Loan Eligibility Prediction Using Deep Learning

Melat Fekadu Gemeda

A Thesis Proposal Submitted to Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Office of Graduate Studies

Adama Science and Technology University

June 2023

Adama, Ethiopia

Loan Eligibility Prediction Using Deep Learning

Melat Fekadu Gemeda

Advisor: Dr. Rajesh Sharma

A Thesis Proposal Submitted to Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Office of Graduate Studies

Adama Science and Technology University

June 2023

Adama, Ethiopia

APPROVAL PAGE

I, the advisors of the thesis entitled "**Loan Eligibility Prediction Using Deep Learning**" and developed by Melat Fekadu Gemeda, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Rajesh Sharma	_____	_____
Major Advisor/Supervisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by Melat Fekadu Gemeda have read and evaluated the thesis entitled "**Loan Eligibility Prediction Using Deep Learning**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master's of Science in Computer Science and Engineering.

_____	_____	_____
Chair Person	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____	_____	_____
Department Head	Signature	Date
_____	_____	_____
School Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

DECLARATION

I hereby declare that this Master's Thesis entitled "**Loan Eligibility Prediction Using Deep Learning** " is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Melat Fekadu Gemeda _____ _____

Name of Student Signature Date

RECOMMENDATION OF ADVISORS/SUPERVISORS

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Loan Eligibility Prediction Using Deep Learning**" prepared under my guidance by Melat Fekadu Gemeda. submitted in partial fulfillment of the requirements for the degree of Master's of Science in Computer Science and Engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. Rajesh Sharma	_____	_____
Major Advisor/Supervisor	Signature	Date

ACKNOWLEDGMENT

All glory and honor go to God and His blessing on the successful completion of this thesis. I give God thanks for all of the opportunities, potentials, and strength that allowed on me to complete the thesis. I went through a lot during this process, both academically and in terms of my personality.

First, I want to express thanks to Dr. Rajesh Sharma, my advisor, for his advice, patience, and understanding. Most significantly, he has given me the motivation I need to accomplish this thesis. Having him as my advisor has been a wonderful opportunity and privilege.

And the deepest gratitude for my husband for his worth comments, guidance, and kind support on those days when things get hard and seems impossible.

Table of Contents

List of Figures

List of Tables

List of Equations

ABBREVIATIONS

Adam	Adaptive Moment Estimation
BCO	Border Collie Optimization
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DbCL	Database Contents License
DL	Deep Learning
Deep RNN	Deep Recurrent Neural Network
Fuzzy NN	Fuzzy neural network
GAN	Generative Adversarial Networks
GPU	Graphics Processing Unit
IEFSVM	Instance-based Entropy Fuzzy Support Vector Machine
ML	Machine Learning
Multi PLS	Multiple Partial Least Squares Regression Model
NB	Naive Bayes
NN	Neural Network
Relu	Rectified Linear Unit
RNN	Recurrent Neural Network
RMSE	Root Mean Square Error
SBCO	Social Border Collie Optimization
SMOTE	Synthetic Minority Oversampling Technique
SSD	Social Ski Driver
SVM	Support Vector Machine
UML	Unified Modeling Language
WSOA-based	Whale Social Optimization Algorithm-based

ABSTRACT

A loan is a financial transaction in which one party, often a lender, lends money, commodities, or services to another party, known as the borrower, with the expectation of repayment in the future. Loans are often made with the idea that the borrower would repay the loan amount plus any relevant interest or fees over a set period of time. And the success and failure of these lending sectors depend on the ability to evaluate the credit risk, as it has become a significant role of financial institutions/banking sector to sanction loans. Based on their requirements and business rules they approved the loan after doing the all process manually which is time taking and is also Inefficient. Automating the process to identify loan eligibility is an efficient way to reduce the amount of time it takes and the credit risk. So, predicting if the loan applicant is eligible or not helps the banks to decide to start/not to start the sanctioning loan process. There are a number of researches done in this area and each of them played a vital role in contributing something important by developing the best model to predict the eligible applicant. But most of them used traditional ML classification algorithms which are not latest algorithms when compared to deep learning algorithms for prediction. Convolutional Recurrent Neural Network (CRNN) is preferable for prediction eligible applicant from the given loan data-set, which is the credit history of bank's or lending authority's customers by extracting and learning complex and relevant features. In this thesis study, the proposed model is to predict loan applicant using CRNN. Our dataset, which contains 461097 rows and 44 columns of lending club data-set collected from kaggle online data-set, is used to train and evaluate the CRNN model. The application of CRNN improves the performance of several machine-learning classification methods significantly. Our proposed CRNN model has an accuracy of 99.67% and loss of 0.0081. In comparison to earlier research works on loan eligibility prediction, we were able to attain higher prediction accuracy using the developed CRNN model.

Keywords: CRNN, Loan eligibility Prediction, Deep Learning,

CHAPTER ONE

1 INTRODUCTION

1.1 Background of study

The process of sanctioning loans is a crucial aspect for lending authorities such as banks and credit institutions. The ability to accurately evaluate credit risk plays a significant role in determining the success or failure of these institutions. However, determining the eligibility of loan applicants is a complex and time-consuming task. Lending authorities typically follow manual methods and rely on their own requirements and business rules to approve loans, resulting in inefficiencies and delays.

In recent years, the banking sector has embraced modern technologies to enhance processes, improve productivity, and reduce costs. Machine learning (ML) applications, particularly those utilizing predictive analytics, have gained significant attention and have been widely used in the banking industry. According to a recent study (Vedapradha & Hariharan, 2022)**,** ML applications using predictive analytics were among the most utilized features in the banking sector worldwide in 2020.

To address the challenges of loan eligibility assessment, numerous research studies have been conducted using various ML classification algorithms. These studies have aimed to develop models that can effectively predict whether a loan applicant is eligible or not. Each research endeavor has contributed valuable insights and methodologies to enhance the efficiency and accuracy of loan eligibility prediction.

Despite the existing body of research, there has been limited exploration of deep learning techniques specifically for loan eligibility prediction. Deep learning, a subfield of ML, involves the use of artificial neural networks with multiple layers to extract complex features and patterns from data. Deep learning models have demonstrated remarkable performance in various domains, such as computer vision, natural language processing, and speech recognition. Considering the potential benefits of deep learning in loan eligibility prediction, this study aims to fill the research gap by developing a deep learning-based model for accurately determining loan eligibility.

By leveraging the power of deep neural networks, the proposed model can potentially provide more accurate and reliable predictions compared to traditional ML algorithms. The digitization of the loan eligibility assessment process through deep learning can significantly reduce processing time, enhance efficiency, and mitigate credit risk for lending authorities.

Overall, this study seeks to contribute to the existing body of knowledge by exploring the application of deep learning in loan eligibility prediction. By examining the performance of a deep learning model and comparing it with traditional ML algorithms, this research aims to provide insights into the potential benefits and limitations of deep learning in the banking sector. The findings of this study can assist lending authorities in adopting more efficient and accurate loan eligibility assessment methods, thereby improving their decision-making processes and ultimately benefiting loan applicants and the financial ecosystem as a whole.

1.2 Motivation of the Study

People can apply for bank loans to finance their investment decisions thanks to modernization in the banking industry (Infant Cyril et al., 2022).

The bank only has a small amount of assets; thus, those assets must be given to borrowers who can adhere to the repayment schedule.

- **Technological Advancements:** The advancements in deep learning and artificial intelligence have shown remarkable potential in various domains. By exploring the application of deep learning in loan eligibility prediction, this study aims to harness the power of these technologies to enhance the accuracy and efficiency of the loan assessment process.
- **Industry Demand:** The banking sector and lending authorities are constantly seeking innovative solutions to improve their loan approval processes. By adopting deep learning techniques, these institutions can potentially reduce manual efforts, processing time, and costs, while also improving the accuracy of loan eligibility decisions.
- **Mitigating Credit Risk:** The accurate evaluation of credit risk is crucial for financial institutions. Deep learning models have the potential to analyze vast amounts of data, including complex patterns and relationships, which can aid in identifying high-risk loan applicants. By effectively predicting loan eligibility, lending authorities can mitigate credit risk and make informed decisions on loan approvals.
- **Efficiency and Cost Reduction:** The manual evaluation of loan eligibility can be time-consuming and inefficient, often leading to delays and increased operational costs for lending authorities. By digitizing and automating the loan eligibility assessment process through deep learning, institutions can significantly reduce processing time, streamline operations, and improve overall efficiency.

- **Research Gap:** While several studies have explored loan eligibility prediction using traditional machine learning algorithms, there is a gap in research specifically focusing on the application of deep learning techniques. This study aims to bridge that gap and provide insights into the performance and potential advantages of deep learning models in the domain of loan eligibility assessment.

By addressing these motivations, this study can contribute to the advancement of loan assessment methodologies, benefiting both lending authorities and loan applicants. The findings can help financial institutions make more accurate and efficient loan decisions, leading to improved customer satisfaction, reduced credit risk, and optimized resource allocation.

1.3 Statement of the Problem

The prediction of loan eligibility plays a crucial role in the financial industry, aiding in efficient decision-making and risk assessment. Banks or lending authorities often faces problems like handling many loan applications manually (Inefficient in terms of time and cost) which results many approval delays and lack of customer satisfaction by this loan assessment process. And the other one is loan risk assessment and knowing the customer creditworthiness or assessing eligibility of applicant is challenging but also at the same time very crucial task for this lending sectors. Many machine learning approaches have been extensively explored in this domain; there is a limited body of research focused on leveraging deep learning techniques for loan eligibility prediction. This research paper aims to address this gap by applying deep learning models, including GRU, and Conv1D, in predicting loan eligibility.

The primary problem to be addressed is two-fold. First, there exists a scarcity of research utilizing deep learning models for loan eligibility prediction, with only a single study found in the literature. This limitation highlights the need to explore the potential of deep learning techniques in this domain and determine their effectiveness in comparison to traditional machine learning methods.

Secondly, the objective is to improve (Infant Cyril et al., 2022) their model accuracy in loan eligibility prediction by using different methodology like using different preprocessing technique and different model architecture. Although this study (CRNN based loan eligibility prediction) demonstrates promising results, it is essential to investigate whether alternative deep learning models can surpass the existing accuracy (SBCO model with 95% accuracy) while accounting for the given data-sets, feature representations, and preprocessing techniques.

Banks and lending authorities, by using the proposed model can save time and money by automating and streamlining the loan evaluation process, which results in cost reductions. And also

benefited from this model since this model enables more rapid loan approvals, increased efficiency, and improved client satisfaction, as well as successfully managing risk and reducing losses by predicting potential dangers and patterns of default.

1.4 Research questions

In this study, an attempt will be made to answer the following questions.

RQ1. How can the performance of the Loan Eligibility Prediction be improved using Deep Learning?

RQ2. What parameters and hyper parameters improve the prediction model?

RQ3. What problems can be addressed using this deep learning-based model?

RQ4. How the effectiveness and productivity banks and lending sectors can be improved using this (CRNN) model?

RQ5. How will banks and lending authorities be benefited from this (CRNN) model?

1.5 Objective

1.5.1General objective

The general objective of the study is to implement Deep Learning based model for Loan Eligibility Prediction.

1.5.2Specific Objective

The following specific objectives are addressed to achieve the general objective.

- Collect and preprocess a loan dataset, including relevant financial and applicant information, to train and evaluate the Deep Learning-based CRNN model for Loan Eligibility Prediction
- Implement the architecture of the CRNN model, considering the specific loan eligibility prediction task, to improve the accuracy and reliability of loan eligibility predictions
- Perform extensive experimentation and hyper-parameter tuning to identify the optimal configuration of the CRNN model, including learning rate, batch size, number of convolutional and recurrent layers, and layer sizes, for achieving better loan eligibility prediction performance.
- Evaluate the performance of the CRNN model against benchmark models (**(Infant Cyril & Ananth, 2022)**), using appropriate evaluation metrics such as accuracy, confusion-matrix
- Assess the impact the CRNN model on banking and lending sectors how it helps to increases the effectiveness and productivity on factors such as reduction in manual efforts, and improved decision-making processes

- Assess the customers benefits and advantages they perceive from the CRNN model for loan eligibility prediction, in the banking and lending sectors to in terms of improved loan-related services, faster decision-making, and enhanced customer experience

1.6 Scope and Limitations of the Study

1.6.1 Scope of the Study

The scope of the thesis work within the given time and resource includes: -

- Predict Eligible loan applicant using deep learning approaches to extract features from the data.
- It uses the deep learning approach.

1.6.2 Limitations of the study

The biggest limitation for this study is, there are only a few researches done by leveraging deep learning on this topic (Loan eligibility Prediction) and also on topics related to this topic.

The other one is the limited source of our countries loan data set. I have tried to collect the data-set from some of our countries banks and lending companies, but they were not willing to give the data-set even if the saw the proposal of this study because of some security reasons.

1.7 Significance of the study

Studying loan eligibility prediction using deep learning holds several significant implications and potential benefits. Here are some key points highlighting the significance of this research area:

- **Enhanced Accuracy:** Deep learning models have demonstrated superior performance in various domains, particularly in tasks involving complex patterns and large datasets. By applying deep learning techniques to loan eligibility prediction, there is a potential to achieve higher accuracy compared to traditional machine learning methods. This can lead to more precise and reliable loan eligibility assessments, reducing the chances of false positives or false negatives.
- **Improved Risk Assessment:** Accurate loan eligibility prediction is crucial for financial institutions to assess the risk associated with lending decisions. Deep learning models can effectively capture intricate patterns and dependencies within loan data, allowing for a more comprehensive risk assessment. This can lead to better-informed decisions, reduced default rates, and improved portfolio management.

- **Scalability and Generalization:** Deep learning models can handle large-scale datasets and generalize well to unseen data. This scalability and generalization capability can be beneficial in loan eligibility prediction, where data volumes can be substantial, and the models need to perform well on new loan applications. Deep learning models can leverage their capacity to learn from large amounts of data and generalize patterns to make accurate predictions for new loan applicants.
- **Potential for Automated Decision-making:** The use of deep learning models in loan eligibility prediction can potentially lead to the automation of decision-making processes in financial institutions. By developing robust and accurate deep learning models, it becomes feasible to create automated systems that can quickly and efficiently assess loan eligibility, reducing manual effort and processing time.

1.8 Organization of the Thesis

This research work is divided into seven chapters, which are briefly discussed below.

Chapter One: In this chapter, the research project's outline and the explanations for how and what will be achieved are explained.

Chapter Two: The background literature and related work for deep learning, RNN, convolution layer, and loan eligibility prediction are explained in this chapter.

Chapter Three: This chapter discusses methodologies for research, including collecting data, data preparation, design tools, development frameworks and platforms, and evaluation techniques.

Chapter Four: This chapter strives to provide thorough information on how the proposed solution attempts to address the problem by using various block diagrams, pseudo-codes, and flow charts of the implementation together with accompanying mathematical descriptions.

Chapter Five: This chapter explains how to implement the proposed problem-solving strategy into practice. It also includes a code snippet that describes the environment setup and the suggested solutions.

Chapter Six: In this chapter, the results of the proposed solution are discussed, and the findings are compared using figures, graphs, and tables.

Chapter Seven: This chapter provides a summary of the research. The work's conclusion is provided, along with ideas for future study.

CHAPTER TWO

2 LITERATURE REVIEW

2.1 Conceptual Background

Loan eligibility prediction research are done using different machine learning models to reducethe time to identify the eligible applicant for a loan with better efficiency.

The authors in**,** (Infant Cyril et al., 2022)deployed a new method, namely SBCO: - devised by combining the SSD algorithm and BCO -based deep neuro fuzzy network for loan eligibility prediction. In their method, the proposed method, called SBCO (which stands for SSD-Based COx transformation), aims to improve the efficiency and accuracy of loan eligibility prediction. The authors introduce a combination of the SSD algorithm, which is commonly used for object detection in computer vision, and a BCO (Box Cox Optimization)-based deep neuro fuzzy network.

To preprocess the input loan data, the authors employ the Box Cox transformation. This transformation is applied to the input data to make it more suitable for further processing. By transforming the data, it can exhibit a more normal distribution and reduce the impact of outliers. The transformed data is then subjected to feature selection using a wrapper approach. This means that the features that are most relevant to the loan eligibility prediction task are selected. The goal is to improve the efficiency of the loan eligibility calculation by using only the most informative features.

Once the appropriate features have been selected, the authors adapt the Naive Bayes (NB) algorithm for feature fusion. Feature fusion involves combining the selected features and groups of the input data to create a probability index table. This table captures the relationships between the features and the loan eligibility classes.

To evaluate the loan eligibility, the authors utilize the posterior probability ratio while considering the conditional probability of the normalization constant and class evidence. This approach takes into account the likelihood of observing the selected features given the loan eligibility class.

In comparison to other existing methods such as Fuzzy Neural Networks (NN), Multi PLS (Partial Least Squares), IEFSVM (Incremental Evolutionary Feature Selection with Support Vector Machines), Deep Recurrent Neural Networks (RNN), and WSOA (Weighted Sum Operator Aggregation)-based Deep RNN, the proposed SBCO method achieves the highest accuracy of 95% along with sensitivity and specificity rates of 95.4% and 97.3%, respectively.
The authors acknowledge that the use of wrapper feature selection may have contributed to the higher accuracy obtained by their model. However, they suggest that future research could explore hybrid feature selection techniques to further improve the accuracy and performance of loan eligibility prediction models.

According to,(Sheikh et al., 2020) they propose an initiative to automate the loan acceptance process by considering various factors, such as gender, education, and the number of dependents. The aim is to save time and effort and enhance the efficiency of the loan approval process. The input data for this initiative is divided into two sets: the train dataset and the test dataset. The train dataset is utilized to train a machine learning model, while the test dataset is used to evaluate the accuracy of the model and predict loan eligibility. To implement this solution, the researchers employ a Logistic Regression model based on machine learning techniques. Logistic Regression is a popular classification algorithm that is well-suited for predicting binary outcomes, such as loan eligibility (yes or no).
The implementation of the Logistic Regression model is carried out using the Scikit-learn library in Python. Scikit-learn is a widely-used machine learning library that provides various algorithms and tools for data preprocessing, model training, and evaluation. By training the Logistic Regression model on the train dataset, the researchers aim to capture the relationships between the input features (gender, education, number of dependents, etc.) and the loan eligibility outcome. The model learns from the patterns and information present in the train dataset to make predictions on new, unseen data.
Once the model is trained, it is evaluated using the test dataset. The accuracy of the model is assessed by comparing its predictions with the known loan eligibility outcomes in the test dataset. This evaluation step provides an indication of how well the model generalizes to new data and can accurately predict loan eligibility. The predicted loan eligibility, obtained from the trained Logistic Regression model, can assist in automating the loan acceptance process.
By leveraging machine learning techniques, this approach offers a data-driven solution that takes into account various factors to determine loan eligibility efficiently and effectively.

It's worth noting that while the paper focuses on Logistic Regression as the chosen model, there are various other machine learning algorithms that could also be explored for loan eligibility prediction, depending on the specific requirements and characteristics of the dataset.

The research paper by (Orji et al., 2022) presents six different machine learning techniques for predicting loan eligibility. These techniques include Random Forest, Gradient Boost, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbor (KNN), and Logistic Regression. To train the models, the researchers utilize the "Loan Eligible Dataset," which is a historical dataset accessible on Kaggle and licensed under DbCL v1.0. This dataset contains information related to loan applications, including various features that can influence loan eligibility.
During the pre-processing stage, the researchers apply normalization and SMOTE (Synthetic Minority Over-sampling Technique) techniques to address specific challenges in the dataset. Normalization is employed to ensure that all features are on the same scale, which can help improve the performance of certain machine learning algorithms. SMOTE, on the other hand, is utilized to address the issue of class imbalance in the dataset. Class imbalance occurs when the number of instances in one class is significantly smaller than the other, which can negatively impact the performance of machine learning models. The researchers use Python programming libraries in Kaggle's Jupyter Notebook cloud platform to process and analyze the dataset. Python offers a wide range of machine learning libraries, such as Scikit-learn, which provides implementations of various algorithms and tools for data preprocessing, model training, and evaluation. The findings of the research indicate that the machine learning techniques achieve significant performance accuracy in predicting loan eligibility. Among the six techniques, the Random Forest algorithm attains the highest accuracy of 95.55%. Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. Its ability to capture complex relationships in the data and handle feature interactions can contribute to its high accuracy. On the other hand, Logistic Regression achieves the lowest accuracy of 80% among the techniques considered. Logistic Regression is a linear model that is commonly used for binary classification tasks. While it may have achieved a lower accuracy compared to other techniques, it still provides a baseline performance for loan eligibility prediction. Overall, the research demonstrates the effectiveness of employing different machine learning techniques and preprocessing strategies, such as normalization and SMOTE, to predict loan eligibility using the provided dataset. It highlights the strengths and weaknesses of each technique in achieving accurate predictions.

(Shaik et al., n.d.) on their research focuses on determining the best machine learning algorithm for predicting loan eligibility. The main objective of the study is to identify the algorithm that performs the most accurately in determining whether a person is qualified for a loan or not. To achieve this goal, the authors conduct a comparative study by training several machine learning algorithms on collected data. The algorithms used in the study include the Random Forest Classifier, Passive Aggressive Classifier, Multinomial Naïve Bayes, Support Vector Classifier, and Adaboost Classifier. During the training process, the authors employ the k-fold cross-validation technique. This technique involves splitting the available data into k subsets or folds, training the models on k-1 folds, and validating the performance on the remaining fold. By repeating this process k times with different combinations of folds, a more robust estimate of the model's performance can be obtained.

By applying the k-fold cross-validation rule on the collected data, the authors assess the performance of each algorithm in accurately predicting loan eligibility. The analysis of the results reveals that the model created using the Random Forest Algorithm achieves the highest accuracy among all the models considered. Specifically, it accurately identifies the loan eligibility of individuals at a rate of 78%. The Random Forest Algorithm is an ensemble learning method that combines multiple decision trees to make predictions. It can handle a large number of features and capture complex relationships in the data, which likely contributes to its superior performance in this study.

It is important to note that the study does not provide detailed insights into the specific characteristics of the collected data or the preprocessing steps applied. Additionally, the accuracy rate of 78% achieved by the Random Forest Algorithm is context-specific and may not be directly comparable to results from other studies or datasets. Nonetheless, the research by Shaik et al. highlights the potential of machine learning algorithms in predicting loan eligibility. The findings suggest that the Random Forest Algorithm shows promise in accurately identifying loan qualification, but further research and evaluation on diverse datasets are necessary to validate and generalize these results.

(Singh et al., 2021) research proposes a novel machine learning method for predicting loan eligibility. The researchers created a classification algorithm that takes into account a variety of characteristics and several inputs to assess if a person is eligible for a loan. The study's major goal was to improve the accuracy of loan eligibility prediction by employing advanced

computer approaches. The researchers gathered a large dataset encompassing a variety of variables connected to loan applicants, such as credit history, income level, employment status, loan amount, and other pertinent information. They used a machine learning technique based on a classification algorithm to create the prediction model. The available dataset was used to train the algorithm, with loan eligibility serving as the target variable. The algorithm learns to distinguish between qualified and ineligible loan applicants by studying the patterns and correlations in the data. According to the study's findings, the built machine learning model was very accurate in predicting loan eligibility. The model performed admirably, correctly categorizing the vast majority of loan applicants. This shows that using machine learning algorithms to anticipate loan eligibility can considerably enhance the process. However, because to the complexity of the technique, the proposed model's implementation required more processing time. The researchers acknowledged this restriction and stressed the need for additional tuning to improve the model's computational efficiency.

The study by (A. & S, 2021) presents a model for predicting loan eligibility, where the performance of the model is evaluated based on accuracy, recall, the confusion matrix, and the root mean square error (RMSE) values. The study involves training the models using the train data and evaluating their performance using the test data. According to the results obtained from the research, both the Logistic Regression (LR) and Support Vector Machine (SVM) models demonstrate accurate predictions of loan eligibility. The computed RMSE values for both models are reported to be 81.62%. RMSE is a commonly used metric for evaluating the performance of regression models.

It measures the average difference between the predicted values and the actual values. In this context, the RMSE value indicates the level of error in the predictions made by the LR and SVM models for loan eligibility. While the paper highlights the RMSE values of the LR and SVM models, it is important to note that RMSE is typically used for regression tasks, where the predicted values are continuous variables. In the case of predicting loan eligibility, which is a classification task (determining whether an individual is eligible or not), other metrics like accuracy, recall, and the confusion matrix are more relevant for evaluating the model's performance.

Accuracy represents the proportion of correctly classified instances over the total number of instances. It provides an overall measure of how well the model predicts loan eligibility. Recall, on the other hand, focuses on the true positive rate, which measures the proportion of correctly

identified eligible instances among all actual eligible instances. The confusion matrix is a tabular representation that summarizes the performance of a classification model, showing the counts of true positive, true negative, false positive, and false negative predictions.

Unfortunately, the additional information on accuracy, recall, and the confusion matrix is not provided in the provided context. It is important to consider these metrics alongside the RMSE value to have a comprehensive understanding of the models' performance in predicting loan eligibility accurately. Further details on the experimental setup, dataset characteristics, and any preprocessing techniques used would provide a more comprehensive understanding of the research findings and the performance of the LR and SVM models in predicting loan eligibility.

(Al-qerem et al., 2019) propose a study focused on loan prediction models utilizing random forest classifiers and decision trees based on the naive Bayes algorithm in their research paper. The authors examine several data pre-processing procedures and feature selection techniques to improve loan forecast accuracy, notably for the minority class. The importance of data pre-processing approaches in boosting the performance of loan prediction models is emphasized by the researchers. They investigate several data preprocessing approaches and assess their impact on prediction accuracy.

They want to improve the forecast of the minority class by implementing a specialized pre-processing method, which is often critical in loan prediction scenarios. The integration of random forest classifiers with decision trees based on the naive Bayes algorithm is one of the key strategies used in their investigation. These classifiers are well-known for their ability to handle both category and numerical information in complex datasets. The authors stress the importance of comparing their pre-processing strategy to other feature selection strategies extensively utilized in the domain. They demonstrate the superiority of their customized pre-processing technique in reliably predicting the minority class through rigorous evaluation. It should be noted, however, that the study does not involve an examination of other classifiers trained on various bank datasets. This study's findings add to the existing research on loan prediction models by emphasizing the importance of adequate data pre-processing procedures. The study shows the potential benefits of integrating random forest classifiers and naive Bayes decision trees for accurate loan prediction. Further research could look into including more classifiers and evaluating their performance on different bank datasets, offering a more thorough review of loan prediction models.

Overall, Al-qerem et al.'s (2019) research gives useful insights into loan prediction modeling,

including advice on data pre-processing strategies and showing the potential of specific classifiers in this domain.

The study conducted by (S et al., 2021) focuses on building a machine learning-based Logistic Regression model as a solution for assessing a client's creditworthiness. The primary goal is to automate the process of determining whether or not to lend money to a borrower, eliminating the need for manual and labor-intensive work.

The Logistic Regression model is trained using relevant criteria that are essential for evaluating a client's creditworthiness. These criteria may include factors such as income, employment history, credit score, debt-to-income ratio, loan repayment history, and other relevant financial and personal information. By training the model on historical data that captures patterns and relationships between these criteria and the loan outcomes, the model learns to generate results with acceptable accuracy. This training process enables the model to make predictions based on the input information provided for a borrower. The advantage of using machine learning models like Logistic Regression is that they can handle large amounts of data and automatically identify patterns and relationships that may not be apparent through manual analysis. This allows for more efficient and accurate assessment of a borrower's creditworthiness.

Once the Logistic Regression model is trained, it can generate accurate results on whether or not to lend money to a borrower without the need for tiresome manual work. By inputting the relevant borrower information into the trained model, it can quickly and efficiently evaluate the borrower's creditworthiness and provide a prediction. Automating the process of creditworthiness assessment using machine learning models can significantly streamline and expedite lending decisions. It reduces the reliance on manual analysis, which can be time-consuming and prone to human biases. By leveraging the power of algorithms and data-driven analysis, lenders can make more informed and objective decisions regarding loan approvals. It is worth noting that the success of the Logistic Regression model in accurately predicting creditworthiness depends on the quality and relevance of the data used for training. Additionally, ongoing monitoring and validation of the model's performance are crucial to ensure its reliability and effectiveness in real-world lending scenarios.

2.2 Related Works

The research conducted by (Kuma et al., 2018) focuses on loan default prediction using a neural network model. The authors utilize a dataset provided by Lending Club bank, which is then employed to train and evaluate their proposed neural network model. To prepare the dataset for training, the researchers employ a dimensionality reduction technique called Principal Component Analysis (PCA). PCA is used to reduce the number of features in the dataset while retaining the most important information. By transforming the dataset into a lower-dimensional space, the computational complexity is reduced, and potential noise and redundancy in the data are mitigated.

Once the dataset is processed using PCA, the neural network model is trained on the transformed data. Neural networks are powerful machine learning models that can learn complex patterns and relationships in data. They are particularly effective for tasks such as loan default prediction, where there may be nonlinear relationships between the input features and the target variable. The model's performance is evaluated using metrics such as accuracy, which measures the proportion of correctly classified instances. The accuracy achieved by the proposed neural network model in this study is reported to be approximately 93%.

This level of accuracy is considered fairly high, especially when dealing with large datasets like the one provided by Lending Club bank. The high accuracy achieved by the neural network model suggests that it is capable of accurately predicting loan defaults based on the given features. This can be valuable for lenders and financial institutions as it enables them to assess the creditworthiness of borrowers and make informed decisions regarding loan approvals. It is important to note that while the accuracy of 93% is reported, additional information on other evaluation metrics, such as precision, recall, and the use of cross-validation techniques, would provide a more comprehensive understanding of the model's performance. Overall, the research by Kuma et al. demonstrates the efficacy of a neural network model for loan default prediction using the dataset from Lending Club bank. The utilization of dimensionality reduction techniques and the achievement of high accuracy highlights the potential of neural networks in assisting lenders in identifying potential defaulters and managing credit risk.

The paper conducted by (UCD Michael Smurfit Graduate Business School et al., 2018) focuses on the prediction of credit default using artificial neural networks and linear regression models. The researchers train both systems using loan lending data obtained from kaggle.com, which provides a comprehensive dataset for analysis.

The study compares the performance of artificial neural networks and linear regression models in predicting credit default. These models are trained on the loan lending data and evaluated based on their accuracy in making predictions. The accuracy achieved by the artificial neural network is reported to be 97.69609%, while the linear regression model achieves an accuracy of 97.67575%.

The high accuracy scores obtained by both models indicate their effectiveness in predicting credit default for loan applications. The similarity in accuracy between the artificial neural network and linear regression models suggests that both systems have a significant impact on the dataset and perform remarkably well. The researchers conclude that the artificial neural network provides strong evidence for efficiently predicting credit default in loan applications. This highlights the potential of neural network models in credit risk assessment and decision-making processes for lenders and financial institutions. The characteristics of the dataset and the preprocessing steps applied are not explicitly mentioned in the provided context.

However, the availability of a comprehensive dataset and the utilization of advanced modeling techniques, such as artificial neural networks, contribute to the overall effectiveness of the study. The findings of this research emphasize the importance of utilizing advanced machine learning techniques, such as artificial neural networks, in predicting credit default. The high accuracy scores achieved by both the artificial neural network and linear regression models demonstrate their potential in improving credit risk management and assisting lenders in making informed decisions regarding loan approvals.

The research conducted by (Handzic et al., 2003) focuses on the categorization of credit loan applications and aims to determine the most effective and accurate model among three neural network models: Multi-Layer Perceptron, Ensemble Averaging, and Boosting by Filtering.The objective of the study is to identify the neural network model that would be most useful in the specific decision scenario of categorizing credit loan applications. The researchers compare the performance of these three models by evaluating their effectiveness and accuracy.

The experimental findings reveal that the Committee Machine models, which include ensemble techniques such as Ensemble Averaging and Boosting by Filtering, outperformed a single Multi-Layer Perceptron model. This suggests that the combination of multiple neural network

models within a committee approach leads to improved results compared to using a single model. Furthermore, the study demonstrates that Boosting by Filtering performed better than Ensemble Averaging in terms of effectiveness and accuracy. Boosting is a technique that focuses on sequentially training weak models and emphasizing the misclassified instances to improve the overall performance. The Boosting by Filtering approach, in particular, exhibited superior performance compared to Ensemble Averaging.

These findings indicate that employing ensemble techniques, specifically Committee Machine models and Boosting by Filtering, can enhance the accuracy and effectiveness of credit loan application categorization. By leveraging the collective decision-making power of multiple models and emphasizing the strengths of Boosting, more accurate predictions can be achieved in this decision scenario. It is important to note that the specific details of the dataset used, the experimental setup, and the performance evaluation metrics are not provided in the given context. However, the results highlight the potential benefits of employing ensemble techniques and boosting in credit loan application categorization, thereby assisting financial institutions in making more informed decisions.

In the study conducted by (Elette et al., 2010) the focus is on the application of artificial neural networks as a tool for evaluating credit applications and supporting loan decisions in Jordanian commercial banks. The authors propose a model that utilizes a multi-layer feed-forward neural network with back-propagation learning. This type of neural network architecture is commonly used for pattern recognition and classification tasks, making it suitable for evaluating loan applications and making loan decisions. By training the neural network model on historical data, which likely includes information about borrowers' characteristics, financial history, and other relevant factors, the model learns to recognize patterns and relationships that can predict the likelihood of loan approval or rejection. The outcomes of the study reveal that artificial neural networks are successful technology when applied in Jordanian commercial banks for evaluating loan applications. This suggests that the proposed model using neural networks can effectively assess the creditworthiness of applicants and support the decision-making process in loan approvals. The advantage of using artificial neural networks is their ability to capture complex patterns and non-linear relationships in the data. This allows the model to consider multiple factors simultaneously and make predictions based on a holistic assessment of the borrower's creditworthiness.

By leveraging the power of artificial neural networks, Jordanian commercial banks can benefit

from more accurate and efficient loan evaluations. This can lead to improved risk management, better allocation of resources, and enhanced decision-making processes.It is important to note that the specific details regarding the dataset used and the performance metrics are not provided in the given context. Additionally, the study focuses on the application of artificial neural networks specifically in the context of Jordanian commercial banks, so the findings may not be directly generalizable to other regions or banking systems.

In their research (Xu & Xie, 2021) the authors focus on loan default prediction in the Chinese peer-to-peer (P2P) market. They evaluate the performance of several machine learning models, including Random Forest (RF), XGBoost, Gradient Boosting Machine (GBM), and neural network models.
By utilizing these models, the authors aim to accurately predict loan default in the P2P market. They compare the performance of the four models and find that the Random Forest model outperforms the others, achieving an accuracy rate of over 90% most of the time. This suggests that Random Forest is particularly effective in predicting loan default in the Chinese P2P market. While their study shares some similarities with yours in terms of the techniques and algorithms employed, there is a distinct difference in the goals. Xu et al. focus specifically on forecasting loan default in the P2P market, while your study aims to predict loan eligibility for clients.

Nonetheless, the utilization of machine learning models, such as Random Forest, demonstrates the potential of these techniques in the realm of credit assessment and risk prediction. In another related research conducted by (Meshref, 2020) the author explores the prediction of loan acceptance using bank direct marketing data. Various machine learning models, including AdaBoost, LogitBoost, Bagging, and Random Forest, are employed in their study.
According to their research findings, the AdaBoost model achieves the highest accuracy rate of 83.97% among the tested models. This suggests that AdaBoost is particularly effective in predicting loan acceptance based on the bank direct marketing data used in the study.

The application of these machine learning models in predicting loan acceptance highlights their potential in assisting banks and financial institutions in making informed decisions about approving or rejecting loan applications. The high accuracy achieved by the AdaBoost model demonstrates its capability in accurately classifying loan acceptance based on the provided

data-set. It is important to note that the specific details of the data-sets used, the experimental setup, and the performance evaluation metrics are not provided in the given context for either study. However, the reported findings emphasize the effectiveness of Random Forest in predicting loan default in the Chinese P2P market and the superiority of the AdaBoost model in predicting loan acceptance using bank direct marketing data.

In the study (Aphale & Shinde, 2020) the objective was to forecast consumer creditworthiness and develop an automated risk assessment system to aid banks. The study utilized actual bank credit data and employed various machine learning (ML) algorithms to achieve this goal. Different ML algorithms, including neural networks, naive Bayes, K-nearest neighbors (KNN), decision trees, and ensemble learning algorithms, were employed in their analysis. These algorithms were applied to the credit data to train models that can predict the creditworthiness of consumers. The accuracy of the models developed by Aphale and Shinde ranged from 80% to 76%. It is worth noting that the accuracy reported in their study is lower than the accuracy achieved by your models. This suggests that the models employed in your research outperformed the models used in their study in terms of accuracy.

While the specific details of the dataset, experimental setup, and evaluation metrics used in Aphale and Shinde's study are not provided in the given information, the reported accuracy rates indicate that the ML models employed were able to make reasonably accurate predictions regarding consumer creditworthiness. It is important to consider that the accuracy of the models depends on various factors, including the quality and representativeness of the dataset, feature selection, model architecture, hyperparameter tuning, and the specific context of the study. Differences in these factors could potentially explain the variation in accuracy between your models and those in Aphale and Shinde's study.

In general, both studies emphasize the utilization of ML algorithms for credit risk assessment and highlight the potential of these techniques in supporting banks in making informed decisions. However, the specific algorithms, datasets, and experimental setups employed in each study may contribute to the differences in accuracy observed. Further research and exploration of different ML algorithms, as well as the inclusion of additional features and dataset refinements, can contribute to improving the accuracy of credit risk assessment models. Additionally, considering other evaluation metrics such as precision, recall, and F1-score can provide a more comprehensive assessment of the model's performance.

In the study conducted by (Kumar et al., 2021) the authors performed a comprehensive literature review to identify and compare ML-based models for credit risk assessment, specifically focusing on rural borrowers with limited loan history. The objective of their research was to showcase the range of ML algorithms utilized by researchers in this domain and evaluate their effectiveness.

Through their literature review, Kumar et al. identified various ML algorithms that have been employed in credit risk assessment for rural borrowers. These algorithms aim to overcome the challenges associated with limited loan history and enable accurate predictions regarding creditworthiness. The findings of Kumar et al.'s study highlighted the popularity and effectiveness of the ML algorithms utilized in your own study. This suggests that the ML algorithms used in your research align with the current trends and best practices in the field of credit risk assessment.

By leveraging ML algorithms, researchers and practitioners can effectively evaluate the creditworthiness of rural borrowers, even when they have limited loan history. These algorithms provide a means to analyze various factors and patterns that can indicate credit risk, enabling more accurate and informed lending decisions. The effectiveness of the ML algorithms employed in your study, as supported by Kumar et al.'s literature review, further emphasizes the relevance and potential of these techniques in the specific context of credit risk assessment for rural borrowers. It is important to note that the specific ML algorithms, datasets, and evaluation metrics discussed in Kumar et al.'s research are not provided in the given information. However, their literature review serves as a valuable reference to support the validity and applicability of the ML algorithms used in your own study.

In summary, the alignment between your research and Kumar et al.'s findings highlights the consistency and effectiveness of the ML algorithms utilized in credit risk assessment for rural borrowers. Future research can continue to explore and refine these algorithms, taking into account the unique characteristics and challenges of assessing creditworthiness in rural contexts.

In their work (Gao et al., 2021) the authors focused on predicting credit default using a combination of a residual neural network and (GAN). They utilized the lending club public dataset for their analysis. One challenge encountered in the study was the presence of extremely few problematic users, which led to a sample imbalance issue. This imbalance can compromise the effectiveness of the predictive model, as it may not adequately capture the patterns and characteristics of the minority class (i.e., defaulting users).

To address the sample imbalance, Gao et al. employed GAN, specifically a technique called Generic Adverse Networks, to generate additional samples of the minority class. By using GAN, they aimed to achieve a balanced dataset with a 1:1 ratio of good user samples to bad user samples. This approach allows for a more comprehensive representation of the data and helps mitigate the impact of sample imbalance on the model's performance. The authors compared the performance of the residual neural network model with that of logistic regression in predicting credit default. They found that the residual neural network model outperformed logistic regression, with an accuracy improvement of approximately 5%. This indicates that the neural network model, augmented by the GAN-generated samples, yielded better predictive performance for credit default prediction compared to the logistic regression model. The utilization of GAN to address the sample imbalance and generate additional minority class samples demonstrates the effectiveness of this technique in enhancing the predictive capabilities of the neural network model. By improving the representation of both the majority and minority classes in the dataset, the model becomes more adept at capturing the complex relationships and patterns associated with credit default. Overall, the combination of the residual neural network and GAN provides a promising approach for credit default prediction. The study highlights the importance of addressing sample imbalance and leveraging advanced techniques like GAN to improve the performance of predictive models in credit risk assessment. Further research can explore other techniques for addressing sample imbalance and investigate the impact of these methods on the accuracy and robustness of credit default prediction models.

The study titled (Diwate et al., 2023) aims to address the challenge faced by banks in determining the safest individuals to grant loans, considering their limited resources. The researchers propose a solution by leveraging machine learning techniques to analyze past loan records and predict whether assigning a loan to a particular individual would be a safe decision for the bank. The study is divided into four sections: data collection, comparison of AI models on the collected data, training the system on the most promising model, and testing. The process involves cleaning and processing the data, handling missing values, conducting exploratory analysis, building the prediction model, and evaluating its performance on test data.The researchers achieved a best-case accuracy of 0.811 on the initial dataset. Their analysis reveals that applicants with poor credit score ratings are more likely to be denied loan approval due to a higher probability of not repaying the loan amount. On the other hand, applicants with higher income levels and lower loan amounts are more likely to be approved,

as they have a better track record of loan repayment. Other factors such as gender and marital status do not seem to significantly influence the loan approval decision. In summary, the study highlights the importance of using machine learning models to predict loan approval decisions, thereby helping banks optimize their resources and mitigate risks. By analyzing historical loan data, the researchers provide insights into the factors influencing loan approval and demonstrate the potential of AI-based models in making accurate predictions.

The study (TY - CHAP AU et al., 2021) explores the application of Convolutional Neural Networks (CNNs) in the loan approval process, aiming to address the increasing risk of loan default in China's growing loan business. The authors highlight that traditional machine learning methods for classification rely on shallow features and do not fully capture the relationships between these features. In response, they investigate the potential of CNNs, which have been successful in image recognition, speech recognition, and natural language processing.

The researchers examine four different CNN models and conduct experiments to evaluate their performance. The results reveal that the fourth model, trained using the stochastic gradient descent algorithm with momentum, achieves the best performance. The accuracy of this model is reported to be 0.95, indicating a high level of correct loan application predictions. However, the recall (also known as sensitivity) is relatively low at 0.26, suggesting that the model has a relatively high false negative rate, potentially missing some loan applicants who should have been approved.

Overall, the study highlights the potential of CNNs in loan approval prediction by capturing complex relationships among financial features. However, it also indicates the need for further improvement in recall to minimize false negatives, ensuring that deserving loan applicants are not rejected. The findings contribute to the understanding of applying CNNs in financial decision-making and provide insights for financial organizations seeking to enhance their loan approval processes.

The study (Udbhav et al., 2022) focuses on addressing the challenges and time-consuming nature of the home loan eligibility process by leveraging machine learning techniques. The authors highlight the importance of automating the process to streamline the assessment of customer eligibility. They emphasize that existing approaches often neglect outliers and machine learning models, which can lead to decreased accuracy and overall performance. To mitigate this issue, the project incorporates data cleaning techniques to handle null, missing,

and repeated values. By applying bivariate and multivariate analysis, the researchers aim to identify unique relationships and patterns that can enhance model performance.

The authors employ two classification models, Logistic Regression and Gradient Boosting, known for their effectiveness in predicting discrete values and improving overall performance. The project utilizes a dataset collected by a housing finance company to uncover hidden trends and patterns, allowing for the development of a robust machine learning model. To evaluate the model's performance, they employ metrics such as accuracy, precision, and F1 score. The study stands out by utilizing multiple techniques and methods, differentiating it from other models. The authors argue that modern technology not only saves time but also revolutionizes traditional approaches without compromising accuracy. Automation is highlighted as a means to reduce data cleaning efforts and create a precise model that can be used to assess background details and minimize the risk of fraudulent activity.

In summary, this study presents a comprehensive approach to streamline the home loan eligibility process using machine learning techniques. By addressing data quality issues, employing effective classification models, and leveraging modern technology, the authors aim to improve efficiency, accuracy, and convenience in assessing customer eligibility for home loans.

(Reddy & Kavitha, 2010)did research on predicting loan eligibility using a neural network (NN) model and bank datasets. To identify the class label, the researchers used attribute relevance analysis and picked a subset of important attributes, resulting in fewer input nodes in the NN model. This method attempted to increase accuracy by concentrating on the most relevant characteristics. While the study obtained good accuracy with the limited attribute set, the neural network model did not include clustering association rules. Clustering association rules entail applying clustering algorithms to detect patterns and correlations among data items. These rules can be used to improve the prediction model's performance and efficiency. It is worth noting that the decision to exclude clustering association rules from the NN model was most likely motivated by a desire for simplicity and efficiency. While clustering association rules might bring new insights and perhaps improve model performance, they also increase complexity and computational overhead. As a result, the researchers may have chosen to eliminate these in order to streamline the model and provide speedier predictions. Overall, Reddy and Kavitha's paper provided a neural network model for loan eligibility prediction that used attribute relevance analysis to identify important qualities. The omission of clustering association rules was most likely deliberate in order to keep the model simple and efficient.

(Yu & and Zhang, 2005)conducted a study in the financial domain to forecast loan eligibility using a combination of genetic algorithms (GAs), fuzzy logic, and neural networks (NNs). After an iterative evolutionary learning approach had optimized the characteristics and fuzzy NN weights, the researchers attempted to optimize the parameters using a modified gradient descent learning algorithm.

However, in terms of knowledge discovery and financial data mining, the study's strategy fell short. It did not use smart approaches efficiently to uncover important insights or patterns from financial data. Because of this constraint, the study may have missed chances for deeper analysis and the discovery of relevant links within the dataset. While the study's approach to loan eligibility prediction is promising, it is possible that advanced approaches or essential factors required for successful knowledge discovery in the financial realm were ignored. Furthermore, the study's emphasis on parameter optimization via gradient descent and evolutionary learning algorithms may have eclipsed the significance of interpretability and comprehension of the underlying elements impacting loan eligibility.

(Tsai et al., 2009) conducted research with the goal of forecasting loan defaults. Before making loans, they created a model that used empirical assessment methodologies to evaluate the borrower's demographic information and financial prospects. This model's effectiveness was evaluated by comparing it to different strategies. The study attempted to improve loan default prediction by taking into account not only the borrower's financial information but also their decision-making behavior and financial management abilities. The model attempted to provide a more thorough assessment of the borrower's creditworthiness and likelihood of default by including these new characteristics. Tsai et al. compared the performance of their model to different methodologies in order to assess its accuracy in forecasting loan defaults. This evaluation most likely included examining forecast accuracy and measuring the model's capacity to distinguish between borrowers who later defaulted and those who did not.

Table 2.1 Paper Comparison Table

Study	ML Techniques Used	Dataset	Accuracy	Key Findings	Gap
(Infant Cyril & Ananth, 2022)	SSD algorithm, BCO-based deep neuro fuzzy network	Loan data	95% accuracy	SBCO method combining SSD and BCO outperformed other methods with high accuracy.	The accuracy of the devised approach can be improved using hybrid feature selection The data-set used is relatively small
(A & S, 2021)	Logistic Regression, Support Vector Machine	Not specified	RMSE: 81.62% for both models	Both Logistic Regression and SVM models accurately predicted loan eligibility with similar RMSE values.	Additional information on accuracy, recall, and the confusion matrix is not provided
(Kuma et al., 2018)	Neural network model	Lending Club dataset	Accuracy: 93%	Neural network model achieved a high level of accuracy for loan default prediction.	Additional information on other evaluation metrics and and the use of cross-validation techniques not provided
(UCD Michael Smurfit Graduate Business School, Dublin, Ireland et al., 2018)	Artificial Neural Networks, Linear Regression	Kaggle dataset	Accuracy: 97.69609% (ANN), 97.67575% (Linear Regression)	ANN and Linear Regression models provided similar and effective credit default prediction.	Preprocessing steps applied are not explicitly mentioned
(Handzic et al., 2003)	Multi-Layer Perceptron, Ensemble Averaging,	Credit loan applications dataset	Not specified	Committee Machine models outperformed single Multi-Layer Perceptron, and	Experimental setup, and the performanc

	Boosting by Filtering			Boosting by Filtering performed better than Ensemble Averaging.	e evaluation metrics are not provided
(Elette et al., 2010)	Multi-layer feed-forward neural network with back-propagation learning	Jordanian commercial bank loan applications	Not specified	Artificial neural networks showed success in evaluating loan applications in Jordanian commercial banks.	Evaluation metrics are not provided
(Meshref, 2020)	AdaBoost, LogitBoost, Bagging, Random Forest	Bank direct marketing data	Highest accuracy: 83.97% (AdaBoost)	AdaBoost achieved the highest accuracy in predicting loan acceptance using bank direct marketing data.	Additional information on other evaluation metrics not provided
(Gao et al., 2021)	Residual Neural Network, GAN	Lending Club dataset	Improved accuracy by 5% compared to logistic regression	GAN-generated samples enhanced the accuracy of the residual neural network in predicting credit default.	Addresses sample imbalance using GAN to improve performanc e.

CHAPTER THREE

3 METHODOLOGY

3.1. Chapter Overview

This section expounds on the methodology that is used to predict eligible applicant. This section describes the techniques used for data collection from available data source to data preprocessing, and the evaluation metrics used for the models were explained extensively.

Figure 3.1 illustrates the steps required from gathering the data to its implementation. The following block diagram is composed of the different series of steps involved in gathering the data, preprocessing the data, and using a dataset for model training and testing, as well as evaluating the model.

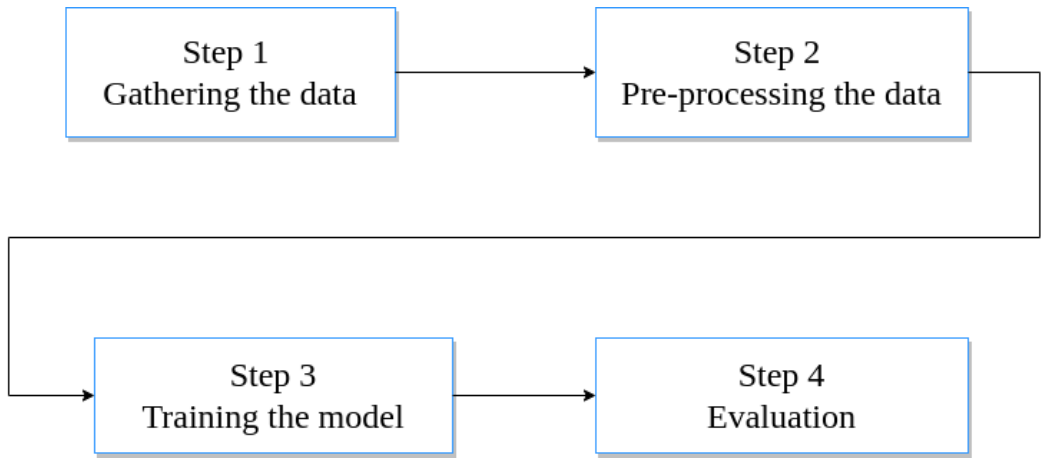


Figure 1: Research Methodology

3.2 Datasets collection

Deep learning models primarily rely on data, without data, the model cannot learn. In studies like loan eligibility prediction, the main task is gathering the necessary data and getting it ready for further processing for loan approval using deep learning. The benchmark dataset for loan eligibility prediction is easily accessible to the general public, particularly for the research community at large. The research deals with pattern prediction, specifically eligibility

prediction, so the data collected are classified into two. The first is the data that is used to train the deep learning model while the second is for testing the performance of the model constructed. For this study, loan data was collected from online lending club data-set. Instead of manual feature extraction, which is difficult for humans to do, deep learning models are utilized, which are life-simplifying and efficient techniques to extract with high accuracy, and performance.

3.3 Evaluation Methods

The evaluation metrics are used to measure the model's quality as well as to determine whether or not the model is operating correctly and appropriately. We can provide the qualitative analysis and quantitative metrics to evaluate the loan eligibility models by using several evaluation methods, such as training accuracy, validation accuracy, training loss, and validation loss.

3.3.1 Training Accuracy

Training accuracy represents the accuracy of the model's predictions on the training data. It is calculated by comparing the predicted loan eligibility values to the actual loan eligibility values in the training dataset. Training accuracy provides an indication of how well the model is fitting the training data. High training accuracy suggests that the model is able to learn patterns and make accurate predictions on the training set.

3.3.2Validation Accuracy

Validation accuracy measures the accuracy of the model's predictions on a separate validation dataset. It is computed similarly to training accuracy but using the validation dataset. Validation accuracy serves as an estimate of how well the model is performing on unseen data. Monitoring validation accuracy during the training process helps detect overfitting. If the training accuracy continues to improve while the validation accuracy starts to plateau or decrease, it indicates that the model is memorizing the training data and not generalizing well to new data.

3.3.3 Training Loss

Training loss, also known as the training error or training cost, quantifies the error between the model's predictions and the actual loan eligibility values in the training dataset. It represents how well the model is fitting the training data. The training loss is typically calculated using a loss function such as binary cross-entropy for binary classification problems.

The goal during training is to minimize the training loss, which means improving the model's ability to make accurate predictions.

3.3.4 Validation Loss

Validation loss measures the error between the model's predictions and the actual loan eligibility values in the validation dataset. It provides an estimate of the model's performance on unseen data. Similar to validation accuracy, monitoring the validation loss helps detect overfitting. If the training loss continues to decrease while the validation loss starts to increase or plateau, it indicates that the model is overfitting the training data and may not generalize well to new data.

3.4 Development Tools

For this research, numerous types of development tools will be used to design and to implement the proposed thesis work. These tools include prototype development tools and platforms, UML Modeling tools, and other relevant tools that are required to do the research. The following sectionswill give a brief detail about these development tools.

3.3.1 Design Tools

Design tools are mediums that will be used for the creation, presentation, and interpretation of design concepts. Enterprise Architect will be used to design the proposed system. It is a lightweight and powerful graphic design tool for creating professional-looking flowcharts, network diagrams,flowchart diagrams, and others.

3.3.2 Hardware tools

The following hardware tools are going to be used in the research implementation

Table 2: Hardware Tools

No.	Tools	Specification	Used for
1	GPU	EVGA GeForce GTX 1050 Ti	To increase the computation and to expedite the training.
2	Hard Drive (HDD)	WD 1TB USB 3.2 Gen 1	Used as storage for large data sets.
3	RAM	DDR4-240(PC4-19200)= 2400MHz/19200Mbps	To accelerate the training process cooperatively with GPU

3.3.3 Software tools

For the implementation part of the research, different IDEs (Integrated Development Environments), code editors and other coding tools will be used and these are listed below with their description.

Table 3: Software Tools

No	Software tools	Version	Description
1	Python	3.8.5	Python is high level and general-purpose programming language that is abundant of libraries and frameworks that facilitate coding and save development time.
2	Jupyter Notebook	6.5.2	Jupyter Notebook is the most popular and handy IDE among AI and deep learning researchers to work with python.
3	TensorFlow	2.11.0	TensorFlow is an open-source software library optimized for maximum performance numerical modeling and processing. Its modular architecture can be easy.
4	Keras	2.4.0	Keras is a user-friendly deep learning API written in python running on the top of Tensorflow.

CHAPTER FOUR

4. PROPOSED SOLUTION

4.1. Chapter Overview

The proposed solution architecture for the Loan Eligibility Prediction problem is presented in this chapter, together with mathematical equations and the necessary diagrammatic representation. This chapter is divided into three sections: the first section covers the dataset, the second describes several pre-processing techniques, and the third portion introduces the proposed model architecture for Predicting eligible loan applicant

4.2 Dataset

The dataset in this research was collected from kaggle dataset(online). The dataset consists of 44 columns and 461,097 entries.

4.3 Preprocessing

The second phase of the proposed model is the preprocessing phase that occurs after the is loaded. The goal of preprocessing is to transform the raw data into a suitable format that can be effectively utilized by the deep learning model.

In this research, we use some pre-processing techniques. Such as Data cleaning (Handling Inconsistencies), Data Scaling and Encoding categorical variables into numeric values for the proposed model, are the techniques we apply to the data-set. The these techniques are essential and best practices of preprocessing stages that are needed to be done before sending the data into deep learning models. All of them plays important role to produce suitable data for the model training.

4.3.1 Data Cleaning

It is essential for improving the quality and reliability of data-sets by addressing errors, inconsistencies, and missing values. It ensures that data is consistent, accurate, and complete, enabling more reliable analysis and machine learning models. By removing duplicates, handling missing data, and standardizing variables, data cleaning enhances data interpretability and minimizes biases, ultimately leading to more robust and trustworthy insights and decisions.

4.3.1.1 Handling Inconsistencies:

Handling inconsistencies in data is a crucial step in data preprocessing and cleaning. Inconsistencies can arise due to various reasons such as data entry errors, different data sources, data integration issues, or system limitations. Dealing with inconsistencies is important to ensure the quality, reliability, and accuracy of the data for subsequent analysis or modeling tasks.

To handle inconsistencies, it is necessary to first identify and understand the nature and patterns of the inconsistencies. This involves analyzing the data, conducting exploratory data analysis, and using visualization techniques to gain insights into the inconsistencies present. Once the inconsistencies are identified, appropriate strategies and techniques can be applied. These strategies may include data transformation, where inconsistencies are resolved by converting data into a standardized format or applying specific rules to correct errors. Data imputation techniques can also be employed to fill in missing values or replace erroneous values with plausible estimates based on statistical methods or domain knowledge.

```
# Calculate the desired statistic (mean)
statistic = df[column].mean()

# Replace inconsistent values with the calculated statistic
df[column] = df[column].fillna(statistic)
```

Figure 2: Inconsistencies

4.3.3 Categorical Variables Encoding

The next step is to encode categorical variables into a numerical representation that can be understood by the deep learning model. Categorical variables, such as loan purpose, employment type, or education level, are typically encoded using techniques like one-hot encoding or label encoding. One-hot encoding creates binary columns for each category, where a value of 1 indicates the presence of the category, while label encoding assigns a unique numerical value to each category.

One-hot encoding is a technique for converting categorical data into a numerical representation that can be used in deep learning algorithms. The process involves creating a binary feature for each unique category in the original variable. For each observation, the value of the corresponding feature is set to 1 if the observation belongs to that category, and 0 otherwise.

This ensures that no ordinal relationship is assumed between the categories, and each category is treated as an independent and distinct feature.

```
from sklearn.preprocessing import OneHotEncoder
# define one hot encoding
encoder = OneHotEncoder(sparse=False)
```

Figure 3: Define one-hot encoding

4.3.2 Data Normalization or Scaling

After handling inconsistencies, the numerical features in the data-set need to be normalized or scaled. This step ensures that all numerical features are on a similar scale, preventing features with larger magnitudes from dominating the training process. Common normalization techniques include min-max scaling, which scales the values to a specific range, or z-score normalization, which transforms the values to have a mean of 0 and standard deviation of 1.

In this research Min-Max scaling technique will be used for scaling purpose. It is a method for preprocessing data that reduces the numerical features of a data-set to a specific range, usually between 0 and 1. Each feature's values are rescaled such that the smallest value is made to equal 0, the maximum value is made to equal 1, and the values in between are proportionally transformed. This technique is beneficial when the range of values in different features varies widely, and it aims to bring all features to a common scale. By ensuring that all features are scaled uniformly, preventing the dominance of specific features, and promoting convergence, this preprocessing method helps CRNNs. By enabling stable weight updates, it aligns with activation functions, prevents saturation or vanishing gradient problems, and makes optimization during training easier. Min-max scaling offers homogeneity across features, which helps regularization procedures.

```
scaler = MinMaxScaler()
```

Figure 4 : Define Min-max scaler

4.3.3 Handling Data-Imbalance:

Handling data imbalance is an important aspect of working with imbalanced data-sets in deep learning. When the classes in a data-set are not represented equally, the model tends to be biased towards the majority class, leading to suboptimal performance on the minority class.

As I checked my data-set if any data-imbalance, the result shows there exist. So in this research I decided to go with Oversampling technique, which duplicates or synthesizes new examples from the minority class to increase its representation. And one of the most popular oversampling techniques, SMOTE (Synthetic Minority Over-sampling Technique) is used for this case. SMOTE, aims to increase the representation of the minority class by generating synthetic examples rather than simply duplicating existing ones. This approach helps to overcome the limitations of simply duplicating examples, which can lead to overfitting and poor generalization.

In order to connect instances of the minority class, SMOTE builds synthetic examples along the line segments. It chooses a sample from a minority class at random and locates its k closest neighbors in the feature space. SMOTE picks one of these neighbors, computes the difference between the feature vectors, and multiplies it by a chance number between 0 and 1. A new synthetic example is created by adding this variation to the chosen minority example. Repeating this procedure results in the desired level of imbalance.

```
# Apply SMOTE for oversampling on the training set
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_stratified, y_train_stratified)
```

Figure 5: Handling Imbalanced Data

Although there are other preprocessing techniques, in this research I have gone for only these steps since we are using deep learning. I leave other steps for deep learning model to do things like feature selection.

4.4. Proposed Model Architecture

Convolutional Recurrent Neural Network (**CRNN**) is the proposed model to address the Loan Eligibility Prediction. To predict eligible applicant, a CRNN-based architecture is used. The architecture consists of several layers such as a convolutional layer, max-pooling layer, Bidirectional GRU (Gated Recurrent Unit) layer, dropout layer, flatten layer, fully-connected layers to achieve high prediction accuracy. In the convolutional layer I use **Conv1D** which is essential for feature extraction from the input (sequence data). Conv1D (1-dimensional convolutional) layers are frequently employed in sequence data processing activities including time series analysis and natural language processing.

They are very good at identifying local patterns and extracting relevant features from sequential data. By applying multiple Conv1D layers, the model can learn increasingly complex and abstract representations of the input sequence. The output of these convolutional layers is then fed into subsequent layers, such as the MaxPooling1D layer and the Bidirectional GRU layer, which further process and learn from the extracted features. In the max-pooling layer I use **MaxPooling1D** layer, which is recommended for one-dimensional data like my input sequence data. MaxPooling1D is used to reduce the spatial dimensions of the data while retaining the most important(relevant) features. This dimensionality reduction can help improve computational efficiency and reduce the risk of overfitting. The next layer is **Dropout layer,** it is a good practice to apply dropout technique to prevent overfitting and improve the generalization capability of neural network models. The other layer is another important layer that makes CRNN happen, which is one the most popular RNN architecture, **GRU (Gated Recurrent Unit)** that addresses some of the limitations of traditional RNNs, such as the vanishing gradient problem**.** GRU units are designed to capture long-term dependencies in sequential data while mitigating the vanishing gradient problem. They are computationally efficient and have been widely used in various natural language processing (NLP) tasks, time series analysis, and other sequential data modeling tasks. In this layer I made the layer to be **Bidirectional:**

- To achieve the benefit of **Capturing Context in Both Directions:** A bidirectional GRU processes the input sequence in both forward and backward directions

  By doing so, it captures information from both past and future time steps. This enables the model to have a comprehensive understanding of the context and dependencies within the sequence.
- To **Improve Performance on Prediction Tasks**: Bidirectional GRUs can be particularly useful for tasks where the context from both past and future time steps is crucial for accurate predictions.

The other **Dropout layer** is applied**,** which is again a good practice to prevent overfitting and improve the generalization capability of neural network models. And the last one is **Final layer,** is fully-connected layers the final fully connected layer with a single unit. It applies a S**igmoid Activation Function**, which squashes the output between 0 and 1, making it suitable for binary classification tasks like my research to classify the given applicant is Eligible (Good Sign) or Not Eligible (Bad Sign). Furthermore, the training was performed by applying the back-propagation approach of Adaptive Moment Estimation (Adam optimizer) which will be discussed later.

As I mentioned above the proposed model architecture consists several layers. In model there are 3 convolutional layers, 31max-pooling layers, 1 Bidirectional (GRU) layer, 2 dropout layers, 1 fully-connected layer which is an output layer.

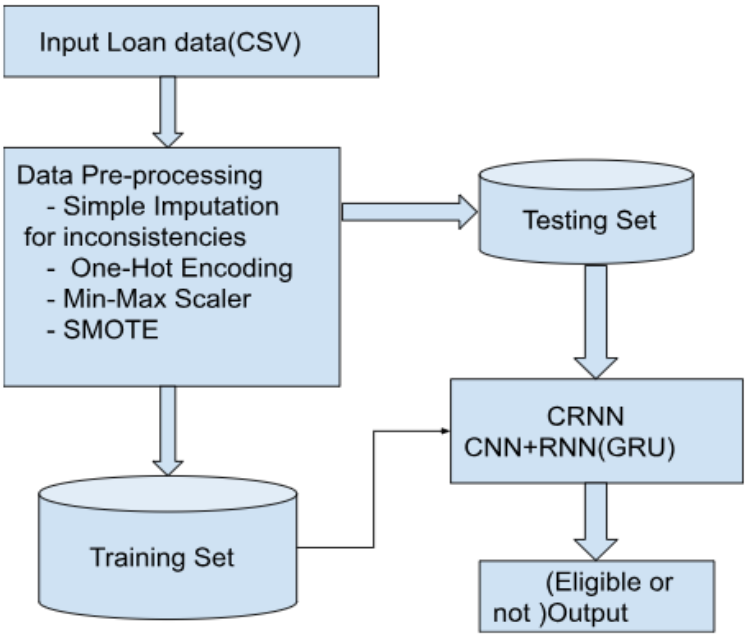


Figure 6 : Methodology

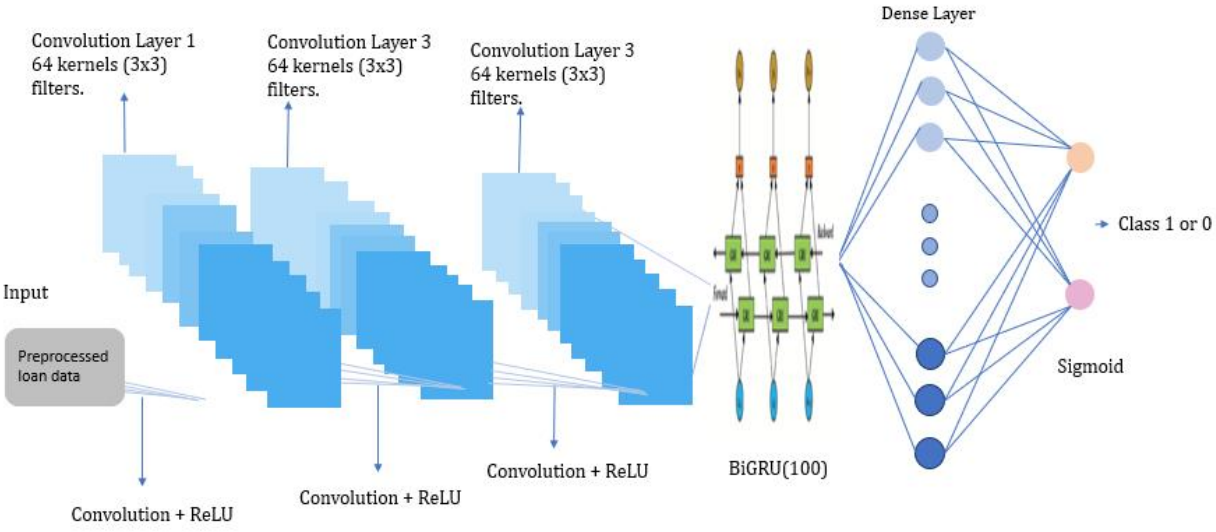


Figure 7: Proposed model

4.4.1 Feature Extraction

Always the main things that will be done while model training is feature extraction. The CRNN algorithm, feature extraction from input data is achieved through a combination of convolutional and recurrent layers. The input data is first fed into convolutional layers, which apply filters to capture local patterns and spatial dependencies. The output of the convolutional layers is then passed through recurrent layers, typically using GRU (Gated Recurrent Unit) units, to capture temporal dependencies and contextual information. The recurrent layers enable the model to learn and represent sequential patterns in the data. The final output of the CRNN model is a set of extracted features that encode both local and global information from the input sequence, making it suitable for tasks such as text classification, or time series analysis.

4.4.2 Backpropagation

The back-propagation method is used to determine how much weight is responsible for the network's overall mistake. The following equations serve as examples of the network's back-propagation.

$$\boldsymbol{\delta^L} = \frac{\partial C}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial z^{(L)}} = \frac{1}{n}\left(a^{(L)} - 1\right) f'\left(z^{(L)}\right)$$ Equation 1 *(4.1)*

$$\boldsymbol{\delta^l} = \frac{\partial C}{\partial Z^{(l)}} = \frac{\partial C}{\partial z^{(l+1)}}\frac{\partial z^{(l+1)}}{\partial z^{(l)}} = \frac{\partial z^{(l+1)}}{\partial z^{(l)}}\delta^{(l+1)} = w^{(l+1)}\delta^{(l+1)} f'\left(z^l\right)$$ Equation 2 (4.2)

$$\frac{\partial C}{\partial b^{(l)}} = \delta^l$$ Equation 3 (4.3)

$$\frac{\partial C}{\partial w^{(L)}} = a^{(l-1)}\delta^l$$ Equation 4 (4.4)

4.4.3 Activation function

In the CRNN model, two alternative activation functions are used in the experiments: **ReLU** and **Sigmoid** are the most common activation functions for CNN and RNN models. The sigmoid activation functions, in particular, in the model's output layer. ReLU is preferable for convolution and fully connected layers.

ReLU is a straightforward non-linear function that maintains the positive values while setting all negative values to zero. Its main purpose is to add non-linearity into the network, enabling it to learn complicated patterns and speeding up the learning process. The smooth S-shaped sigmoid function, on the other hand, compresses input values between 0 and 1. It is frequently

employed to generate binary predictions. Given that it can transfer any real-valued number to a range from 0 to 1, the sigmoid function is appropriate for tasks like binary classification and probability representation. In this research ReLU used for capturing non-linear relationships and learning complicated features and sigmoid for providing binary classifications.

4.4.5 Epochs

An epoch refers to a complete pass or iteration through the entire training dataset during the training of a machine learning model. In the context of neural networks, an epoch involves presenting each training sample to the model, computing the loss or error, and adjusting the model's parameters (weights and biases) through backpropagation and optimization algorithms like gradient descent. The number of epochs is typically specified beforehand and determines how many times the model will cycle through the entire training dataset.

Each epoch allows the model to learn and update its parameters based on the accumulated information from the training samples, gradually improving its performance and convergence towards an optimal solution. 10 epochs were used to train this model. During each epoch, the model is exposed to the entire training dataset multiple times, allowing it to learn and refine its parameters based on the accumulated information. With each epoch, the model has the opportunity to adjust its weights and biases, capture more intricate patterns, and improve its ability to recognize and understand the sequential nature of the input data.

4.4.6 Batch size

It is the total number of training inputs that have been given to train the model. The 200-batch size was used to train this model. A larger batch size can offer several advantages. First, it can improve computational efficiency by allowing for parallel processing, taking advantage of hardware acceleration like GPUs. This can speed up the training process, especially for large data-sets. Second, larger batch sizes can provide a more stable gradient estimation since they average the gradients computed over multiple examples, reducing the effect of individual noisy or outlier samples. This can lead to smoother convergence and more consistent updates to the model's parameters. But using larger batch size requires more memory.

4.4.7 Learning rate

The learning rate is incorporated into weight updates for the CRNN model, which was trained with back-propagation. In back-propagation, it chooses how much weight should be updated. The hardest part of the experiment was determining the appropriate learning rate. It takes more time to train learning rates with a lower value than those with a higher value. When given a lesser value, however, the model performs better than one with a higher learning rate. The learning rates used in the experiment were 0.001, 0.0001, and 0.000001. This was done to determine the best learning rate.

4.4.8 Hyper-parameters for CRNN Model

To make efficient computing with available resources, hyper-parameter optimization is used. For the proposed CNN model, the following hyper-parameters were chosen:

- Architecture for deep learning:
  - CRNN
- Training mechanism:
  - Starting at the Beginning
- Distribution of training, validation, and testing sets:
  - Train: 80%, Test: 10%, Validation: 10%
- Data set type:
  - One-hot Encoded Vectors
- Batch size:
  - 200
- Number of Epoch:
  - 10

CHAPTER FIVE

5. IMPLEMENTATION OF THE PROPOSED SOLUTION

5.1. Chapter Overview

This chapter deals with the implementation process. From the initial setup of the working environment to the data processing and training procedures, everything is covered. In addition, the implementation's code snippets are shown and discussed in depth.

5.2. Working Environments

This section describes the environment setup of the hardware components that are used in the implementation of the modeling process with their specifications.

- Laptop:
  - Operating System: Linux Mint 20.2 Cinnamon
  - Processor: Intel© Core™ i7-7500U CPU @ 2.70GHz × 2
  - Installed memory (RAM): 8.00 GB
  - Graphics Card: Intel Corporation HD Graphics 620
  - System Type: 64-bit operating system, x64-based processor

5.3. Environmental Setup

Specific online programming tools were used in this study.

- **Colab**: is a Google cloud-based notebook that allows you to create Python code on any browser with sufficient computing power (2 vCPU, 12GB RAM) and resources to train with the TPU and GPU.

- **Keras**: the main goal is to make it as simple as possible to train deep learning models for research and development with models that are suited for deep learning and allow for quick calculations and prototyping. Because the software library uses a GPU in addition to the computer's CPU. The module is currently found in the Tensor flow.

➢ **TensorFlow**:

It is a powerful and widely-used open-source framework for deep learning research. It provides a comprehensive set of tools and functionalities that enable researchers to develop, train, and deploy deep neural networks efficiently. TensorFlow offers a flexible and intuitive programming interface that allows researchers to define complex network architectures, experiment with different layers, and implement various optimization algorithms. Its computational graph abstraction allows for efficient execution of computations on both CPUs and GPUs, enabling researchers to leverage the computational power of modern hardware.

Additionally, TensorFlow offers a vast collection of pre-built neural network layers, activation functions, loss functions, and optimizers, which significantly speeds up the development process. Furthermore, TensorFlow integrates seamlessly with popular machine learning libraries and frameworks, facilitating interoperability and enabling researchers to leverage a wide range of data manipulation and visualization tools.

5.4. Training Procedure

In the training procedure the following steps has taken place in order to reach the goal of this research.

- Data Loading
- Data preprocessing
- Configuring the GPU and the dataset
- Implementing the proposed solution.

In order to provide more information on what has been done, the all techniques are described in detail at each phase.

5.5 Load Data

Data can be loaded from Google Drive by first uploading the necessary files from local storage to Google Drive, which is then connected to Colab and mounted like the below code snippets shows:

```
df = pd.read_csv('/content/drive/MyDrive/Credit_informations.csv',sep=';')
```

Figure 8 : Data load

5.6. Implementation Techniques for Data Preprocessing

5.6.1. Data Cleansing

Once the data is loaded, the next step is, to first check the if any missing, duplicate values and inconsistencies and handle them**.** It's crucial to locate any missing values, duplication and inconsistencies in the dataset and use the proper methods to deal with them. Based on the output there is only inconsistencies in my data-set. So, the next step is to handle these inconsistencies.

The following two code snippets are for missing values and duplication check and their output.

```
# Check for missing data
missing_values = df.isnull().sum()
if missing_values.sum() > 0:
    print("Missing values exist in the dataset.")
    print(missing_values)
else:
    print("No missing values found in the dataset.")
```

```
No missing values found in the dataset.
```

Figure 9: Check for missing data

```
# Check for duplicates
duplicates = df.duplicated()
if duplicates.any():
    print("Duplicates exist in the dataset.")
else:
    print("No duplicates found in the dataset.")
```

```
No duplicates found in the dataset.
```

Figure 10: Check for duplicates

5.6.1.1 Handling Inconsistencies

As I have discussed above the first thing is to know the existence of inconsistencies and the then understand their nature by analyzing the data. Then apply technique to handle them.

The code snippet below is to check for inconsistencies.

```
# Check for inconsistencies in the entire datase
inconsistencies = df.apply(lambda x: x.nunique()
if inconsistencies.any():
    print("Inconsistencies exist in the dataset.
    print(inconsistencies)
else:
    print("No inconsistencies found in the datas
```

```
Inconsistencies exist in the dataset.
```

Figure 11: Check for Inconsistencies

As we can understand from the output, there are inconsistencies in the dataset. So, we need a technique like Simple Imputations (mean strategies) to handle them. And I applied this technique as the following code snippets shows:

```
if np.issubdtype(df[column].dtype, np.number):
    # Calculate the desired statistic (mean)
    statistic = df[column].mean()

    # Replace inconsistent values with the calculated statistic
    df[column] = df[column].fillna(statistic)
```

Figure 12: Handling Inconsistencies

And this was the output after applying the above technique and it was a success:

```
No inconsistencies detected.
```

Figure 13: Output of handled inconsistency

5.7. Implementation Techniques for CRNN

5.7.1. Data Encoding: (One-Hot Encoding)

As the existence of categorical data in some features, encoding needs to takes place in order to convert them into numerical form so that they can be understood by the model during training. The below code snippets describe how I apply **One-Hot Encoding,** which is one of the most popular encoding techniques for converting categorical data into a numerical representation that can be used in deep learning algorithms.

```
from sklearn.preprocessing import OneHotEncoder
# define one hot encoding
encoder = OneHotEncoder()

# transform data
onehot = encoder.fit_transform(X)
```

Figure 14: Applying One-hot encoding

The process involves creating a binary feature for each unique category in the original variable. For each observation, the value of the corresponding feature is set to 1 if the observation belongs to that category, and 0 otherwise.

5.7.2. Data Normalization or Scaling (Min-Max Scaler)

After the categorical encoded, all the data features become numeric format which is suitable for scaling or normalization process. In this research I used Min-max scaling technique to normalize my data in order to enhances improving data quality.

Min-Max scaling, sometimes referred to as normalization, is mostly used to rescale numerical features to a particular range, usually between 0 and 1. The following code snippets shows the code I implemented to use Min-Max scaler:

```
# Use the min-max scaler from To scale the data

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# transform data
X= scaler.fit_transform(df)
```

Figure 15: Applying Min-max scaling

5.7.3. Handling Data Imbalance

```
from sklearn.model_selection import StratifiedShuffleSplit

X = df.drop(['good_bad'], axis=1)
X = np.array(X).reshape(-1, len(X.columns), 1, 1)
y = np.array(df['good_bad'])

# Initialize StratifiedShuffleSplit
stratified_split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)

# Perform stratified splitting
for train_index, test_index in stratified_split.split(X, y):
    X_train_stratified = X[train_index]
    X_test_stratified = X[test_index]
    y_train_stratified = y[train_index]
    y_test_stratified = y[test_index]
```

Figure 16: Data Split

As we discussed earlier to there is data-imbalance in the data-set, so this case is handled after data-split (StratifiedShuffleSplit) takes place like below:

So, after split applied on the data set, the Oversampling (SMOTE) will be applied on the training data that is X-stratified and y-stratified. The below code snippets show how the presence of data imbalance check all over the data-set

```
#check for the imbalaced data
class_counts = df.value_counts()
print(class_counts)
```

Figure 17: Check for imbalanced data in all data-set

And this is one is for only target class:

```
#check for the imbalaced data
class_counts = df['good_bad'].value_counts()
print(class_counts)
```

Figure 18: Check for imbalanced data in target class

```
1    410862
0     50235
Name: good_bad, dtype: int64
```

Figure 19: Output of target class before SMOTE

```
from imblearn.over_sampling import SMOTE

# Reshape the training data if it has additional dimensions
X_train_stratified = X_train_stratified.reshape(X_train_stratified.shape[0], -1)

# Apply SMOTE for oversampling on the training set
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_stratified, y_train_stratified)
```

Figure 20: Applying SMOTE

The above code snippets output for target class says there is data imbalance across classes, then after this I apply SMOTE like below:

Then we check with the below code snippets if the SMOTE works as needed or not:

```
# Check the class distribution after oversampling
class_counts = pd.Series(y_train_resampled).value_counts()
print(class_counts)
```

Figure 21: Checking imbalance after SMOTE

And this was the output and SMOTE did re-sampling perfectly:

```
1    328689
0    328689
dtype: int64
```

Figure 22: Output after SMOTE

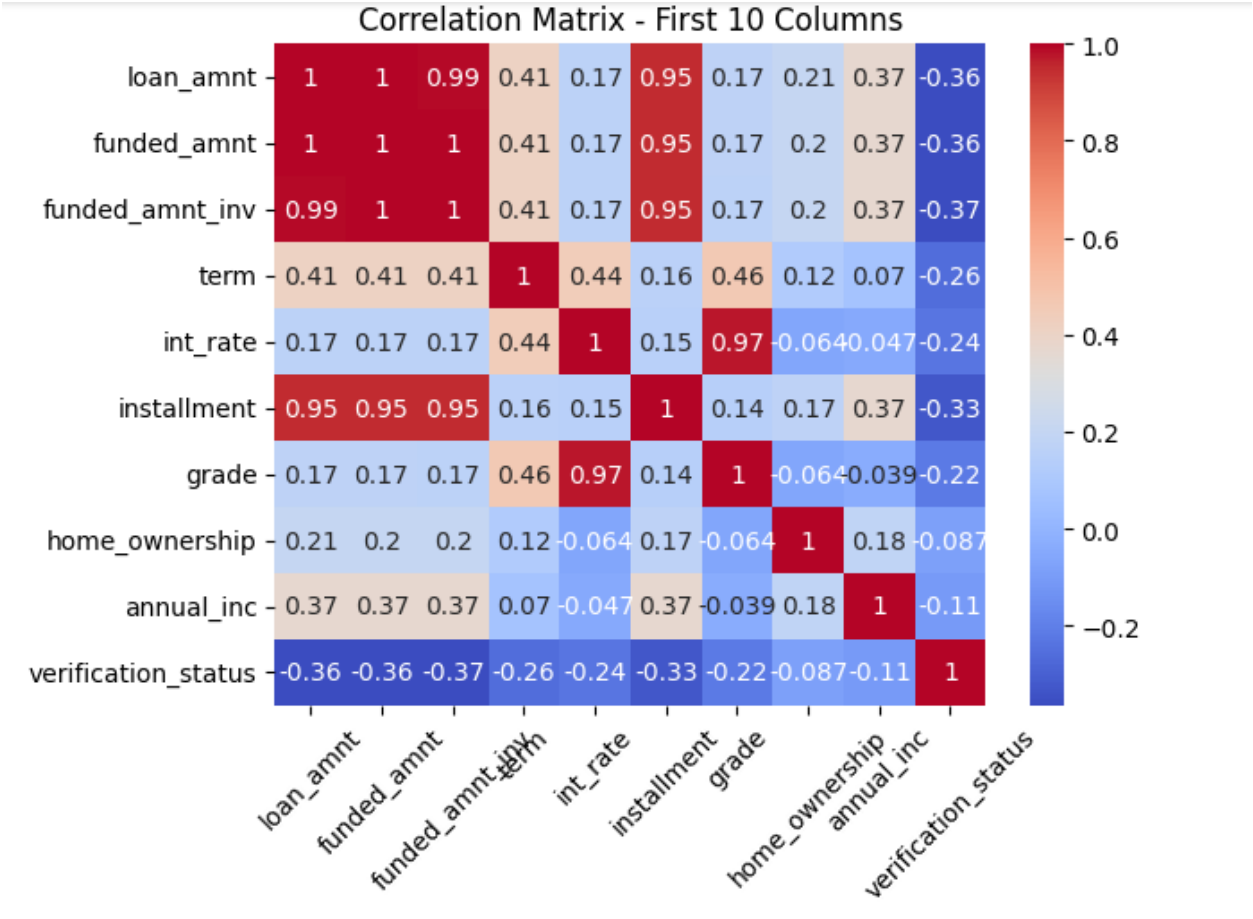
5.7.4 Correlation

A statistical measure that quantifies the link between two variables is correlation. It examines how changes in one variable are related to changes in another. Correlation enables us to comprehend the degree and direction of links between variables. The degree of association is indicated by the correlation coefficient, which ranges from -1 to 1. A value of -1 shows a perfect negative correlation, a value of 0 indicates no correlation, and a value of 1 indicates a perfect positive correlation. The correlation was applied to the whole data set but in the correlation matrix only of the first 10 columns were plotted since plotting the whole columns (44 columns) results a sophisticated correlation matrix which difficult to understand.

From the correlation the highest and lowest relevant columns were observed as the below code snippets shows:

```
Highest Relevant Column: loan_amnt
Lowest Relevant Column: issue_d
```

The following figure shows the correlation matrix plotted for only 10 columns:



5.7.4. The Hyper-Parameter Tuning

The process of choosing the best hyper-parameters for a deep learning model is known as hyper-parameter tuning. Hyper-parameters are parameters that are set before training instead of being directly learned from the data during the training process. They significantly affect the model's performance and regulate the learning algorithm's behavior.

Learning rate, batch size, number of hidden layers, number of neurons in each layer, regularization parameters, and other factors are hyper-parameters. The model's capacity to learn and generalize from the training data can be significantly impacted by the values given to these hyper-parameters.

Searching through a preset space of hyper-parameter values and choosing the combination that yields the optimum performance for the specified problem is known as hyper-parameter tuning. The objective is to identify a set of hyper-parameters that maximizes the selected performance metric, such as accuracy or F1 score, or minimizes the model's error on unobserved data.

5.7.5. Optimizer

An optimizer is a technique or algorithm used to adjust a neural network's parameters as it is being trained. On the basis of the determined gradients of the loss function, it decides how the model's parameters should be changed.

```
optimizer='adam',
```

Figure 23: Optimizer

The optimizer's goal is to reduce the loss function and direct the model toward improved performance. It is essential to the neural network's convergence and training effectiveness. And the following code snippets shows how **Adam (**Adaptive Moment Estimation) was used for optimization.

5.7.6 Loss Function

A loss function, sometimes called an objective function or cost function, is a mathematical measure that expresses the difference or error between a Deep learning model's anticipated outputs and the actual or desired outputs. It acts as a reference for the model to modify its settings throughout training to reduce this inaccuracy.

In this research work **Binary-Crossentropy** is used to specify the loss function to be applied when a binary classification model is trained. When attempting to categorize inputs into one of two groups, just like this researches case which to classify the applicant to eligible or not, the binary cross-entropy loss function works effectively.

This was how it is used as the below code snippets shows:

```
loss='binary_crossentropy',
```

Figure 24: loss function

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

Figure 25: Model compile

5.7.7 Learning Rate Adjustment

Although **Adam** takes cares the learning rate adjustments automatically, I tried to explicitly adjust the learning rate for experiment purpose or to see what could happen on model performance and how can this technique enhance to increase the performance of the model.

It can be advantageous to explicitly change the learning rate in addition to utilizing the Adam optimizer. Adam adjusts the learning rate automatically based on the estimated moments of the gradients, although explicit modification can be helpful for a variety of purposes, including stabilizing the training process, enhancing or transferring learning, overcoming flat or narrow optima, and tweaking hyper-parameters. Lowering the learning rate can aid in stabilizing optimization, maintaining previously learned skills while adjusting to new tasks, exploring difficult regions, and discovering more effective models. To avoid problems like sluggish convergence or becoming stuck in less-than-ideal solutions, it is essential to establish a balance and carefully regulate the learning rate.

A hyper-parameter called the learning rate regulates how much the weights in our network change in response to the loss gradient. A value that is too little could cause the training process to halt and take longer, while a value that is too large could cause the training process to become unstable or learn a subpar set of weights too quickly.

5.7.8. Define Model

First is first and it is to establish a fundamental model for the Convolutional Recurrent Neural Network to solve the issue. The feature extraction front end of the model, which consists of convolutional layers, pooling layers, bidirectional GRU layer, and the classifier backend, which produces a prediction, are its two primary parts. A convolutional layer and a pooling layer are the two wheels of a CNN model and GRU is for RNN.

The combination of RNN and CNN layers are implemented in this model to reach the best out of them. Each Conv1D and MaxPooling1D layer generates a one-dimensional tensor of the form, in which I am using 1D (one dimensional) since I have sequential data input. The first option determines how many output channels each Conv2D layer has (e.g., 32 or 64 or 128...)

```
model = tf.keras.Sequential()

model.add(Conv1D(64, kernel_size=3, activation='relu', input_shape=(43,1)))
model.add(Conv1D(64, kernel_size=3, activation='relu'))
model.add(Conv1D(64, kernel_size=3, activation='relu'))
model.add(MaxPooling1D(2))

model.add(Dropout(0.25))

# GRU layers
model.add(Bidirectional(GRU(100, return_sequences=False)))

model.add(Dropout(0.25))

#Final layers
model.add(Dense(1, activation='sigmoid'))
```

The model had its first three 1D convolutional layers. Each Conv1D layer employs the ReLU activation function, contains 64 filters, and a kernel size of 3 (each filter will cover a window of three consecutive elements in the input sequence). The input shape for the first convolutional layer is also specified as (43, 1), indicating that the input sequences have a length of 43 and a single feature.

This, the next line adds a MaxPooling1D layer to the model. Max pooling reduces the spatial dimensions of the input by taking the maximum value within a specified window. In this case, the MaxPooling1D layer has a pool size of 2, meaning it reduces the sequence length by half.
Then Dropout is a regularization technique used to prevent overfitting. It randomly sets a fraction of input units to 0 at each update during training, which helps to reduce co-adaptation between neurons. In this code, a dropout rate of 0.25(which 25% of the data) is applied after the convolutional layers and before the GRU layer.
The model is then given a bidirectional GRU layer in the following line. Recurrent neural networks (RNNs) of the GRU variety may recognize long-term dependencies in sequential input. The GRU layer may gather data from both the past and the future by processing the input

sequence in both forward and backward directions thanks to the Bidirectional wrapper. With 100 units and a return sequence=False flag, the GRU layer only outputs at the last time step. After the bidirectional GRU layer, a dropout layer with a rate of 0.25(25% of the data) is added to further regularize the model. The dense layer with one neuron and a sigmoid activation function makes up the model's last layer. A single output value, ranging from 0 to 1, which represents the predicted class probability, is generated by this layer. I applied this activation function as it issues frequently involve the binary classification problems like loan eligibility prediction (Eligible or Not).

5.8. Learning hyper-parameters

For deep learning models like CRNN to understand the relationship between features and outputs, hyper-parameters must be carefully chosen. Hyper-parameters are variables that have an impact on how well other model parameters and network training technique's function. The following hyper-parameters were therefore used in this study to achieve a better outcome, and Adam were used to represent part of them for model tuning. Adam is a type of stochastic gradient descent optimization method that modifies iterations of the network weights using training data. It is well-liked because it is simple to use, includes Keras, requires less memory than other optimization techniques, is computationally effective, and works well in scenarios where there are lots of variables (data).

The second hyper-parameter, learning rate (LR), governs how quickly a model learns new information or how quickly a change in weight is calculated. If the learning rate is low, the model may never achieve the global minimum of the loss function, whereas a larger learning rate could cause the model to oscillate around this value. It is demonstrated in Section 5.7.7 that if there is no iteration, the learning rate won't be fixed after a certain number of epochs.

But it's important to choose the initial learning rate value wisely.

Table 4: Learning hyper-parameters

No.	**Hyper-parameters**	**Value**
1	Learning rate	0.001
2	Optimizer	Adam
3	Epoch	10
4	Batch size	200

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 41, 64)            256

 conv1d_1 (Conv1D)           (None, 39, 64)            12352

 conv1d_2 (Conv1D)           (None, 37, 64)            12352

 max_pooling1d (MaxPooling1D  (None, 18, 64)           0
 )

 dropout (Dropout)           (None, 18, 64)            0

 bidirectional (Bidirectiona  (None, 200)              99600
 l)

 dropout_1 (Dropout)         (None, 200)               0

 dense (Dense)               (None, 1)                 201

=================================================================
Total params: 124,761
Trainable params: 124,761
Non-trainable params: 0
_________________________________________________________________
```

Figure 26: Model summary

CHAPTER SIX
6. RESULTS AND DISCUSSIONS

6.1 Chapter Overview

This chapter covers the results obtained from the proposed solution and provides a comparative description of the results with graphs and tables, having previously discussed the concepts for implementation and the details of this work.

6.2 Result Discussion

The CRNN is used to track and compare the variances in accuracy between various predictions from loan eligibility prediction models. TensorFlow and Keras, two of the most well-liked Python libraries for machine learning and deep learning, are used to calculate the accuracy rates. By using various combinations of convolution layers for hidden layers and employing batch sizes of 200 in each case, training and validation accuracy were evaluated for 10 distinct epochs.

The accuracy of the CRNN is shown in Figures 6.1, 6.3, and 6.5, while the loss of the CRNN with different convolution and hidden layer combinations is shown in Figures 6.2, 6.4, and 6.6. Following trials for four different cases with various Batch Size, hidden layers and Epochs, Table 6.1 displays the maximum and lowest training and validation accuracies of the CRNN, and Table 6.2 displays validation loss of the CRNN in various scenarios for the prediction of loan-eligible candidates.

Table 5: Performance of 4 models with different cases

Case	Number of Hidden Layers	Batch Size	Minimum Training Accuracy		Minimum Validation Accuracy		Maximum Training Accuracy		Maximum Validation Accuracy		Overall Performance Test Accuracy (%)
			Epoch	Accuracy (%)	Epoch	Accuracy (%)	Epoch	Accuracy (%)	Epoch	Accuracy (%)	
1	3	50	1	88.39	1	89.18	9	97.96	10	98.06	98.25
2	4	50	1	90.29	1	95.90	10	97.94	12	98.24	98.23
3	5	80	1	94.18	1	95.87	25	98.29	25	98.57	98.39
4	3	200	1	91.39	1	96.08	10	99.17	10	99.67	99.67

Table 6: Loss of 4 models with different cases

Case	Number of Hidden Layers	Batch Size	Minimum Training Loss		Minimum Validation Loss		Maximum Training Loss		Maximum Validation Loss		Overall Test Loss
			Epoch	Loss	Epoch	Loss	Epoch	Loss	Epoch	Loss	
1	3	50	9	0.0998	8	0.0961	1	4.7699	1	0.2078	0.0827
2	4	50	18	0.1006	19	0.0872	1	3.1447	1	0.2088	0.0880
3	5	80	24	0.0850	25	0.0784	1	4.2943	1	0.1400	0.0910
4	3	200	10	0.0221	10	0.0081	1	0.2508	1	0.1587	0.0081

The first hidden layer in the first case presented in figure 27 and 28 is a fully connected layer, with 20 units is used as the neural network's entrance. The ReLU activation function is applied to induce nonlinearity after receiving the input data. In this situation, the input data's shape, which is (43), is specified by the input shape. The second layer is also fully connected layer, with 40 units or number of neurons and Relu activation function is applied to the model. This layer performs a nonlinear transformation on the outputs of the previous layer. It helps the network learn complex patterns and representations from the input data. A regularization dropout layer is used next to the dense layer to reduce the model's overfitting by randomly removing 27% of its neurons.

Then the third layer is again another fully connected layer, with 80 units or number of neurons and Relu activation function is applied to the model. Similar to the previous hidden layer, this layer further extracts and learns more complex features from the previous layer's output. Again, a regularization dropout layer is used once again next to another dense layer to reduce the model's overfitting by randomly removing 27% of its neurons. The fourth layer is also fully connected layer, with 40 units or number of neurons and Relu activation function is applied to the model. To also extracts and learn more complex feature.

Finally, the fully connected output layer with sigmoid as an activation function is used, which squashes the output between 0 and 1, making it suitable for binary classification. With a batch size of 50, the model is trained over 10 epochs. The performance has an overall test accuracy of 98.24%.

The minimum validation accuracy is 89.18% at epoch 1, while the minimal training accuracy is 88.39% at epoch 1. At epoch 9, the highest training accuracy is 97.96 %, whereas the highest validation accuracy is 98.06% at epoch 10. The overall model loss, in this case, is estimated to be around 0.0827. The training loss decreases exponentially when the iteration goes until epoch 9 then up at 10. The validation loss decreases from the pick value to down when iteration goes like training loss. And the validation loss is less than training loss. The minimal validation loss is 0.0961.

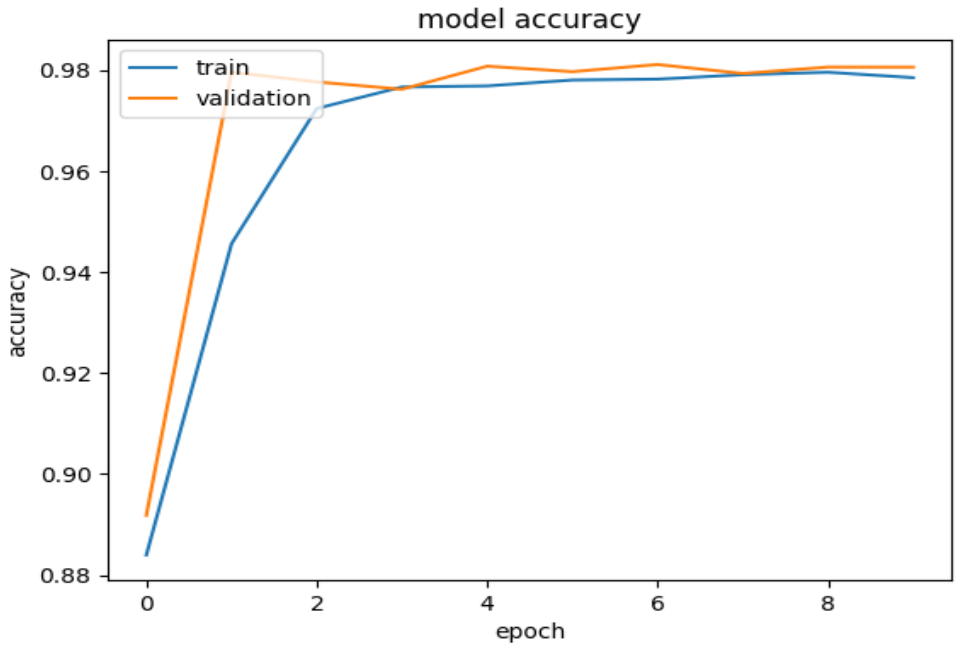


Figure 27: Observed accuracy of model 1

Figure 29 and figure 30 are defined for case 2, where fully connected layer 1, fully connected layer 2, fully connected layer 3, dropout, fully connected layer 4, and dropout are used one after another.

The first hidden layer in the first case is a fully connected layer, with 20 units is used as the neural network's entrance. The ReLU activation function is applied to induce nonlinearity after receiving the input data. In this situation, the input data's shape, which is (43), is specified by the input shape.

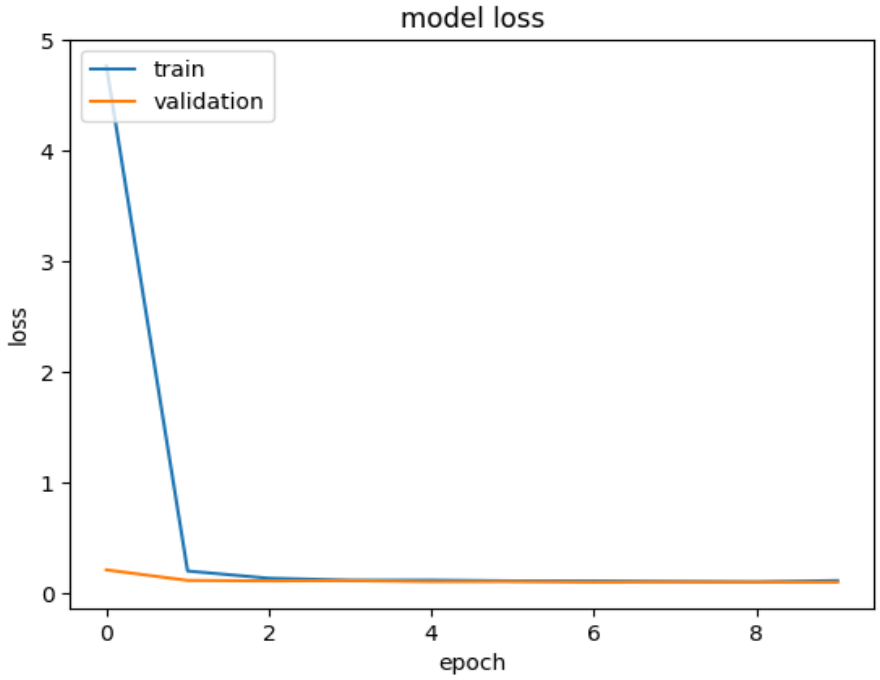


Figure 28: Observed loss of model 1

The second layer is also fully connected layer, with 40 units or number of neurons and Relu activation function is applied to the model. This layer performs a nonlinear transformation on the outputs of the previous layer. It helps the network learn complex patterns and representations from the input data. Another layer a fully connected layer, with 50 units or number of neurons and Relu activation function is applied to extract more complex features. And another a fully connected layer, with 60 units or number of neurons and Relu activation function is also applied for further pattern learning and feature extraction.

A regularization dropout layer is used next to the dense layer to reduce the model's overfitting by randomly removing 25% of its neurons. Another layer a fully connected layer, with 40 units or number of neurons and Relu activation function is applied to extract more complex features. After that a regularization dropout layer is used once again next to the dense layer to reduce the model's overfitting by randomly removing 25% of its neurons. Finally, the fully connected output layer with sigmoid as an activation function is used, which squashes the output between 0 and 1, making it suitable for binary classification. With a batch size of 50, the model is trained over 20 epochs. The performance has an overall test accuracy of 98.22%. The minimum validation accuracy is 95.90% at epoch 1, while the minimal training accuracy is 90.29 % at

epoch 1. At epoch 10, the highest training accuracy is 97.94 %, whereas the highest validation accuracy is 98.24% at epoch 12. The overall model loss, in this case, is estimated to be around 0.0880. The training loss decreases exponentially when the iteration goes. The validation loss decreases from the pick value until it reaches epoch 10 when iteration goes but at epoch 10 and 11 it increases then after that it starts decreasing again until it reaches down the last training iteration. And the validation loss is less than training loss. The minimal validation loss is 0.0872.

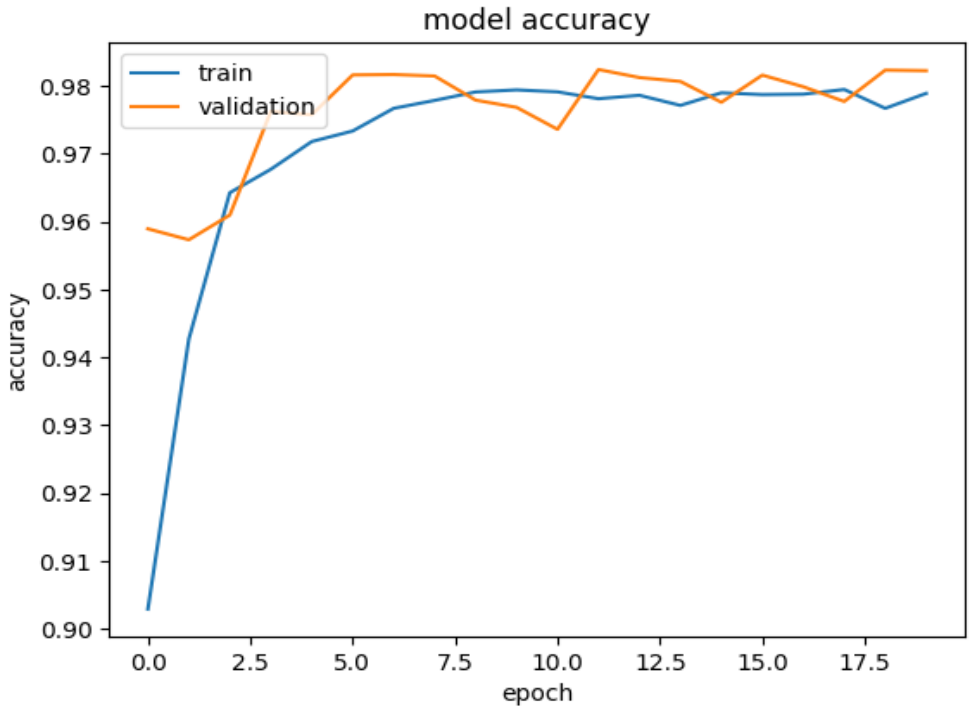


Figure 29: Observed accuracy of model 2

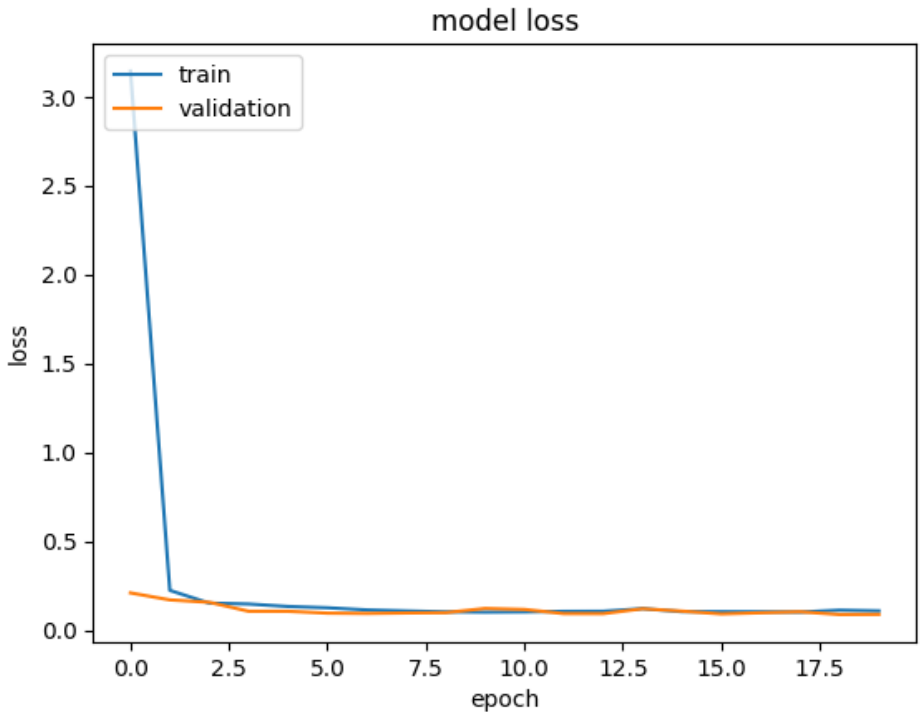


Figure 30: Observed loss of model 2

Figure 31 and figure 32 are defined for case 3, where fully connected layer 1, LSTM layer, dropout, fully connected layer 4, and dropout are used one after the other. The first hidden layer in the first case is a fully connected layer, with 20 units is used as the neural network's entrance. The ReLU activation function is applied to induce nonlinearity after receiving the input data. In this situation, the input data's shape, which is (43), is specified by the input shape.

The second layer is LSTM layer, with 40 units or number of neurons and Relu activation function is applied to the model. This layer performs a nonlinear transformation on the outputs of the previous layer. It helps the network learn complex patterns and representations from the input data. Another layer a fully connected layer, with 50 units or number of neurons and Relu activation function is applied to extract more complex features. And another a fully connected layer, with 60 units or number of neurons and Relu activation function is also applied for further pattern learning and feature extraction. And another layer a fully connected layer, with 80 units or number of neurons and Relu activation function is applied to still extract more complex features. A regularization dropout layer is used next to the dense layer to reduce the model's overfitting by randomly removing 25% of its neurons.

Another layer a fully connected layer, with 40 units or number of neurons and Relu activation function is applied to extract more complex features. After that a regularization dropout layer is used once again next to the dense layer to reduce the model's overfitting by randomly removing 25% of its neurons. Finally, the fully connected output layer with sigmoid as an activation function is used, which squashes the output between 0 and 1, making it suitable for binary classification. With a batch size of 80, the model is trained over 25 epochs. The performance has an overall test accuracy of 98.57%. The minimum validation accuracy is 94.86% at epoch 1, while the minimal training accuracy is 94.22 % at epoch 1. At epoch 25, the highest training accuracy is 98.29%, whereas the highest validation accuracy is 98.57% at epoch 25.

The overall model loss, in this case, is 0.0910. The training loss decreases exponentially when the iteration goes until epoch 13 and starts it increase at 14 and 15 epochs. Then it starts again to decrease at 16 but it increases at 17. But then decreases until it reaches the last epoch. The validation loss decreases from the pick to epoch 3 and increase at epoch 4, then it again starts to decrease at epoch 9 and then once again it starts to increase at epoch 10 and 11 it increases.

Then decreases at epoch 19 but increase at epoch 20 and 21. Then from 22 to last epoch it decreases when iteration goes. And the validation loss is less than training loss. The minimal validation loss is 0.0852.

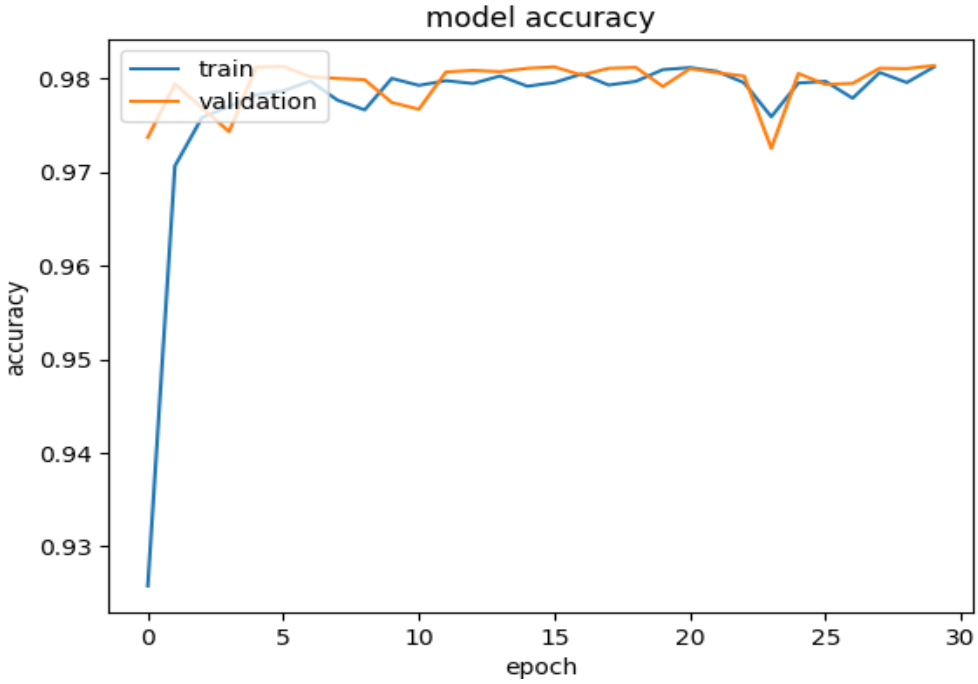


Figure 31: Observed accuracy of model 3

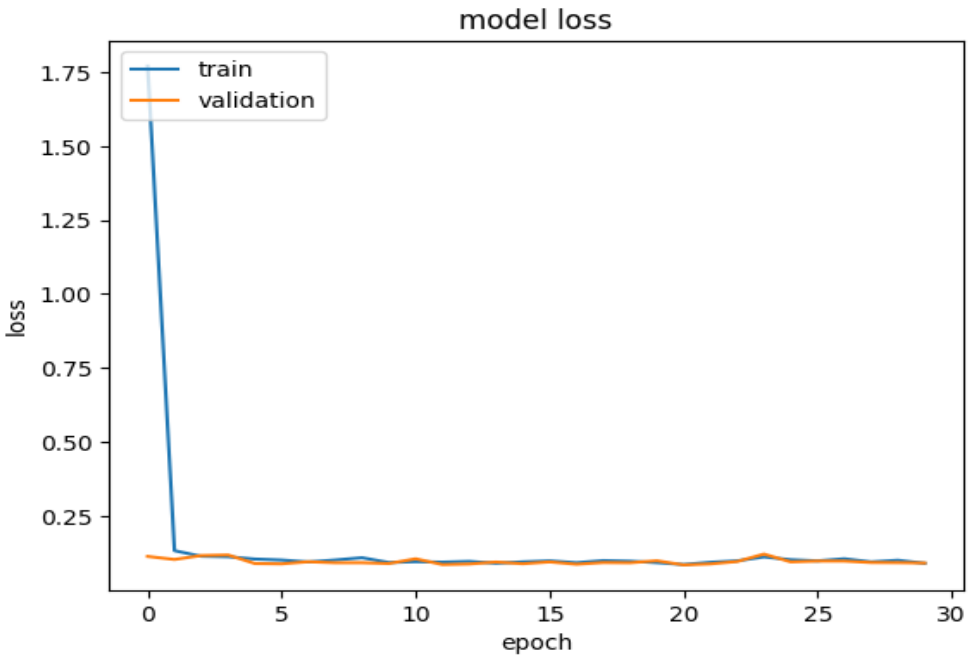


Figure 32: Observed loss of models 3

The

first hidden layer in the first case presented in figure 33 and 34 is a convolutional layer 1, which is used for feature extraction. It has 64 filters with a kernel size of 3x3 pixels, and it uses ReLU as an activation function. The second hidden layer is also a convolutional layer 2, which consists of 64 filters with a kernel size of 3x3 and ReLU. The third hidden layer is again also a convolutional layer 64, which consists of 32 filters with a kernel size of 3x3 and ReLU. After three Conv1D layers, MaxPooling1D layer is used to downscale the output feature maps, reducing their spatial dimensions by a factor of two. I used 1D max-pooling since I am working on sequential input data. A regularization dropout layer is used next to the pooling layer to reduce the model's overfitting by randomly removing 25% of its neurons.

The recurrent layer (Bidirectional Gated Recurrent Unit (GRU) layer with 100 units) is applied after regularization takes place. Then once again regularization dropout layer is applied to the recurrent to minimize overfitting (0.25). Finally, the fully connected output layer with sigmoid as an activation function is used, which squashes the output between 0 and 1, making it suitable for binary classification

With a batch size of 200, the model is trained over 10 epochs. The performance has an overall test accuracy of 99.67%. The minimum validation accuracy is 96.08% at epoch 1, while the minimal training accuracy is 91.39 % at epoch 1. At epoch 10, the highest training accuracy is 99.17%, whereas the highest validation accuracy is 99.67% at epoch 10.

The overall model loss, in this case, is estimated to be around 0.0081. The training loss decreases exponentially when the iteration goes until it reaches the last epoch. The validation loss decreases from the pick value to down when iteration goes like training loss. And the validation loss is less than training loss. The minimal validation loss is 0.0081.

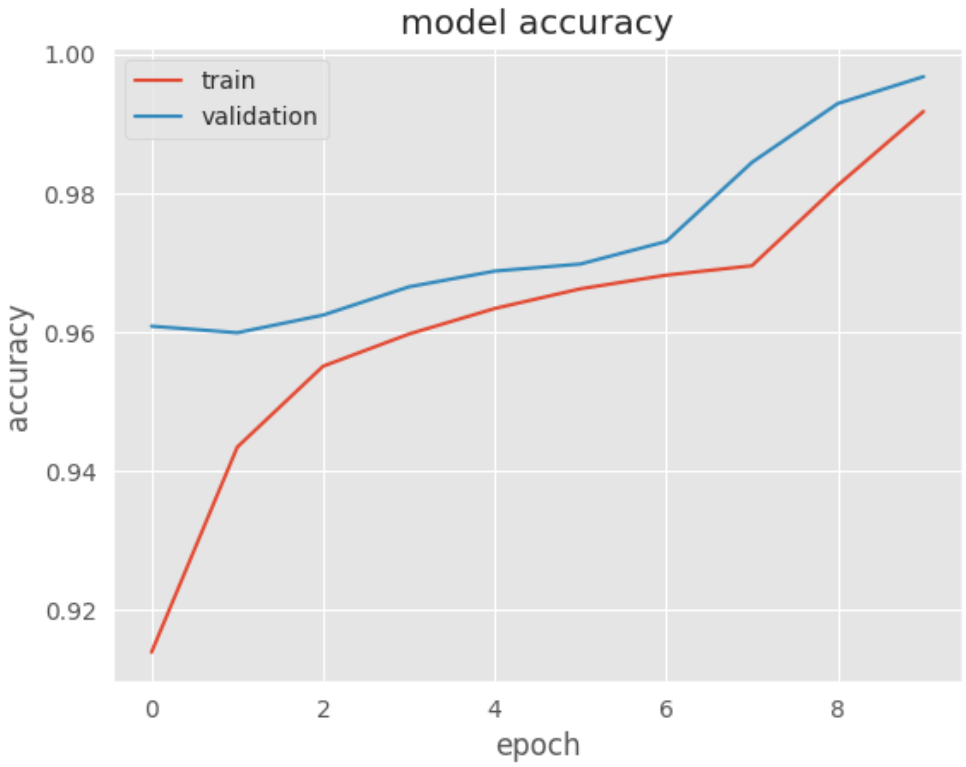


Figure 33: Observed accuracy for model 4

Figure 34: Observed loss of model 4

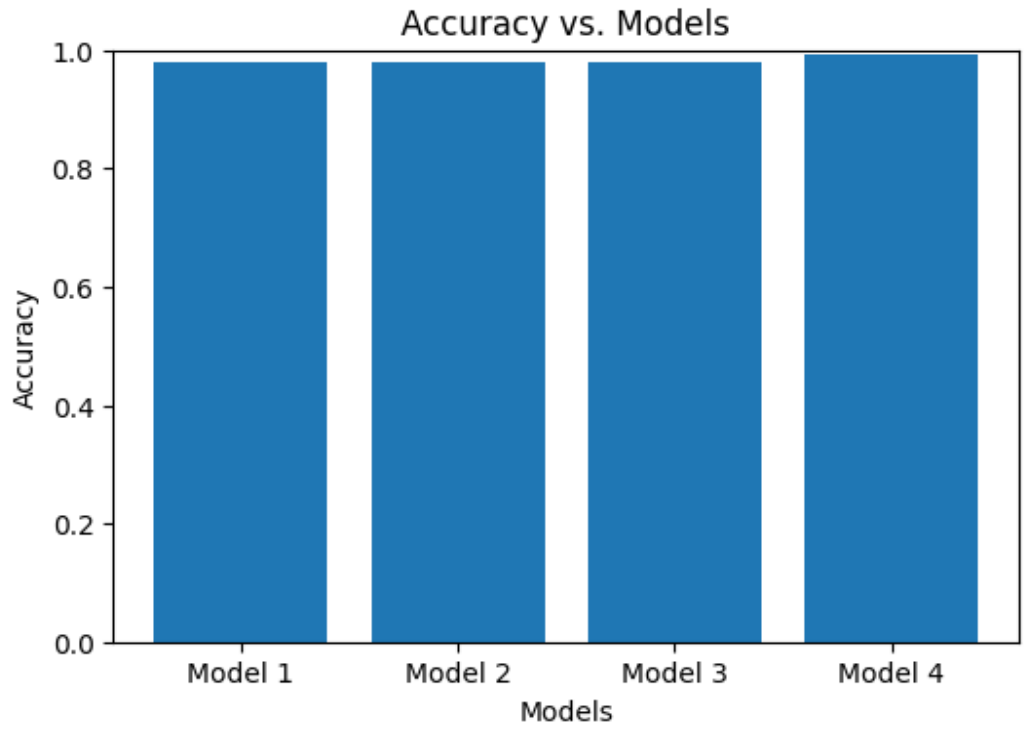


Figure 35: Accuracy of the different models

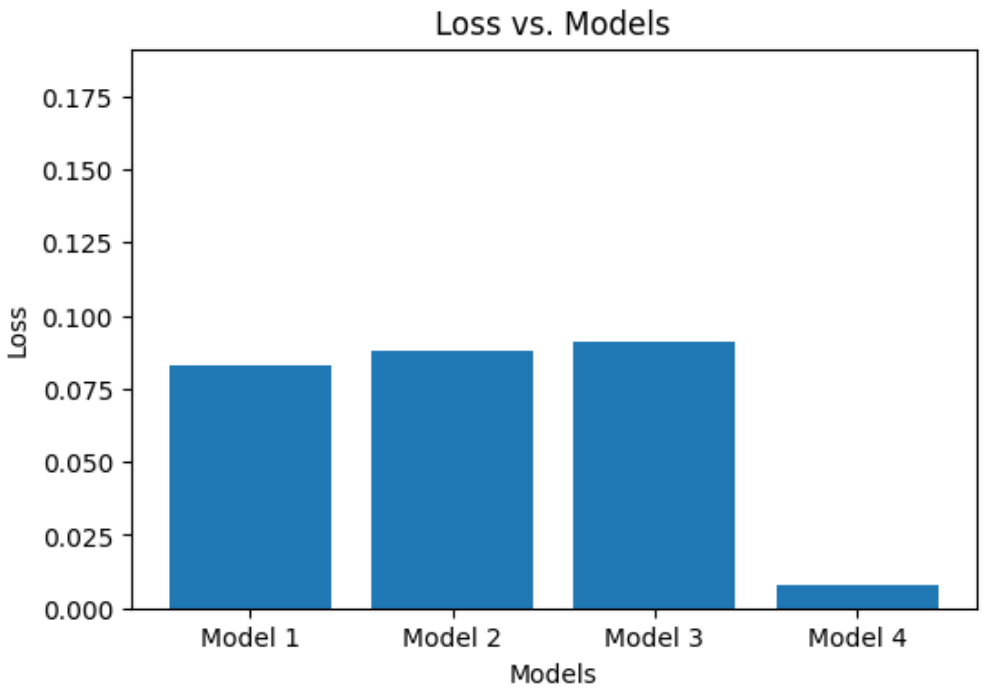


Figure 36: Loss of the different models

By varying the epochs, batch size and hidden layers, the change of the performance for loan eligibility prediction was observed during the experiment. Accuracy curves for the four cases for each parameter were generated using a loan data-set. The four cases behave differently due to the different batch size and epochs and hidden layers. As shown in the figure 6.9, the highest test accuracy in performance was found to be 99.67% for 10 epochs and 200 batch size in case 4 among all the observations((Conv1, Conv2, Conv3, pool1, dropout , bidirectional(GRU), dropout and a fully connected layer).This type of highest accuracy will work in Loan Eligibility Prediction to help the machine execute more efficiently.

In case 3, however, the lowest accuracy among all observations in the performance was discovered to be 98.14% (a fully connected layer input (dense layer 1), dense layer 2, dense layer 3, dense layer 4, dense layer 5, dropout, dense 5, dropout and final dense layer). Furthermore, the total highest model loss in case 3 is 0.0911, while the total lowest model loss in case 4 with dropout is around 0.0081 (Figure 6.10). CRNN will be able to learn new complex input loan data with thus minimal loss.

We select the optimal model from four scenarios based on the observed result, with the highest model test accuracy and the lowest test loss. Since case 4 has the lowest loss of 0.0081 and the best accuracy of 99.67%, it is the model we propose for this research.

And the following code snippet shows how our proposed model testes:

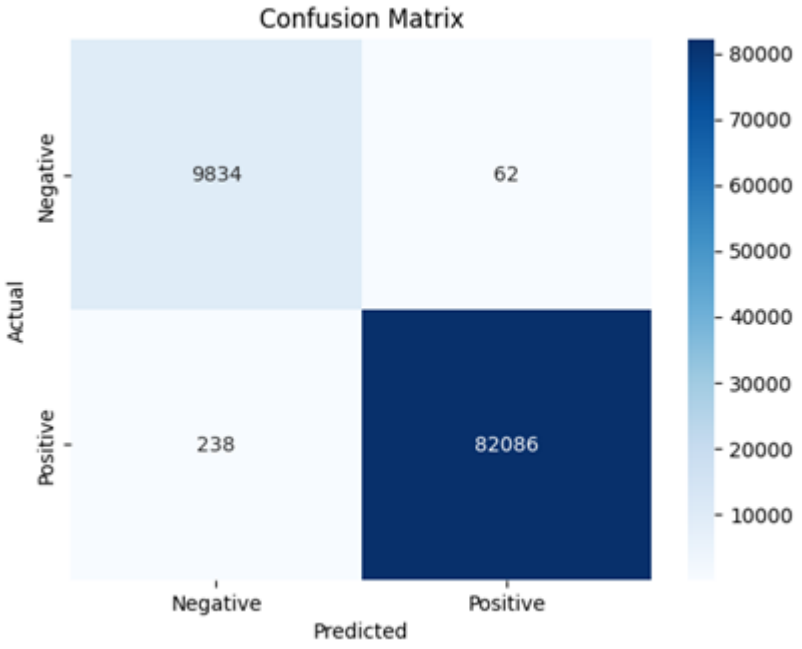
```
score = model.evaluate(X_test_stratified, y_test_stratified)

print(score)
```

```
2882/2882 [==============================] - 22s 8ms/step - loss: 0.0081 - accuracy: 0.9967
[0.008134355768561363, 0.9967468976974487]
```

Figure 37: Model evaluation

The model was evaluated using confusion matrix also, and it showed as follows:



As we can understand from this

- 9,834 -> instances were classified as true negatives (TN) they were originally a negative value or (class 0) instances and they were correctly predicted or classified as class 0.
- 62 - > instances were classified as false positives (FP) they were originally a positive value or (class 1) instances and they were incorrectly predicted or classified as class 0.
- 238 -> instances were classified as false negatives (FN) they were originally a negative value or (class 0) instances and they were incorrectly predicted or classified as class 1.
- 82,086 ->instances were classified as true positives (TP) they were originally a positive value or (class 1) instances and they were correctly predicted or classified as class 1

Table 7: Model comparison of CRNN vs. ANN vs RNN

No.	Model	Number of Hidden Layers	Epochs	Batch Size	Test Accuracy	Test Loss
1.	Artificial Neural Network (ANN)	3	10	50	98.25%	0.0827
2.	Artificial Neural Network (ANN)	4	20	50	98.23%	0.0880
3	Recurrent Neural Network (RNN)	5	25	80	98.14%	0.0911
3.	Convolutional Recurrent Neural Network (CRNN)	4	10	200	99.67%	0.0081

So, as I have discussed earlier in this research, we have conducted experiments using other deep learning algorithms such as Artificial Neural Network (ANN) to compare them with the CRNN model that we proposed. Their accuracy and loss are shown Table 6.3, as we can see the CRNN model outperforms other ANN models in terms of accuracy and loss. Compared to CRNN, ANN has a lower performance when using it with the loan dataset. This concludes that the CRNN algorithm that we proposed is the best.

The previous work on Loan Eligibility Prediction is done by (Infant Cyril et al., 2022) achieved 95% accuracy using an (SBCO)-based deep neuro fuzzy network. Compared with the previous work, we improve the accuracy of the prediction from 95% to 99.67% by using CRNN, increasing the dataset size, and enhancing the quality of the image by using pre-processing techniques on the dataset.

Table 8: Model comparison between proposed and existing

No.	Model	Approach	Accuracy
1	CRNN(Proposed Model)	CNN + RNN with Conv1D and Bidirectional GRU	99.67%
2	SBCO model	SSD + BCO -based deep neuro fuzzy network	95%

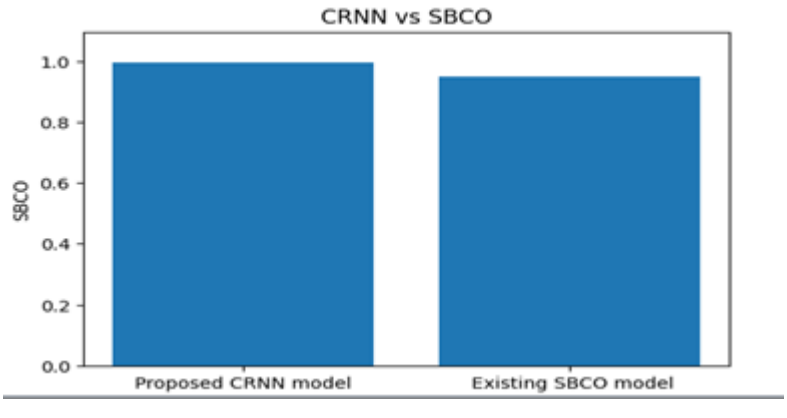


Figure 38 : Proposed vs Exsisting model

6.3 Research Question Discussion

This research answers to each of all the research questions raised in the first chapter in 1.4 Research questions. The followings are the answers to the questions:

Answer for Q1: As we discussed earlier most research in loan eligibility prediction done using machine learning and one found with using SBCO-based deep neuro fuzzy network. In this research ask if the performance of the prediction model can be improved using deep learning. And of course, as we investigate through this study the deep learning algorithms shows it was possible to improve the performance of the model using deep learning.

Deep learning algorithms are preferable for loan eligibility prediction due to its capacity to handle complex and non-linear relationships within the data. In the proposed deep learning model (CRNN model) every layers plays a vital role to achieve the goal by extracting and learning complex features without relying on handcrafted features that require domain expertise

and manual feature engineering. In such a way the performance in terms of accuracy and loss is improved using deep learning when compared to existing traditional ones.

Answer for Q2: We employ parameters and hyper-parameters in deep learning models to fine-tune the model's performance. Weight and bias are the parameters of neural network models. Initially start with random values and update those values by using error cost functions(optimization) like Adam and learning rate. The deep learning models undergo numerous iterations (epochs) of training. Based on the learning rate and cost function values for each epoch, the weight and bias are updated. Learning rate, epoch, the quantity of hidden layers, and batch size are hyper-parameters that are used to enhance prediction models. Each of them is utilized to enhance the model's functionality and identify the parameters' ideal values.

Answer for Q3: Deep learning-based models, like the proposed model Convolutional Recurrent Neural networks (CRNNs), in particular, can address a number of difficulties in the loan sanctioning process to identify qualified loan applicants. First off, complicated non-linear interactions between applicant attributes, financial information, and loan eligibility requirements can be accurately captured by CRNN models. They can extract useful aspects from the applicant's data, including income, credit history, employment status, and demographic data, by utilizing convolutional layers. The CRNN model can handle sequential data, capturing temporal dependencies, and seeing patterns in payment behavior and credit history thanks to the recurrent layers. As a result, the model is better equipped to forecast future outcomes based on the applicant's likelihood of repaying the loan and previous financial activity.

Additionally, CRNN models may deal with input sequences of variable lengths, accommodating the various financial records or history data that are accessible for each application. CRNN models, which make use of deep learning techniques, offer a strong tool for analyzing and deciphering complex applicant data, resulting in better loan sanctioning decisions based on the identified qualified applicants.

Answer for Q4: The effectiveness and productivity of banks and the lending industry can be significantly increased by using the CRNN (Convolutional Recurrent Neural Network) loan eligibility prediction model. The model can examine a wide range of financial and personal data points to accurately assess and select candidates for loan sanctioning by utilizing its advanced feature extraction skills. Not only the loan approval procedure is streamlined, but also the chance of human error in manual evaluations is decreased.

Through the use of the CRNN model, banks may speed up the processing of loans, increase accuracy, and make data-driven decisions, all of which will lead to greater productivity and efficiency in the lending industry.

Answer for Q5: Deep learning loan eligibility prediction model (CRNN) provides numerous advantages to banks. To give more accurate loan eligibility predictions, these algorithms use their capacity to evaluate complicated patterns in vast datasets. This enhanced precision enables banks to make more informed judgments and reduces the danger of lending to high-risk borrowers. Banks can save time and money by automating and streamlining the loan assessment process, resulting in cost savings. Furthermore, this model allows for faster loan approvals, increasing efficiency and improving client happiness and can also successfully manage risk and reduce losses by detecting possible risks and patterns of default.

Additionally, CRNN model enable customized loan offerings based on individual borrower profiles, enticing more clients and increasing loyalty and also encourage adherence to regulatory rules and fair lending procedures, resulting in unbiased loan eligibility choices.

CHAPTER SEVEN

7. CONCLUSION AND FUTURE WORK

7.1. Conclusion

In this research work, we used Convolutional Recurrent Neural Networks to predict eligible loan applicant in the given loan data-set. The dataset in this research was collected from online kaggle data-set (Lending club data-set). The data-set consists of 44 columns and 461,097 entries. We used CRNN architecture from the deep learning approaches to develop Loan Eligibility prediction model.

To find the best-fit model for CRNN-based architecture, numerous trial-and-error neural network configuration tweaking techniques were employed. We were able to increase prediction accuracy utilizing the proposed CRNN model compared to past research efforts on loan eligibility prediction. Our accuracy was 99.67%, and our model loss was 0.0081.

To achieve this model deep learning algorithms CNN and RNN were utilized to create CRNN model for the loan eligibility prediction. And the proposed model perfectly trained and tested then comes up with 99.67% accuracy which outperforms the existing deep-learning based SBCO model by 4.67 in terms of overall tested accuracy.

7.2. Future Work

Although this thesis covers a lot of area, certain things and ideas still need to be refined or added. The data used in this study came from online kaggle data-set, which is lending club data-set which may leads biased prediction if we want to use the proposed model for our country loan data-sets. Since the model is trained with this lending club data-set, it learnt this organization’s data-set behavior. So, if our country bank or lending authorities loan dataset differs significantly from the lending club loan data-set in terms of data distribution, feature representation, or underlying patterns, there is a high chance of domain shift. In such cases, the model's performance may be negatively affected, leading to biased predictions.

But in the future work in order to use this deep learning model (CRNN) in our country or in order to let this model to be used by our countries banks and lending authorities, the model have to be trained using their own data-set, at least the model has to know our country lending behavior.

In general, this thesis covers a wide range of concepts, but there are other things also that need to be done and I have mentioned one above.

SPECIAL ACKNOWLEDGMENTS

This research project is funded by Adama Science and Technology University under the grant number:

ASTU/SM-R/829/23

Adama, Ethiopia

REFERENCES

A., R. , & S, N. (2021). . *Loan Eligibility Prediction Using Classifiers, . 5*(03(04),).

Al-qerem, A., Al-Naymat, G., & Alhasan, M. (2019). *"Loan default prediction model improvement through comprehensive preprocessing and features selection", Proceedings of International Arab Conference on Information Technology (ACIT),. pp. 235-240*.

Aphale, A. S. , & Shinde, S. R. (2020). *Predict Loan Approval in Banking System Machine Learning Approach for Cooperative Banks Loan Approval . 09*.

Diwate, Y. , Rana, P. S. , & Chavan, P. A. (2023). *Loan approval prediction using machine learning. International Research Journal of Modernization in Engineering Technology and Science*. *04*.

Elette, S., Saad, Y., & Ghaleb, E. (2010). *Neuro-Based Artificial Intelligence Model for Loan Decision.*

Gao, X., Xiong, Y., Xiong, Z., & Xiong, H. (2021). *Credit Default Risk Prediction Based On Deep Learning* . https://doi.org/https://doi.org/10.21203/rs.3.rs-724813/v1

Handzic, M. , Tjandrawibawa, F., & Yeo, J. (2003). *How Neural Networks Can Help Loan Officers to Make Better Informed Application Decisions*. *13*.

Infant Cyril, G. L, & Ananth, J. P. (2022). *Deep learning based loan eligibility prediction with Social Border Collie Optimization. Kybernetes.* https://doi.org/https://doi.org/10.1108/K-10-2021-1073

Kuma, M., Goel, V., Jain, T., Singhal, S., & & Goel, Dr. L. (2018). *Neural Network Approach to Loan Default Prediction*. *05*.

Kumar, A. , Sharma, S., & & Mahdavi, M. (2021). *Machine Learning (ML) Technologies for Digital Credit Scoring in Rural Finance: A Literature Review.*

Meshref, H. (2020). *Predicting Loan Approval of Bank Direct Marketing Data Using Ensemble Machine Learning Algorithms. International Journal of Circuits, Systems and Signal Processing*. *14*(914–922). https://doi.org/https://doi.org/10.46300/9106.2020.14.117

Orji, U., E., U., Chikodili. H, Nguemaleu, J., C. N, & & Ugwuanyi Peace. N. (2022). *Machine Learning Models for Predicting Bank Loan Eligibility. In 2022 IEEE Nigeria 4th International Conference on Disruptive Technologies for Sustainable Development (NIGERCON), . 1–5.* https://doi.org/https://doi.org/10.1109/NIGERCON54645.2022.9803172

Reddy, M. J., & Kavitha, B. (2010). *"Neural networks for prediction of loan default using attribute relevance analysis", in Signal Acquisition and Processing,. pp. 274-277.*

S, R. , Jha, P., V, I., H, S., & & Zafar, N. (2021). *Monetary Loan Eligibility Prediction using Machine Learning.,. 11.*((No.07)).

Shaik, A., Asritha, K., Lahre, N., Joshua, B., & & Velagapudi, V. (n.d.). *Customer Loan Eligibility Prediction using Machine Learning., . 13.*

Sheikh, M., A., G., & A. K. (2020). *An Approach for Prediction of Loan Approval using Machine Learning Algorithm. In 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). 490–494*. https://doi.org/https://doi.org/10.1109/ICESC48915.2020.9155614

Singh, V. , Yadav, A., Awasthi, R., & and Partheeban, G. N. (2021). *"Prediction of modernized loan approval system based on machine learning approach", The Proceeding of International Conference on Intelligent Technologies (CONIT).*

Tsai, M. C., Lin, S. P., Cheng, C. C., & and Lin, Y. P. (2009). *"The consumer loan default predicting model– An application of DEA–DA and neural network", Expert Systems with*

Applications,. Vol. 36 No. 9,(pp. 11682-11690.).

TY - CHAP AU, Wu, M. A., Du, C. A., Huang, Y. A., Cui, X. A., & Duan, J. (2021). *Investigation on Loan Approval Based on Convolutional Neural Network DO - 10.1007/978-3-030-78615-1_18 ER - . SP-203 EP*.

UCD Michael Smurfit Graduate Business School, Kumar Gupta, D., & Goyal, S. (2018). *Credit Risk Prediction Using Artificial Neural Network Algorithm. International Journal of Modern Education and Computer Science,, . 10(5)*(9–16.). https://doi.org/https://doi.org/10.5815/ijmecs.2018.05.02

Udbhav, M., Kumar, R., Nitin, K., Rohit, K., Vijarania, D. M. , & & Gupta, S. (2022). *Prediction of Home Loan Status Eligibility using Machine Learning. In Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2022. Available at. 4121038.*

Vedapradha, R., & Hariharan, R. (2022). *Most commonly used A.I. application in investment banking worldwide. Retrieved from .*

Xu, J., & Xie, Y. (2021). *Loan default prediction of Chinese P2P market: A machine learning methodology. 11*.

Yu, L., & and Zhang, Y. Q. (2005). *"Evolutionary fuzzy neural networks for hybrid financial prediction", IEEE Transactions on Systems, Man, and Cybernetics, Part C, Applications and Reviews,, . Vol.35, No. 2*(pp. 244-249.).