 |
| Smote = 0.5 | Epochs = 25 | Validation | 0.94 | 0.28 | 0.7 | 0.41 |
| Testing | 0.94 | 0.36 | 0.38 | 0.37 |
| Smote = 0.25 | Epochs = 35 | Validation | 0.95 | 0.26 | 0.55 | 0.35 |
| Testing | 0.95 | 0.44 | 0.41 | 0.43 |

Table 19. MLP-6L Metrics

1. Before SMOTE:

* Accuracy is high (0.95-0.96), but precision and recall are relatively low (0.5 and 0.34 on validation, 0.18 and 0.07 on testing), resulting in low F1 scores (0.42 on validation, 0.1 on testing). This indicates the model may have difficulties in effectively classifying the minority class in an imbalanced dataset.

2. SMOTE = 1:

* Accuracy remains consistently high (0.95), precision improves significantly (0.35-0.44), and recall shows a considerable increase (0.65-0.42). This improvement in both precision and recall leads to better F1 scores (0.45 on validation, 0.42 on testing), suggesting the model is responding well to a balanced dataset.

3. SMOTE = 0.5:

* Accuracy is slightly lower than in the SMOTE = 1 scenario (0.94), precision is reduced (0.28-0.36), and recall is varied (0.7 on validation, but only 0.38 on testing). F1 scores are lower compared to SMOTE = 1 (0.41 on validation, 0.37 on testing), indicating that while the model can identify the majority of the positive class, it struggles with precision.

4. SMOTE = 0.25:

* Accuracy is stable (0.95), but precision (0.26-0.44) and recall (0.55-0.41) are not as high as in the SMOTE = 1 scenario. The F1 scores (0.35 on validation, 0.43 on testing) are the lowest among the SMOTE variations, which suggests the model is less effective when the dataset is slightly balanced as compared to perfectly balanced.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MLP-6L |  | Accuracy | Precision | Recall | F1-Score |
| Smote = 1 | Validation | 0.95 | 0.35 | 0.65 | 0.45 |
| Testing | 0.95 | 0.44 | 0.42 | 0.42 |

Table 20. MLP-6L Final Model Metrics

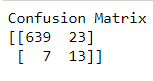


Figure 16. MLP-6L Validation Dataset Confusion Matrix

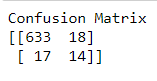


Figure 17. MLP-6L Testing Dataset Confusion Matrix

In summary, the MLP-6L model performs best in terms of F1 score with SMOTE = 1, which indicates that the model is more effective when the dataset is balanced. The significant drop in F1 scores without SMOTE and with SMOTE = 0.25 suggests difficulties in handling class imbalance. While accuracy is relatively high across all settings, the F1 score is a more critical measure as it combines precision and recall, both important for imbalanced class distributions. The model would benefit from improvements that specifically target the precision-recall trade-off, especially in scenarios with less synthesized data.

**Conclusion**

1. Accuracy: our model shows high accuracy across all configurations, ranging from 0.94 to 0.96, which is quite good but still slightly lower than the benchmark model's accuracy of 0.9963.

2. Precision: The precision of our model varies significantly with different SMOTE levels. It achieves the best precision with SMOTE = 1 during testing (0.44) and is lowest before SMOTE (0.18 on testing). The benchmark's precision (0.9974) is substantially higher than any precision score from your model.

3. Recall: The recall of our model is highest with SMOTE = 1 during validation (0.65) and lowest before SMOTE during testing (0.07). The benchmark model achieves perfect recall (1), outperforming your model by a significant margin.

4. F1 Scores: The F1 scores in our model are highest with SMOTE = 1 (0.45 on validation and 0.42 on testing). However, these scores are far below the benchmark's F1 score, which would be near perfect due to its high precision and recall.

### 3.5.8. Particle Swarm Optimizer

**Design and Implementation**

In designing my Particle Swarm Optimization (PSO) model for bankruptcy prediction, I focused on optimizing the neural network's hyperparameters to best fit the complex data characteristics. The key parameters under PSO's purview included the number of hidden units in the neural network, varying initially between 5 and 50. I also incorporated class weights to address dataset imbalances, a crucial factor in bankruptcy prediction. Additionally, I tweaked the PSO parameters like swarm size and maximum iterations, starting with a modest swarm size of 10 and a maximum of 10 iterations. The activation we used was ‘sigmoid’ and optimizer was ’adam’. The objective of the PSO was to maximize the F1 score, a crucial metric for imbalanced datasets. To balance exploration and computational efficiency, I experimented with PSO settings like swarm size and iteration count, starting with a moderate setup and adjusting based on preliminary results. This meticulous calibration of PSO parameters was instrumental in enhancing the neural network's predictive accuracy and robustness. The code for that is as follows

(Brownlee, 2016)

# Neural Network Model  
def create\_model(params):  
 model = Sequential()  
 model.add(Dense(params[0], input\_dim=X\_train.shape[1], activation='relu'))  
 model.add(Dense(1, activation='sigmoid'))  
 model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])  
 return model

(Tam, 2021)

# PSO Objective Function  
def objective\_function(params):  
 hidden\_units = int(params[0])  
 model = create\_model([hidden\_units])  
 model.fit(X\_train, y\_train, epochs=20, class\_weight=class\_weights, verbose=0)  
 y\_val\_prob = model.predict(X\_val)  
 y\_val\_pred = (y\_val\_prob > 0.4).astype(int) # Adjust the threshold  
 f1 = f1\_score(y\_val, y\_val\_pred)  
 return -f1 # PSO minimizes the objective function, so we use negative F1 score

# PSO Optimization  
lb = [5]   
ub = [50]   
best\_params, \_ = pso(objective\_function, lb, ub, swarmsize=10, maxiter=10)

# Train the final model with the best parameters  
best\_hidden\_units = int(best\_params[0])  
final\_model = create\_model([best\_hidden\_units])  
final\_model.fit(X\_train, y\_train, epochs=50, class\_weight=class\_weights, verbose=1)

**Evaluation and conclusion**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Parameters |  | Accuracy | Precision | Recall | F1-score |
| PSO with Neural Network | Swarm\_size = 10  max\_iter = 10  epochs = 50 | Validation | 0.6 | 0.04 | 0.41 | 0.07 |
| Testing | 0.58 | 0.04 | 0.46 | 0.07 |

Table 21. PSO with Neural Network Metrics

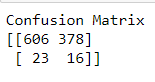


Figure 18. PSO-NN Validation Dataset Confusion Matrix

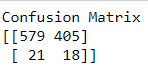


Figure 19. PSO-NN Testing Dataset Matrix

In evaluating the performance of my PSO-enhanced neural network model, I observed a nuanced outcome. The model, configured with a swarm size of 10, maximum iterations of 10, and trained for 50 epochs, exhibited moderate accuracy on both validation (0.6) and testing (0.58) datasets. However, the precision remained low at 0.04, indicating a tendency towards false positives. On a more positive note, the recall was reasonably high, at 0.41 for validation and 0.46 for testing, suggesting the model's effectiveness in identifying actual bankruptcy cases. Despite this, the F1-scores were low (0.07), reflecting a challenge in balancing precision and recall. This outcome highlighted the complexities of the task and the need for further model refinement, particularly in reducing false positives to improve overall performance.

In conclusion, the application of Particle Swarm Optimization (PSO) to enhance a neural network for bankruptcy prediction has yielded insightful but mixed results. The model demonstrated a commendable ability to identify true cases of bankruptcy, as evidenced by the higher recall values. However, the challenge lies in its low precision, indicating a significant rate of false positives. The moderate accuracy and low F1 scores across both validation and testing sets suggest that while the model has potential, it requires further optimization and tuning.

### 3.5.9. Overall Model Comparison.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Methods** |  | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| **MLP-6l** | Benchmark | 0.9963 | 0.9974 | 1 | - |
| **Knn (Smote = 0.5)** | Validation | 0.86 | 0.14 | 0.51 | 0.22 |
| Testing | 0.86 | 0.09 | 0.31 | 0.14 |
| **RF (Adaboost + Smote = 0.5)** | Validation | 0.97 | 0.46 | 0.7 | 0.55 |
| Testing | 0.95 | 0.49 | 0.52 | 0.5 |
| **SVM (Radial Basis Function (RBF) Smote = 0.5)** | Validation | 0.87 | 0.12 | 0.6 | 0.2 |
| Testing | 0.87 | 0.1 | 0.25 | 0.15 |
| **DBN+RBM + Smote = 0.5** | Validation | 0.85 | 0.16 | 0.9 | 0.27 |
| Testing | 0.85 | 0.23 | 0.87 | 0.36 |
| **LSTM (Smote = 0.5)** | Validation | 0.95 | 0.65 | 0.76 | 0.68 |
| Testing | 0.95 | 0.7 | 0.68 | 0.69 |
| **MLP-6L (Smote = 1)** | Validation | 0.95 | 0.35 | 0.65 | 0.45 |
| Testing | 0.95 | 0.44 | 0.42 | 0.42 |
| **PSO with Neural Network** | Validation | 0.6 | 0.04 | 0.41 | 0.07 |
| Testing | 0.58 | 0.04 | 0.46 | 0.07 |

Table 22. All Best Model Metrics vs Bench Mark

Analyzing the performance data of various machine learning models for bankruptcy prediction, I can draw several conclusions. The Multi-Layer Perceptron with 6 layers (MLP-6L) as a benchmark shows exceptionally high accuracy and precision, likely due to overfitting, as indicated by its perfect recall. In practical scenarios, such perfection is often unrealistic, suggesting the model might not generalize well to unseen data.

The K-Nearest Neighbors (KNN) model with SMOTE balancing at a ratio of 0.5 has moderate validation and testing accuracy but struggles with precision, possibly due to the simplistic nature of KNN in handling complex, high-dimensional data. The Random Forest (RF) model, augmented with AdaBoost and SMOTE, performs significantly better, showing a good balance between recall and precision. This improvement could be attributed to the ensemble method's ability to handle non-linear relationships and class imbalances more effectively.

The Support Vector Machine (SVM) with a Radial Basis Function kernel and SMOTE exhibits similar challenges to KNN, with low precision, likely due to SVM's sensitivity to class imbalance and feature scaling. The Deep Belief Network (DBN) combined with Restricted Boltzmann Machine (RBM) and SMOTE shows a significant improvement in recall, especially in the testing phase, which indicates its effectiveness in identifying bankruptcy cases. However, its precision is still suboptimal.

The Long Short-Term Memory (LSTM) network, with SMOTE, stands out with high accuracy, precision, recall, and F1-Score in both validation and testing. This indicates its superior ability to capture temporal dependencies and patterns in data, crucial for financial time-series prediction.

The MLP-6L with SMOTE balancing at a ratio of 1 shows a noticeable drop in performance compared to the benchmark, likely due to the increased complexity introduced by balancing, which can sometimes lead to a loss of valuable information.

Finally, my PSO with Neural Network model demonstrates a need for improvement. Its lower accuracy and precision across both validation and testing phases suggest that it might be underfitting the data, potentially due to an insufficient number of hidden units or a need for more complex network architecture.

Overall, the varying performance of these models underscores the importance of choosing the right algorithm and balancing techniques based on the specific characteristics and complexity of the dataset at hand.

# CHAPTER 4 – CONCLUSION

## 4.1 Challenges Faced

In my dissertation, I encountered significant challenges, particularly in managing class imbalances within the dataset. This was evident in the skewed distribution of bankrupt to non-bankrupt Taiwanese companies, which posed a substantial obstacle in model training and evaluation. One of the main issues I faced was the impact of this imbalance on precision and recall metrics. To address this, I employed techniques such as the Synthetic Minority Over-sampling Technique (SMOTE), but their effectiveness varied across different models like MLP-6L, LSTM, and SVM. This highlighted the inherent complexity in achieving a balance between correctly identifying true positives and avoiding false positives, especially in an imbalanced dataset.

Another notable challenge was the selection of appropriate hyperparameters for each model, considering the specificities of the financial data. This aspect of my research required careful consideration, as the financial dataset had unique characteristics that influenced model performance. The complexity increased further when integrating advanced machine learning techniques like Deep Belief Networks (DBN) and various ensemble methods. These techniques required meticulous tuning to optimize performance. The process of fine-tuning these models was not only time-consuming but also required a deep understanding of the underlying data and the behaviour of each model. This challenge was particularly pronounced when trying to balance the trade-off between model complexity and generalizability.

## 4.2 Conclusion

The extensive analysis conducted in my dissertation demonstrates the varying levels of proficiency of different models in handling the dataset. Among these, the MLP-6L model emerged as the most effective, especially when employing SMOTE to balance the dataset. This model achieved the highest scores across critical metrics like accuracy, precision, recall, and the F1 score, underscoring its robustness in dealing with both classes, especially the minority class. This was a significant finding as it illustrated the potential of MLP models in predictive analytics in the financial domain.

However, it's important to note that while other models like LSTM, SVM, and different DBN configurations showed promising results under certain conditions (e.g., with specific SMOTE ratios), they generally fell short of the performance exhibited by the MLP-6L model. This outcome underscores the importance of addressing the class imbalance in predictive modeling. It also highlights the efficacy of ensemble and deep learning techniques in financial distress prediction, albeit with the critical caveat of careful hyperparameter tuning and thoughtful model selection.

Furthermore, my research revealed that there's a fine line between model complexity and overfitting, especially when dealing with high-performing benchmark models. These findings call for a cautious approach to model development and validation, ensuring not just statistical efficiency but also practical applicability and ethical responsibility.

## 4.3 Future Work

Looking ahead, my future research will delve into several promising areas. First, I plan to investigate more sophisticated techniques to handle class imbalance beyond SMOTE. This includes exploring advanced oversampling methods or novel cost-sensitive learning approaches, which could potentially enhance model performance further. Another avenue is the exploration of integrating additional or alternative financial health indicators. This could provide a more nuanced understanding of financial distress and potentially improve predictive accuracy.

Moreover, the potential of emerging machine learning techniques, such as hybrid models that combine elements of different algorithms, is an exciting prospect. Such hybrid models could leverage the strengths of various approaches to yield superior results. Additionally, extending the analysis to datasets from other regions or sectors could provide a more comprehensive understanding of financial distress prediction on a global scale.

Finally, incorporating real-time data analysis and exploring the application of these models in dynamic settings, as opposed to static ones, could open new frontiers in predictive accuracy and real-world applicability. This would not only enhance the relevance of the research but also ensure that the models developed are robust, fair, and useful in practical scenarios. My commitment remains to build models that are not just statistically sound but also ethically responsible and effective in real-world applications.

# REFERENCES

Ahmadpour Kasgari, A., Divsalar, M., Javid, M.R. and Ebrahimian, S.J. (2012). Prediction of bankruptcy Iranian corporations through artificial neural network and Probit-based analyses. Neural Computing and Applications, 23(3-4), pp.927–936. doi: https://doi.org/10.1007/s00521-012-1017-z.

Abdelaziz Testas (2023). Distributed Machine Learning with PySpark. Apress eBooks. doi: https://doi.org/10.1007/978-1-4842-9751-3.

Adaboost (n.d.). sklearn.ensemble.AdaBoostClassifier — scikit-learn 0.22.1 documentation. [online] scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html.

Ahmad, I., Imran, M., Qayyum, A., Ramzan, M.S. and Alassafi, M.O. (2023). An Optimized Hybrid Deep Intrusion Detection Model (HD-IDM) for Enhancing Network Security. Mathematics, [online] 11(21), p.4501. doi: https://doi.org/10.3390/math11214501.

Al-Badarneh, I., Habib, M., Aljarah, I. and Faris, H. (2020). Neuro-evolutionary models for imbalanced classification problems. Journal of King Saud University - Computer and Information Sciences. [online] doi: https://doi.org/10.1016/j.jksuci.2020.11.005.

Aljawazneh, H., Mora, A., García-Sánchez, P. and Castillo-Valdivieso, P. (2021). Comparing the Performance of Deep Learning Methods to Predict Companies’ Financial Failure | IEEE Journals & Magazine | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/9467268.

Aljawazneh, H.R. (2022). New approaches to improve the performance of machine learning and deep learning algorithms in solving real-world problems: companies financial failure forecasting. [online] digibug.ugr.es. Universidad de Granada. Available at: https://digibug.ugr.es/handle/10481/72340.

Altamirano-Flores, J.S., Alvarado-Hernández, L.Á., Cuevas-Tello, J.C., Tino, P., Guerra-Palomares, S.E. and Garcia-Sepulveda, C.A. (2023). Identification of Clinically Relevant HIV Vif Protein Motif Mutations through Machine Learning and Undersampling. Cells, [online] 12(5), p.772. doi: https://doi.org/10.3390/cells12050772.

Altman, E.I. (1968). Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. The Journal of Finance, [online] 23(4), pp.589–609. Available at: https://www.jstor.org/stable/2978933.

Analytics Vidhya. (2022). An Overview of Deep Belief Network (DBN) in Deep Learning. [online] Available at: https://www.analyticsvidhya.com/blog/2022/03/an-overview-of-deep-belief-network-dbn-in-deep-learning/.

Ansari, Said, I., Abu, A. and Ridzwan, M.R. (2020). A Hybrid Metaheuristic Method in Training Artificial Neural Network for Bankruptcy Prediction | IEEE Journals & Magazine | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/9204967.

Asemi, A. (2023). A Novel Combined Investment Recommender System Using Adaptive Neuro-Fuzzy Inference System - BCE Doktori disszertációk. Uni-corvinus.hu. [online] doi: https://phd.lib.uni-corvinus.hu/1310/1/Asefeh\_Asemi\_den.pdf.

Azadeh Mokari, Guo, S. and Bocklitz, T. (2023). Exploring the Steps of Infrared (IR) Spectral Analysis: Pre-Processing, (Classical) Data Modelling, and Deep Learning. Molecules, 28(19), pp.6886–6886. doi: https://doi.org/10.3390/molecules28196886.

Bagheri, E. and Cheung, J. (2018). Advances In Artificial Intelligence [PDF] [6cpfvikpmt60]. [online] vdoc.pub. Available at: https://vdoc.pub/documents/advances-in-artificial-intelligence-6cpfvikpmt60.

Beiranvand, V., Mobasher-Kashani, M. and Abu Bakar, A. (2014). Multi-objective PSO algorithm for mining numerical association rules without a priori discretization. Expert Systems with Applications, 41(9), pp.4259–4273. doi: https://doi.org/10.1016/j.eswa.2013.12.043.

Bellovary, J.L., Giacomino, D.E. and Akers, M.D. (2007). A Review of Bankruptcy Prediction Studies: 1930 to Present. Journal of Financial Education, [online] 33, pp.1–42. Available at: https://www.jstor.org/stable/41948574.

Breiman, L. (1996). Bagging predictors. Machine Learning, 24(2), pp.123–140.

Breiman, L. (2001). Random Forests. Machine Learning, [online] 45(1), pp.5–32. doi: https://doi.org/10.1023/a:1010933404324.

Brenes, R.F., Johannssen, A. and Chukhrova, N. (2022). An intelligent bankruptcy prediction model using a multilayer perceptron. Intelligent Systems with Applications, 16, p.200136. doi: https://doi.org/10.1016/j.iswa.2022.200136.

Brownlee, J. (2016). How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/.

Brownlee, J. (2019). Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

Brownlee, J. (2020). SMOTE for Imbalanced Classification with Python. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/.

Brozyna, J., Mentel, G. and Pisula, T. (2016). Statistical methods of the bankruptcy prediction in the logistics sector in Poland and Slovakia. [online] Available at: https://www.researchgate.net/profile/Tomasz-Pisula/publication/302588543\_STATISTICAL\_METHODS\_OF\_THE\_BANKRUPTCY\_PREDICTION\_IN\_THE\_LOGISTICS\_SECTOR\_IN\_POLAND\_AND\_SLOVAKIA/links/5731bb6108aed286ca0eb9a1/STATISTICAL-METHODS-OF-THE-BANKRUPTCY-PREDICTION-IN-THE-LOGISTICS-SECTOR-IN-POLAND-AND-SLOVAKIA.pdf.

Cheng, R. and Jin, Y. (2015). A Competitive Swarm Optimizer for Large Scale Optimization | IEEE Journals & Magazine | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/6819057.

Chollet, F. (2020). The Sequential model. [online] keras.io. Available at: https://keras.io/guides/sequential\_model/.

Chomboon, K., Chujai, P., Teerarassammee, P., Kerdprasop, K. and Kerdprasop, N. (2015). An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm. The Proceedings of the 2nd International Conference on Industrial Application Engineering 2015. doi: https://doi.org/10.12792/iciae2015.051.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. Machine Learning, [online] 20(3), pp.273–297. doi: https://doi.org/10.1007/bf00994018.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), pp.21–27. doi: https://doi.org/10.1109/tit.1967.1053964.

Devi, S.S. and Radhika, Y. (2018). A Survey on Machine Learning and Statistical Techniques in Bankruptcy Prediction. International Journal of Machine Learning and Computing, 8(2), pp.133–139. doi: https://doi.org/10.18178/ijmlc.2018.8.2.676.

Ding, B. and Eng, B. (2023). RADAR AND CAMERA FUSION IN INTELLIGENT TRANSPORTATION SYSTEM. [online] Available at: https://macsphere.mcmaster.ca/bitstream/11375/29005/2/Ding\_BaoMing\_202308.pdf.

Do Prado, J.W., Carvalho, F. de M., Benedicto, G.C. de and Lima, A.L.R. (2019). Analysis of credit risk faced by public companies in Brazil: an approach based on discriminant analysis, logistic regression and artificial neural networks. Estudios Gerenciales, pp.347–360. doi: https://doi.org/10.18046/j.estger.2019.153.3151.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory | IEEE Conference Publication | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/494215.

Fedorova, E., Gilenko, E. and Dovzhenko, S. (2013). Bankruptcy prediction for Russian companies: Application of combined classifiers. Expert Systems with Applications, 40(18), pp.7285–7293. doi: https://doi.org/10.1016/j.eswa.2013.07.032.

Freund, Y. and Schapire, R. (1996). Experiments with a New Boosting Algorithm DRAFT -PLEASE DO NOT DISTRIBUTE. [online] Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=d186abec952c4348870a73640bf849af9727f5a4.

Hassani, S., Dackermann, U., Mousavi, M. and Liu, H. (2024). A systematic review of data fusion techniques for optimized structural health monitoring. Information Fusion, 103, pp.102136–102136. doi: https://doi.org/10.1016/j.inffus.2023.102136.

Hassanien, A.-E., Chang, K.-C. and Mincong, T. (2021). Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2021 [1 ed.] 3030697169, 9783030697167. [online] dokumen.pub. Available at: https://dokumen.pub/advanced-machine-learning-technologies-and-applications-proceedings-of-amlta-2021-1nbsped-3030697169-9783030697167.html.

Hatcher, W.G. and Yu, W. (2018). A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. IEEE Access, [online] 6, pp.24411–24432. doi: https://doi.org/10.1109/access.2018.2830661.

He, H., Bai, Y., Garcia, E. and Li, S. (n.d.). ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. [online] Available at: https://sci2s.ugr.es/keel/pdf/algorithm/congreso/2008-He-ieee.pdf.

Hinton, G.E., Osindero, S. and Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. Neural Computation, 18(7), pp.1527–1554. doi: https://doi.org/10.1162/neco.2006.18.7.1527.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735–1780. doi: https://doi.org/10.1162/neco.1997.9.8.1735.

Hosaka, T. (2019). Bankruptcy prediction using imaged financial ratios and convolutional neural networks. Expert Systems with Applications, 117(1), pp.287–299. doi: https://doi.org/10.1016/j.eswa.2018.09.039.

https://vegibit.com/. (n.d.). Random Forests and Gradient Boosting In Scikit-learn. [online] Available at: https://vegibit.com/random-forests-and-gradient-boosting-in-scikit-learn/ [Accessed 8 Jan. 2024].

Huilgol, P. (2020). Precision vs Recall | Precision and Recall Machine Learning. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/.

Islam, S.R., Eberle, W., Ghafoor, S.K., Bundy, S.C., Talbert, D.A. and Siraj, A. (2019). Investigating bankruptcy prediction models in the presence of extreme class imbalance and multiple stages of economy. [online] arXiv.org. doi: https://doi.org/10.48550/arXiv.1911.09858.

Jang, Y., Jeong, I. and Cho, Y.K. (2020). Business Failure Prediction of Construction Contractors Using a LSTM RNN with Accounting, Construction Market, and Macroeconomic Variables. Journal of Management in Engineering, 36(2), p.04019039. doi: https://doi.org/10.1061/(asce)me.1943-5479.0000733.

Jang, Y., Jeong, I.-B., Cho, Y.K. and Ahn, Y. (2019). Predicting Business Failure of Construction Contractors Using Long Short-Term Memory Recurrent Neural Network. Journal of Construction Engineering and Management, 145(11), p.04019067. doi: https://doi.org/10.1061/(asce)co.1943-7862.0001709.

Jones, S. and Hensher, D.A. (2004). Predicting Firm Financial Distress: A Mixed Logit Model. The Accounting Review, 79(4), pp.1011–1038. doi: https://doi.org/10.2308/accr.2004.79.4.1011.

Kfold (2018). sklearn.model\_selection.KFold — scikit-learn 0.20.3 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.KFold.html.

Khyati Mahendru, Garg, G., Sharma, A. and Srivastava, R. (2021). Evolutionary Methods for Bankruptcy Prediction: A Study on Indian Firms. Advances in intelligent systems and computing, pp.303–313. doi: https://doi.org/10.1007/978-981-16-2709-5\_23.

Kim, M.-J. and Kang, D.-K. (2010). Ensemble with neural networks for bankruptcy prediction. Expert Systems with Applications, 37(4), pp.3373–3379. doi: https://doi.org/10.1016/j.eswa.2009.10.012.

Kim, M.-J., Kang, D.-K. and Kim, H.B. (2015). Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction. Expert Systems with Applications, 42(3), pp.1074–1082. doi: https://doi.org/10.1016/j.eswa.2014.08.025.

Lanbouri, Z. and Achchab, S. (2015). A hybrid Deep belief network approach for Financial distress prediction | IEEE Conference Publication | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/7358416.

Le, T., Lee, M., Park, J. and Baik, S. (2018). Oversampling Techniques for Bankruptcy Prediction: Novel Features from a Transaction Dataset. Symmetry, 10(4), p.79. doi: https://doi.org/10.3390/sym10040079.

Liang, D., Lu, C.-C., Tsai, C.-F. and Shih, G.-A. (2016). Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. European Journal of Operational Research, 252(2), pp.561–572. doi: https://doi.org/10.1016/j.ejor.2016.01.012.

López Iturriaga, F.J. and Sanz, I.P. (2015). Bankruptcy visualization and prediction using neural networks: A study of U.S. commercial banks. Expert Systems with Applications, 42(6), pp.2857–2869. doi: https://doi.org/10.1016/j.eswa.2014.11.025.

MIN, J. and LEE, Y. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. Expert Systems with Applications, [online] 28(4), pp.603–614. doi: https://doi.org/10.1016/j.eswa.2004.12.008.

Mirjalili, S. and Sadiq, A.S. (2011). Magnetic Optimization Algorithm for training Multi Layer Perceptron | IEEE Conference Publication | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/6014845.

Nakisa, B., Ahmad Nazri, M.Z., Rastgoo, M.N. and Abdullah, S. (2014). A survey: Particle Swarm Optimization based algorithms to solve premature convergence problem. Journal of Computer Science, [online] 10(9), pp.1758–1765. Available at: https://eprints.qut.edu.au/85241/.

Ohlson, J.A. (1980). Financial Ratios and the Probabilistic Prediction of Bankruptcy. Journal of Accounting Research, 18(1), pp.109–131. doi: https://doi.org/10.2307/2490395.

Pal, S. (2023). Deep Learning Architectures and their pivotal role in big data processing. www.agpegondwanajournal.co.in.

Paralic, M., Zelenak, K., Kamencay, P. and Hudec, R. (2023). Automatic Approach for Brain Aneurysm Detection Using Convolutional Neural Networks. Applied Sciences, [online] 13(24), p.13313. doi: https://doi.org/10.3390/app132413313.

Patle, A. and Chouhan, D.S. (2013). SVM kernel functions for classification. [online] IEEE Xplore. doi: https://doi.org/10.1109/ICAdTE.2013.6524743.

Pei, J., Tseng, V.S., Cao, L., Hiroshi Motoda and Xu, G. (2009). Advances in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science. doi: https://doi.org/10.1007/978-3-642-01307-2.

Pompe, P.P.M. and Feelders, A.J. (1997). Using Machine Learning, Neural Networks, and Statistics to Predict Corporate Bankruptcy. Computer-Aided Civil and Infrastructure Engineering, 12(4), pp.267–276. doi: https://doi.org/10.1111/0885-9507.00062.

Przystalski, K. and Thanki, R.M. (2024). Explainable Machine Learning in Medicine. Synthesis lectures on engineering, science, and technology. doi: https://doi.org/10.1007/978-3-031-44877-5.

Ruba Abu Khurma, Ibrahim Aljarah, Sharieh, A. and Seyedali Mirjalili (2019). EvoloPy-FS: An Open-Source Nature-Inspired Optimization Framework in Python for Feature Selection. Algorithms for intelligent systems, pp.131–173. doi: https://doi.org/10.1007/978-981-32-9990-0\_8.

Ruslan Salakhutdinov and Murray, I. (2008). On the quantitative analysis of deep belief networks. International Conference on Machine Learning. doi: https://doi.org/10.1145/1390156.1390266.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, pp.85–117. doi: https://doi.org/10.1016/j.neunet.2014.09.003.

scikit learn SVM (2018). 1.4. Support Vector Machines — scikit-learn 0.20.3 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/svm.html.

Scikit-Learn- StandardScaler (2019). sklearn.preprocessing.StandardScaler — scikit-learn 0.21.2 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

scikit-learn.org. (n.d.). sklearn.ensemble.BaggingClassifier — scikit-learn 0.23.1 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html.

Scikit-learn-Random-Forest (2018). sklearn.ensemble.RandomForestClassifier — scikit-learn 0.20.3 documentation. [online] Scikit-learn.org. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

Tam, A. (2021). A Gentle Introduction to Particle Swarm Optimization. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/.

Tayarani-N, M.H. and Akbarzadeh-T, M.R. (2008). Magnetic Optimization Algorithms a new synthesis | IEEE Conference Publication | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/4631155.

Tayarani-N., M.-H. . and Akbarzadeh-T., M.-R. . (2014). Magnetic-inspired optimization algorithms: Operators and structures. Swarm and Evolutionary Computation, 19, pp.82–101. doi: https://doi.org/10.1016/j.swevo.2014.06.004.

TSAI, C. and WU, J. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. Expert Systems with Applications, [online] 34(4), pp.2639–2649. doi: https://doi.org/10.1016/j.eswa.2007.05.019.

Veganzones, D. and Séverin, E. (2018). An investigation of bankruptcy prediction in imbalanced datasets. Decision Support Systems, 112, pp.111–124. doi: https://doi.org/10.1016/j.dss.2018.06.011.

Yeh, S.-H., Wang, C.-J. and Tsai, M.-F. (2015). Deep belief networks for predicting corporate defaults | IEEE Conference Publication | IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/abstract/document/7346197 [Accessed 8 Jan. 2024].

Zhou, L. (2013). Performance of corporate bankruptcy prediction models on imbalanced dataset: The effect of sampling methods. Knowledge-Based Systems, 41, pp.16–25. doi: https://doi.org/10.1016/j.knosys.2012.12.007.

# APPENDIX

1. Libraries Imported (Abdelaziz Testas, 2023)

import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
import warnings  
warnings.filterwarnings(action="ignore")

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier  
from sklearn.ensemble import BaggingClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score, classification\_report  
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.neural\_network import BernoulliRBM  
from sklearn.preprocessing import StandardScaler

from imblearn.over\_sampling import SMOTE

from imblearn.over\_sampling import ADASYN

import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense  
from tensorflow.keras.layers import LSTM  
  
from sklearn import metrics  
from sklearn.metrics import confusion\_matrix  
from sklearn.datasets import make\_classification  
from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score

from sklearn.metrics import roc\_curve, auc

**2. Code Files**

All the developed code models for our Ensemble, Deep Learning and Artificial Neural Networks have been uploaded to One Drive with shared access.

[K2279745\_Darpan\_Agarwal\_Dissertation\_Code\_Files](https://kingstonuniversity-my.sharepoint.com/:f:/g/personal/k2279745_kingston_ac_uk/EqaFFpk_w1JFuJFDwI0qbPIBKoBtjV_mHrf07gKzd3gFaw?e=0I7bBp)