Credit Default Prediction Using Bi-Directional LSTM and Attention Layers: A Comparative Study

By

Merit Choorappadil Thomas

CHO21538916

Submitted to
The University of Roehampton

In partial fulfilment of the requirements
for the degree of

MASTER OF SCIENCE IN DATA SCIENCE

Abstract

Through the use of machine learning techniques, this initiative seeks to improve financial organisations' appraisal of credit card customers. The main goal is to use two important variables—missed credit card payments and successive missed payments—to build an advanced model that forecasts customer behaviour. The planned model looks for relevant trends in customer behaviour to detect prospective credit defaults over the coming month. To do this, a variety of machine learning techniques will be used in order to improve the predictability of credit defaults.

Implementing feature selection and dimension reduction methods is part of the project's initial phase, which aims to improve classifier learning. Techniques like SMOTE will be used to solve the issue of unbalanced datasets. The context of customer purchasing behaviours will be established using Bidirectional LSTM with attention layers as dynamic pattern recognition classifiers. Estimating consumer behaviour scores will be aided by the internal feature extraction procedure of the LSTM.

To maximise its ability to predict credit card payment behaviour and utilisation rates, the model will be improved. The evaluation will be done by comparing the model's predictions to historical default data and external credit risk indicators to see how well it can identify high-risk accounts. Standard criteria used in the review process include accuracy, precision, recall, and F1 score.

In the comparative analysis of this project, the created LSTM model will be compared to three classical machine learning approaches, namely Logistic Regression, Support Vector Machine and Decision Tree. The work seeks to propose a robust model that improves credit card default prediction within the banking industry by investigating several machine-learning approaches and integrating their capabilities. The findings of this study may enable financial institutions to make more sensible and well-informed business decisions, reducing the risk of credit default and boosting overall operational effectiveness.

Declaration

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

Signed

Merit Choorappadil Thomas

06/09/2023

Acknowledgement

To everyone who helped this project report come to fruition, I would like to offer my profound gratitude and admiration.

First and foremost, I want to express my sincere gratitude to Mrs. Kimia Askir, who has served as my project adviser, for her invaluable direction, constant support, and professional counsel throughout this project. The direction and calibre of this work have been greatly influenced by their mentoring.

I am also incredibly grateful to the teachers and staff at the University of Roehampton for providing the tools, access to the library, and supportive learning environment that made it possible to conduct this study and write this report. I want to express my gratitude to my friends and classmates who helped with the project and encouraged me along the way with their advice and kind comments. Their cooperation has provided drive and inspiration.

To my acquaintances, I'd like to convey my appreciation for their tolerance and encouragement during the challenging portions of this project. My dedication to perfection has been motivated by their everlasting confidence in my abilities.

Last but not least, I would like to thank everyone whose work and research on this subject provided an invaluable source of inspiration and reference for this study.

Without each of these people and organizations' specific contributions and support, this project would not have been possible. I sincerely appreciate their assistance.

Merit Choorappadil Thomas

University Of Roehampton

September 2023

Table of Contents

1. Introduction **07**

1.1 Statement of the Problem and Aims 07

1.2 Objectives 07

1.3 Legal, Social, Ethical and Professional Considerations 07

1.4 Background 08

1.5 Report Overview 08

2. Literature Review **09**

2.1 Literature Review 09

2.2 Technology Review 10

3. Methodology **14**

3.1 Data Collection 14

3.2 Pre-processing 15

3.3 Data Splitting 15

3.4 Feature Selection and Extraction 16

3.5 Handling Class Imbalance 16

3.6 Model Training 16

3.7 Proposed Model 16

3.8 Project Management Tool 17

4. Implementation **19**

4.1 Pre-processing and Exploratory Data Analysis 19

4.2 Feature Importance and Outlier Removal 21

4.3 Data Loading and Splitting 22

4.4 Model Architecture 23

4.5 Model Training 25

4.6 Performance Evaluation 26

5. Results and Discussions **29**

5.1 Logistic Regression Results 29

5.2 Random Forest Classifier Results 30

5.3 SVM Results 31

5.4 Bidirectional LSTM with Attention Results 32

5.5 Evaluation 34
6. Conclusion **37**
6.1 Outcome and Achievement of Aims 37
6.2 Key Output and Discoveries 37
6.3 Reflection and Lesson Learned 37
6.4 Goal Achievement and Hindsight Analysis 37
6.5 Future Work 38
7. References 39
8. Appendices 40

1. INTRODUCTION

Financial institutions, often banking institutions are corporate bodies that offer services as middlemen in various financial and monetary transactions. The main function of financial institutions like banks is to accept deposits and lend loans. Nowadays banks change their service strategies of lending loans by way of providing credit cards to the customer. Hence it can be considered one of the main income or beneficiary sources for these institutions. The banks must consider many factors like credit score, and spending history before lending credit cards to their customers to reduce their risk of default. Banks have huge databases and proper understanding and analysis of these databases results in improved performance and making progressive business decisions. Various socioeconomic aspects influence each customer's ability to repay the loan and their ability to maintain their financial stability. In light of their payment and purchasing habits, customers are thus treated noticeably differently.

1.1 Statement of the Problem and Aims

This study's major goal is to assist bank management by modelling and anticipating customer behaviour in light of two criteria for assessing credit card users using machine learning. Using consumers' credit card missing payments, whether one or more are consecutive. The suggested enhanced model is expected to be able to accurately predict customer credit default for the upcoming month by capturing significant patterns in consumer behaviour. The efficiency of predicting credit default is expected to be improved by leveraging the strengths of various Machine Learning techniques in combination. Preliminary investigations will be conducted to develop a credit default prediction model that solely relies on attention and a transformer architecture, without employing recurrent networks for sequence processing.

1.2 Objectives

The project's initial development would be to incorporate feature selection and dimension reduction algorithms, for optimizing the process of learning classifiers. The problem with imbalanced data sets could be solved with the use of techniques like SMOTE. As a next step, LSTM recurrent neural networks will be employed as a dynamic pattern recognition classifier which would construct a context of consumer spending behaviour. LSTM feature extraction internally takes place and is hidden from the observer. It is used for the estimation of customer behaviour scores [1].

Fine-tune the model to optimize its ability to predict customers' credit card payment behaviour and utilization rates. By contrasting the model's predictions with past default data or outside indicators of credit risk, you may gauge how well it can detect high-risk accounts. Utilize appropriate evaluation measures for the trained model, such as F1 score, recall, precision, and accuracy. Three traditional machine learning algorithms are contrasted with the built LSTM model: logistic regression, decision tree, and support vector machine.

1.3 Legal, Social, Ethical and Professional Considerations

Credit default prediction algorithms create significant legal, social, ethical, and professional issues in the context of this undertaking. Legally speaking, it is essential to follow data protection, privacy, anti-discrimination, and financial standards given that sensitive personal data is used in credit scoring. To prevent unauthorised access to personal information about persons, data privacy and security must be guaranteed. Ethics issues are also relevant since it's essential to deal with bias and guarantee justice to avoid unfair outcomes based on things like colour, gender, or ethnicity.

Furthermore, the credit default prediction model's decision-making process must be transparent and explicable in order to be ethical. The model's forecasts should be clearly understood by consumers in terms of how they affect their financial future. It is crucial to get people to consent after being fully informed.

Before collecting any employee data, a machine learning project should seek their approval. It should also explicitly explain why and how the data will be used and ensure data accuracy, safe storage, and non-discriminatory usage. Employees may be more inclined to voluntarily submit their data if it is demonstrated how machine learning may enhance data processes and help in detecting performance concerns [3].

Furthermore, accurate projections that don't subject people to unneeded dangers require precision, dependability, and ongoing monitoring. Responsible professionals must uphold ethical standards of justice, honesty, and openness in the workplace. To maintain equitable financial services, it is important to take into account how credit default prediction models affect the availability of loans and financial possibilities in society. The ultimate guiding principles for people involved in developing and applying credit default prediction models are accountability and responsibility. It is crucial that people are conscious of the effects of their activities. Building trust and promoting moral and responsible financial behaviour are two benefits of using these models in a way that is open, fair, and consistent with ethical and legal standards.

1.4 Background

Banks must evaluate clients' credit and behaviour in order to determine whether they are good or bad. In order to define the term, Anderson [2] divided it into its two constituent parts: "credit," which refers to the practice of buying something and paying for it later, and "scoring," which describes the procedure used for credit cards. The two primary types of credit scoring are behavioural scoring and application credit scoring, in which a score is utilized to help determine whether to approve a new credit application or to address existing clients once a loan has been approved [2]. A rise in using credit cards increases the chances of missing payments because the same individual has many credit cards from different banks leading to an increase in the probability of missing payments. As a result, financial institutions are under pressure to continuously improve the system which will detect early missing payment predictions in order to mitigate substantial financial losses associated with these card transaction activities. A perfect deep-learning model needs to predict these missing payments in future.

1.5 Report Overview

1.5.1 Literature Review

The pros and weaknesses of various strategies are shown in this section's thorough analysis of the body of research on credit default prediction models. It highlights pertinent research using the LSTM and BiLSTM designs for related applications.

1.5.2 Technology Review

Explain how BiLSTM is different from conventional LSTM models in a quick introduction to the model's design. Declare the meaning of the term "credit default prediction" and the significance of this idea to the financial sector.

1.5.3 Methodology

The processes for data collection, preprocessing, and research design are described in the methodology section. It explains the dataset that was utilised, the characteristics that were chosen, and the justification for their inclusion. There is also extensive information on the model architecture, hyperparameter adjustment, and assessment measures.

1.5.4 Implementation

The stages of data preparation like data cleaning and preprocessing are carried out here. Here conventional algorithm implementation is explained. In addition to that, it also explains how bidirectional LSTM is implemented and its architecture structure and its working are also explained here.

1.5.5 Result and Discussions

The results of conventional algorithms and Bidirectional LSTM are discussed here. The comparison of the results is also discussed and visualized in this section.

1.5.6 Conclusion

Here the outcomes of this particular project work and its future works are discussed. A comparison with related work and future implementation of this project are also included in this section.

1.5.7 Reference

An extensive list of references is included in the paper, including all the books, articles, and studies that were utilised to conduct the study and create the model.

2. LITERATURE REVIEW

2.1 Literature Review

Credit default prediction is a critical aspect of risk assessment for lending institutions [4]. Rigorous research has been conducted to assess the performance of different algorithms, including Logistic Regression, Decision Trees, Neural Networks, Support Vector Machines (SVM), and Least Square SVM. Among these algorithms, LS-SVM and Neural Networks have demonstrated superior performance and are recommended as potential replacements for traditional credit scoring models.

Mathematical techniques such as Decision Trees, Logistic Regression, and Linear Regression are commonly employed for analysing borrower behaviour scores. These techniques take into account economic conditions and transform default probabilities into estimates of potential profit or loss. Additionally, Markov chain processes are utilized to simulate changes in credit scores and delinquent status, while dynamic models are employed to assess consumer credit risk more comprehensively.

A comprehensive literature review conducted by Louzada F underscores the significance of binary classification in credit score analysis. This review emphasizes the importance of credit rating procedures and historical shifts in creditworthiness. Notably, it suggests that utilizing fixed time periods for analysis may lead to unstable behaviour pattern forecasts, advocating instead for a more effective 12-month performance period [4].

In the realm of credit card holder behaviour scoring, research studies have delved into the application of data mining techniques. Among these techniques, Backpropagation Neural Networks have demonstrated superior accuracy in scoring credit card user behaviour. Furthermore, timing default events in conjunction with neural networks have shown promise in enhancing prediction accuracy. Additionally, survival analysis techniques have been employed to simulate credit card account defaults while incorporating macroeconomic variables, resulting in improved model fit and default prediction accuracy. Dynamic models have also emerged as valuable tools for enhancing default forecasts.

Beyond traditional financial data, research has explored the connection between attitudes, personality factors, and credit card usage through mail-in questionnaires. These studies have identified specific credit card features that may potentially encourage indebtedness. Moreover, big data sources, such as Weibo user behaviour, have been leveraged to evaluate creditworthiness and determine credit scores, reflecting the evolving landscape of data sources in credit assessment.

As credit card usage continues to rise, the limitations of traditional behavioural scoring models become more pronounced due to data biases. To address these challenges, Bastani K has proposed a two-stage scoring system that places a strong emphasis on both default prediction and profitability assessment, offering a more comprehensive approach to credit risk evaluation [5,6].

2.2 Technological Review

Technological developments in Bidirectional LSTM:

Bidirectional LSTMs capture both forward and backward dependencies in sequential data, making them ideal for modelling complex spending patterns, especially in financial data like credit card transactions. Bidirectional LSTMs handle noisy data well, making them suitable for scenarios with missing or incomplete transaction records. They effectively capture sequence dynamics [7]. Compared to other sequential data models, Bidirectional LSTMs demonstrate better accuracy in modelling credit card usage patterns, thanks to their ability to capture information from sequences in both directions.

The banking industry relies on real-time credit card usage analysis for risk management. Bidirectional LSTMs are computationally efficient, enabling their use in real-time analysis [8]. Incorporating External Data: Bidirectional LSTMs can integrate external data like demographics, geography, and transaction amounts, enhancing the accuracy and predictive power of credit card usage models.

2.2.1 Benchmark Models

Benchmark models for credit default prediction:

2.2.1.1 Support Vector Machine (SVM): SVMs are innovative machine-learning techniques designed to solve credit rating and order problems. They transform data into higher-dimensional spaces using kernel functions, making them efficient in high-dimensional spaces.

2.2.1.2 Logistic Regression: Logistic regression is a widely used statistical method for creating credit scoring models. It models binary outcome variables, making it suitable for classification tasks.

2.2.1.3 Decision Tree (Random Forest): Decision trees, including random forests, are employed for categorization due to their interpretability and ability to handle complex decision boundaries.

These benchmark models are compared with Bidirectional LSTMs to evaluate their accuracy and performance in predicting credit defaults [9, 10].

2.2.1.4 Bidirectional LSTM

The recurrent neural network is a powerful model for sequence data and good for labelling input and output labelling if the labelling is unknown for classification problems. The combination of a Recurrent neural network with long short-term memory is considered to be more fruitful [11]. The performance of bidirectional LSTM is good compared to unidirectional LSTM and conventional recurrent neural networks. Bidirectional neural networks process input data in both forward and backward directions and hence can process both current and future data [12].

Recurrent neural networks' cell connections allow for the creation of directed cycles. Each cell has a secret state, which is updated every time using its prior values. A memory and internal network state are both established by such a framework.

These are the RNN equations:

$$\begin{cases} S_t = f(U \cdot x_t + W \cdot s_{t-1}) \\ h_t = g(V \cdot s_t) \end{cases}, \qquad (1)$$

x stands for "input vector," "s" for "hidden vector of RNN layer values," "h" for "output vector of RNN layer values," "U" for "weight matrix from input layer to hidden layer," "V" for "weight matrix from hidden layer to output layer," "W" for "weight matrix from previous time point to current time point of hidden layer," and "g" and "f" for "activation functions for output and hidden layers," respectively. Fig. 1. displays the operation of a single RNN cell.

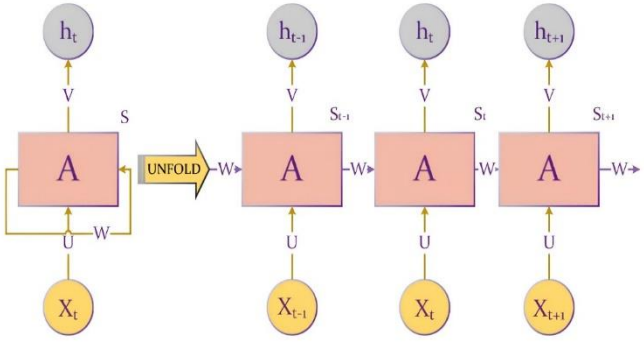


Fig. 1. RNN Model structure

We feed the cell the time series signal X one element at a time. Te vector X may be output from another vector or an input vector. RNN cells taken from the layer before. The RNN cell maintains its present state. At each iteration t, the state st and output ht are determined using equation (1). Because of their structure, RNNs can [13]: On the other hand, developing long-term reliance has its difficulties. Because RNN is prone to gradients disappearing during training [14], it is difficult to comprehend long-term interdependence. In order to address this problem, Hochreiter and Schmidhuber [13] proposed an LSTM based on RNN. The event of the network's inputs always affects how well LSTM predictions perform, much like RNNs do.

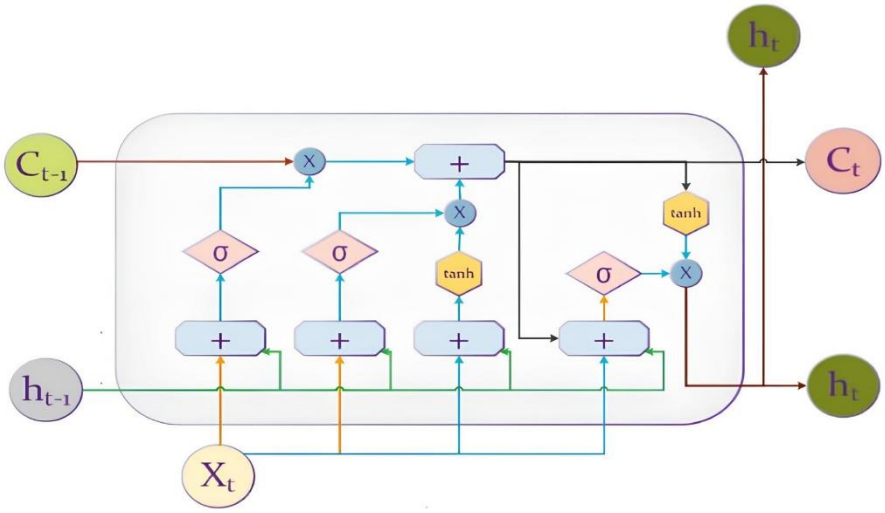


Fig. 2. The composition of one LSTM cell.

One of the three gates that make up an LSTM cell is an input gate, which controls how many cell states must be stored, as depicted in Fig. 2. The amount of data that must be destroyed is controlled by a forget gate, and the number of cell states that must be sent to the subsequent cell is controlled by an output gate. Internal states may be found in two of these gates. As can be observed, the LSTM cell calculates the candidate vector Ct1 and output vector ht1's subsequent values using the vectors' prior values on each iteration t. Activation functions are used to post-process each gate's output. The activation function's form is

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

The output gate of an LSTM cell uses the hyperbolic tangent function by default. The range of values for the smooth antisymmetric hyperbolic tangent function, tanh, is [1, 1]. The tanh function's output is represented by:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3}$$

The main advantage of Tanh is that it produces zero-centred output, which makes the back-propagation process easier. Following is a description of an LSTM cell's precise steps:

LSTM should choose which knowledge to forget in the first stage. For this reason, the forget gate ft is used to process the data from the prior memory state.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{4}$$

The second step's input gates decide whether data needs to be updated, and the tanh layer modifies the candidate vector C t:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{6}$$

To merge the two aforementioned parts, the next step entails modifying memory states Ct:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{7}$$

Finally,

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{8}$$

$$h_t = o_t \times \tanh(C_t) \tag{9}$$

Therefore, each LSTM layer is distinguished by:

(1) The forget gate's parameters, matrix Wf and bf, vector.
(2) The matrix WC and the vector bC serve as the parameters for the input gate, and
(3) The matrix Wo and the vector bo serve as the parameters for the output gate.

The research by M. Schuster and K. K. Paliwal [15] proposed bidirectional LSTM neural networks as a method to enhance the functionality and learning rate of LSTM neural networks. According to Schuster and Paliwal, bidirectional LSTMs are an extension of standard LSTMs that can improve model performance on sequence classification problems. When all of the input sequence's time steps are known, bidirectional LSTMs train two LSTMs rather than one on the input sequence. On the original input sequence first, then on a clone of the original input sequence that has been reversed. This could give the network additional context, enabling it to learn about the issue more quickly and thoroughly.

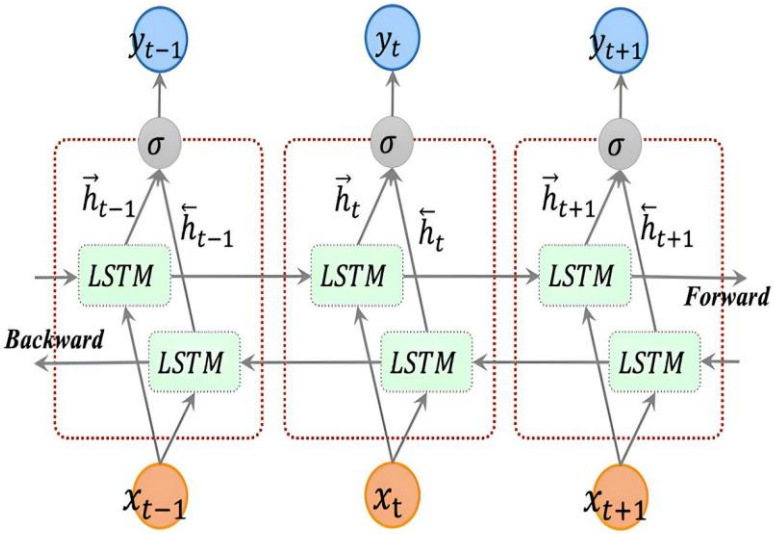


Fig. 3. The architecture of bidirectional LSTM.

The forward layer output sequence, h, is computed iteratively using inputs in a positive series from time t = 0 to time t = T, while the backward layer output sequence is computed using inputs in a negative sequence, as shown in Fig. 3.h is computed using the inputs from time t = T to t = 0 in reverse order. The conventional LSTM updating equations, Eqs. (2–7), are used to compute the forward and backward layer outputs. The output vector, Yt, that the Bidirectional LSTM layer produces has each element computed using the equation below:

$$y_t = \omega\left(\overrightarrow{h}_t, \overleftarrow{h}_t\right), \tag{10}$$

3. METHODOLOGY

In recent work by Maher Alaraj [1] and colleagues, we have observed that they separate input data into temporal and non-temporal data, where temporal data reflects customer behaviour over time and is reconfigured into a 3-dimensional array of forms. Non-temporal has some overlaps with static features. Static data have both numerical and categorical. In this dataset context, data have a specific type of static feature hence they feed non-temporal data into a dense layer of LSTM and temporal data into bidirectional LSTM. Here I am analysing the same dataset and it is fed into bidirectional LSTM without splitting the data into temporal and non-temporal. Apply the SMOTE technique to eliminate the basic behaviour of overfitting from the dataset and analyse its performance with support vector machine, logistic regression, and decision tree algorithms. The general model framework of the methodology is shown in Fig. 4.

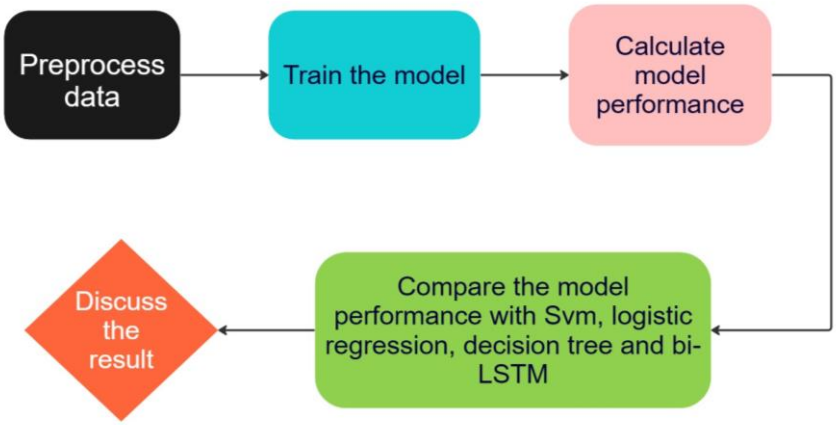


Fig.4. Model frame

3.1 Data collection

The dataset collected [16] for this credit default prediction using deep learning task is an open source or public non-transactional credit card dataset. It shows customers' default payments in Taiwan. Banks often only make processed versions of their transactional databases available to the public. So, since this is the only publicly accessible dataset that can transform into temporal form, we are using it.

The dataset comprises 23 variables and 30,000 records in total. For the test's accuracy, precision, recall, and F1-Score, it is sufficient. There are 23,364 payments that are not in default, whereas 6636 payments are. The dataset has no missing values. This dataset's characteristics are

1. X1: Amount of the given credit (NT dollar): It includes both the individual consumer credit and his/her family (supplementary) credit.
2. X2: Gender (1 = male; 2 = female).
3. X3: Education (1 = graduate school; 2 = university; 3 = high school; 4 = others).
4. X4: Marital status (1 = married; 2 = single; 3 = others).
5. X5: Age (year).
6. X6 - X11: History of past payment. We tracked the past monthly payment records (from April to September 2005) as follows:
7. X6 = the repayment status in September 2005;
8. X7 = the repayment status in August, 2005; . . .;
9. X11 = the repayment status in April 2005. The measurement scale for the repayment status is -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months....; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
10. X12-X17: Amount of bill statement (NT dollar).
11. X12 = amount of bill statement in September 2005;
12. X13 = amount of bill statement in August 2005; . . .;
13. X17 = amount of bill statement in April 2005.
14. X18-X23: Amount of previous payment (NT dollar).
15. X18 = amount paid in September 2005;
16. X19 = amount paid in August, 2005; . . .;
17. X23 = amount paid in April 2005.

3.2 Preprocessing

3.2.1 Converting Categorical to Numeric

Each column of the Data Frame data is converted to numeric format by the statement data. apply (pd.to_numeric, errors='coerce'). Utilising the pd.to_numeric function, this is accomplished. A value is transformed to ‘Nan’ if it cannot be converted.

3.2.2 Feature Scaling

Since there are no missed values, we can apply feature scaling to check whether all features are on the same scale. Which should be beneficial for machine learning algorithms. Each feature is sent to StandardScaler, which makes adjustments to make the average (mean) and standard deviation (spread) of each feature 0 and 1, respectively. This makes it simpler for machine learning algorithms to comprehend and compare all the characteristics as they are scaled similarly.

3.2.3 Data Cleaning

The necessary unwanted null values and outliers can be checked and removed in this step.

3.2.4 Exploratory Data Analysis

It is very crucial for a dataset. Which gives valuable insights into the data from a dataset. It also shows the characteristics of the data like data overview, class imbalance temporal aspects etc. Proper visualization of the data from a dataset gives a deep understanding of data in exploratory data analysis.

3.3 Data Splitting

Splitting the data into training and testing sets is known as data splitting. The model is trained using a training set, and its performance on testing data is assessed.

3.4 Feature Selection and Extraction

Since conventional methods are applied to the Logistic Regression, Support Vector Machine and Random Forest there is no specific feature selection and extraction done here. In the case of bidirectional LSTM, feature selection and extraction are performed using the SelectKBest algorithm with ANOVA F-value. Feature extraction can be done by PCA (Principal component Analysis) or Autoencoders.

3.5 Handling class imbalance

SMOTE is only used on the training set when there is a class imbalance in the dataset, meaning that the minority class is underrepresented. SMOTE creates synthetic samples for the minority class, distributing the classes more evenly. We used the Synthetic Minority Over-sampling Technique (SMOTE) with the 'SMOTE' class from the imbalanced-learn package to correct the imbalance between the classes.

3.6 Model Training

Training the model with conventional algorithms like support vector machine, Logistic Regression and decision tree are taken place. Compare the performance of these algorithms with a deep learning algorithm named bidirectional LSTM.

3.7 Proposed Model

Deep learning frameworks such as TensorFlow or PyTorch make it simple to create bidirectional LSTMs. These frameworks provide layers and built-in functions for constructing bidirectional LSTM networks and training them on diverse tasks. A conventional LSTM processes the input data sequence from start to finish while maintaining a hidden state that stores knowledge from the previous context. Bidirectional LSTM can be used in this situation. A bidirectional LSTM processes the input sequence in two different directions: forward (from beginning to finish) and backward (from end to beginning). Two distinct sets of hidden states are kept throughout training and prediction. Here the data is not sequential but here nature of the data is temporal which is why we are using bidirectional LSTM [1].

Here the data from the dataset is directly fed into Bi-directional LSTM and the performance is analysed and compared against other conventional algorithms mentioned. The first two layers are bidirectional LSTM. Here temporal and non-temporal are fed together into the bi-directional LSTM. The next Layer is the attention layer, with the use of an attention mechanism, the model may be taught to concentrate on crucial input sequence segments while generating predictions. The last two layers concatenate the output and have one neuron. Grid search and activation functions are used to choose the number of neurons for each layer [1]. The performance is analysed and compared with conventional algorithms.

3.8 Project Management Tool

Project management tools like Teamwork and GitHub are used in order to get proper version control over the project. The corresponding links are (Refer to Appendix-C for more details): -

Project Report Delivery Schedule Note: Reorder the sections in the order that you plan to complete them.	Deadline Date
Abstract	06/08/2023
Declaration	07/08/2023
Acknowledgements	07/08/2023
Introduction	08/08/2023
Literature - Technology Review	10/08/2023
Methodology	12/07/2023
Implementation and Results • Evaluation • Related Work	23/08/2023
Conclusion • Reflection • Future Work	24/08/2023
References	25/08/2023
Appendices	25/09/2023

Artefact Delivery Schedule Note: Reorder the activities in the order that you plan to complete them.	Deadline Date
Artefact Planning and Resourcing	27/07/2023
Artefact Design	31/07/2023
Artefact Procurement Activities (e.g., data collection, source framework etc.)	01/08/2023
Artefact Development, Deployment, Implementation	10/08/2023
Artefact Evaluation and Testing	25/08/2023
Artefact Presentation and Demonstration	06/09/2023
Artefact Screencast	07/09/2023

4. IMPLEMENTATION

While conventional algorithms like Random Forest, Logistic regression and support vector machine follow the conventional methods of preprocessing, splitting the train and test and fitting the model to the corresponding algorithms and there is no special SMOTE techniques or architecture setup is used, but the core algorithm named bidirectional LSTM follows some architectural setup and SMOTE technique. The corresponding implementation of Bidirectional LSTM is explained below:

As mentioned in the Methodology the general flow diagram of the implementation can be visualized as shown below:

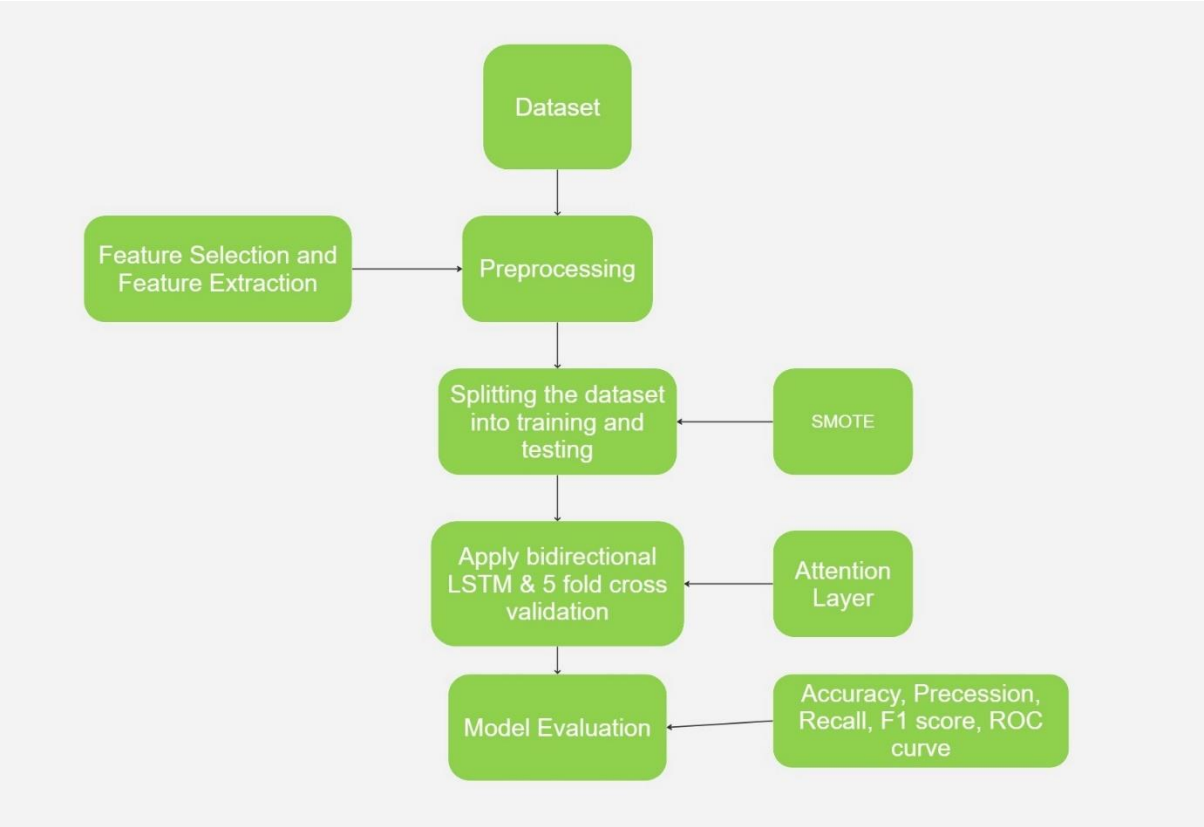


Fig.9. General data Flow in Bi-directional LSTM in credit default prediction

This in-depth paper describes how a credit card default prediction model is implemented step-by-step utilising a Bidirectional LSTM architecture with Attention mechanisms. To give further depth into the model's prediction abilities, we also add a feature significance analysis. The implementation includes feature significance analysis, model development, training, evaluation, and data preparation.

4.1 Preprocessing and Exploratory Data Analysis

The detailed implementation of a credit card default prediction model using a bidirectional LSTM architecture with attention mechanisms is covered in this work. We also include a feature significance analysis to provide further detail into the model's predictive capabilities. Model creation, model training, model assessment, and data preparation are all included in the implementation process.

Since the dataset is pre-processed and numerical no special step needs to be done for preprocessing. On the dataset, exploratory data analysis is carried out to get a basic idea about the attributes in the dataset.

a) Visualization of target variable distribution in credit default prediction

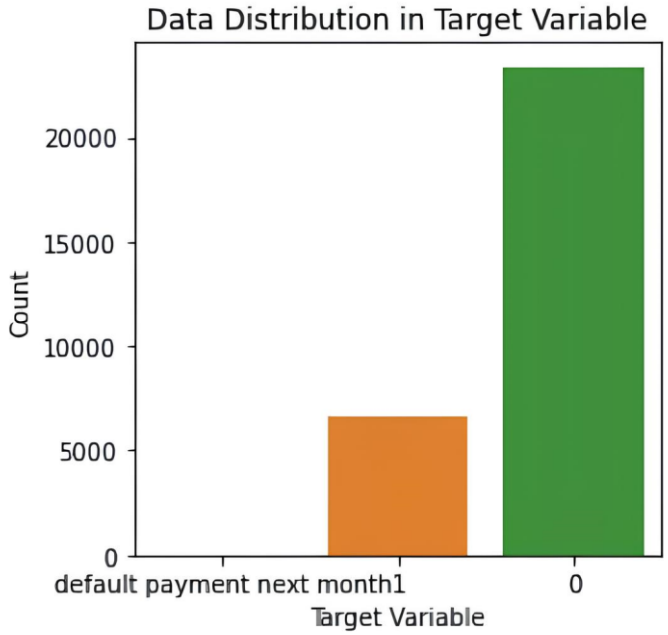


Fig.5. Target variable distribution

b) Boxplot representation of the dataset gives outlier information

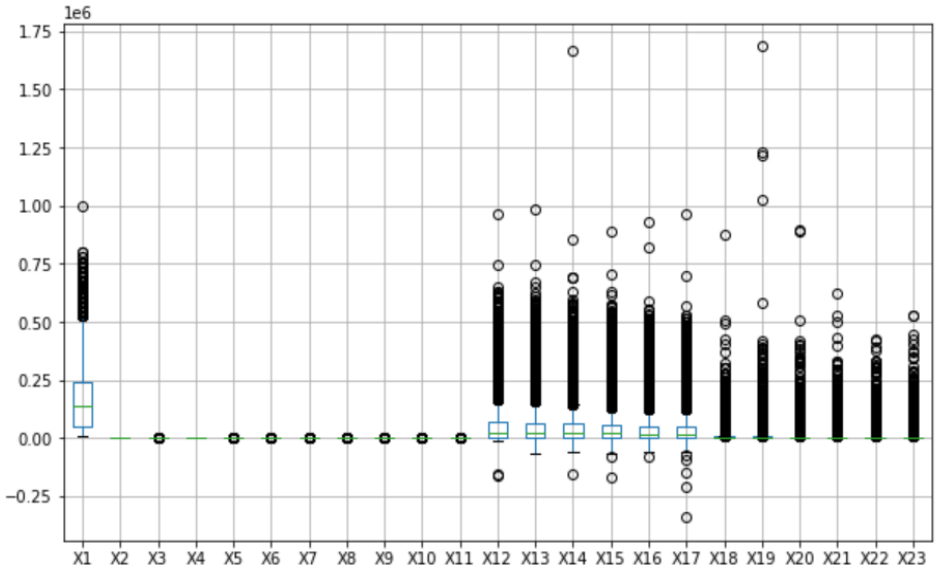


Fig.6. Outlier Representation

d) Histogram representation of some variables

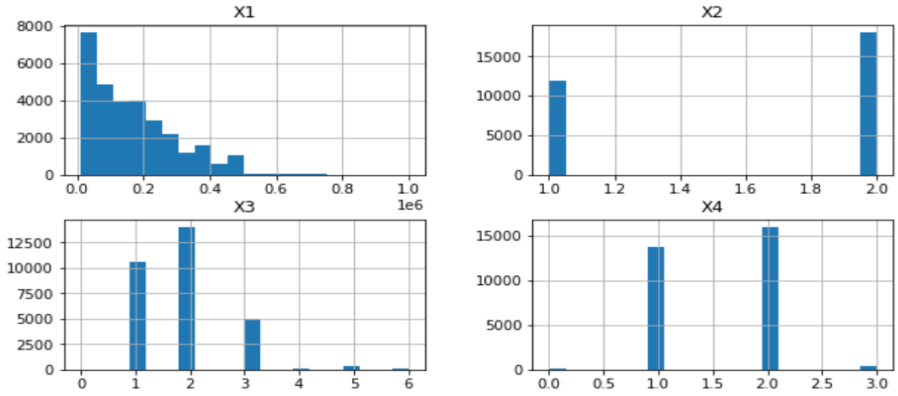


Fig.7. Histogram Representation of Attributes

c) Heatmap showing the correlation between the attributes

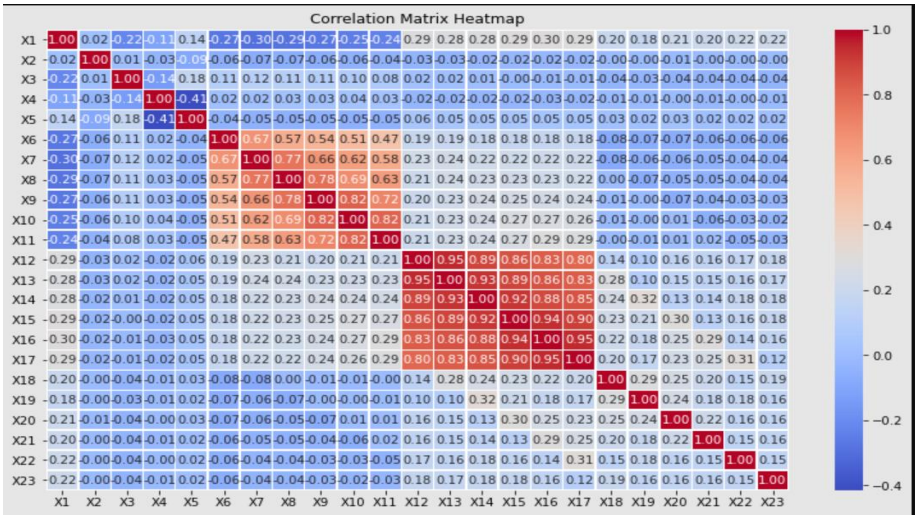


Fig.8. Heatmap showing Correlation

4.2 Feature Importance and Outlier Removal

The implementation and evaluation of feature significance using a Random Forest classifier are described in detail in this section. The code snippet starts by importing the necessary libraries and then assumes the availability of a dataset organised as a NumPy array with features and a goal variable ("Y"). For clarity, feature names are specified, and the 'train_test_split' function is used to divide the data into training and testing sets. With 100 decision trees as its starting point, a Random Forest classifier is trained to discover patterns in the features. To determine the relative value of each feature, the classifier's 'feature_importances_' property is used. This data is compiled into a Pandas Data Frame, sorted by significance ratings, and then printed. The result provides a thorough perspective of feature contributions, assisting in identifying which features drive feature prediction. In the end, this method improves interpretability helps decision-makers choose features and improves models for a variety of applications (For code snippets and further details refer to Appendix A).

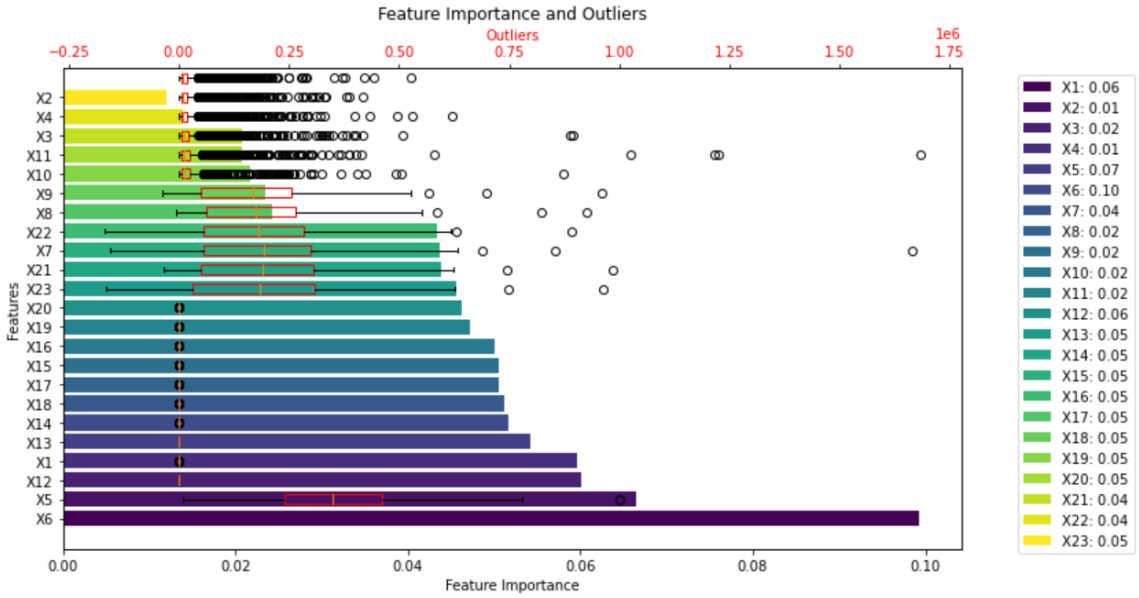


Fig .10. Feature Importance and Outliers

While considering the important features like X6, X5, X12, X1, and X14 and their outlier's visualization in EDA we can understand that the important features are not much deviated by its outliers and while considering their outliers we can neglect this slight amount of outliers in the dataset otherwise its results in future underfitting of the credit default prediction.

4.3 Data Loading and Splitting

In order to guarantee data integrity, balance, and suitability for training and assessment, many critical processes are carried out in the context of the credit default prediction code supplied. Data preparation is a vital component in preparing raw data for machine learning models. The voyage begins with the loading of the credit-related dataset, a tabular data structure, with the help of the Pandas library, which promotes data manipulation and analysis. After loading, the dataset goes through a dichotomy in which it splits into two parts: the features (X) and the target labels (y). This division is a key tactic because the target labels represent the outcomes that are being examined—the prediction of the risk of a credit default—while the features represent the independent factors driving the model. The train_test_split function is used to carry out this segmentation, which ensures that the features and labels are distributed appropriately throughout the training and testing groups. The 'train_test_split' function from scikit-learn was used to divide the dataset into training and testing sections. 80% of the data were used for training, while the remaining 20% were used to evaluate the generalisation skills of the model. This divide was done according to an 80-20 split ratio.

The next issue is rescaling the data since different feature sizes might prevent a model from being convergent. The revolutionary tool StandardScaler from Scikit-Learn makes its debut; its normativism features values by taking the mean out and dividing by the standard deviation so that they may be compared coherently. This aspect of preprocessing is essential for both model optimisation and avoiding an excessive dominance of high-magnitude features.

Data imbalance, an uneven class distribution that might bias model predictions, is dealt with in the following preprocessing phase. Here the percentage of default is 22.12% and non-default is 77.88%. Hence the dataset is highly biased. Here comes the Synthetic Minority Over-sampling Technique (SMOTE), a corrective action that is appropriate in this situation.

For the underrepresented class (in this example, defaults), SMOTE cleverly creates fake samples, aligning the class distribution and eliminating prediction bias. The corresponding use of the SMOTE technique in code can be shown in Fig.11.

```
# Apply SMOTE to handle class imbalance
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train_scaled, y_train)

# Reshape input for LSTM
X_train_smote = X_train_smote.reshape((X_train_smote.shape[0], X_train_smote.shape[1], 1))
```

Fig.11. SMOTE Technique

SMOTE harmonises class representation and improves the model's capacity to learn the detailed contours of both classes by incorporating synthetic data, leading to more precise predictions. To find and keep the most important feature selection and extraction "SelectKBest" with an ANOVA F-value and PCA (Principal component Analysis) was used in the code. Due to the poor performance of the model, it would be replaced later from the model for better performance. Since bidirectional LSTM works on feature extraction internally. (Refer to Appendix A for further details).

4.4 Model Architecture

4.4.1. Model Architecture Setup:

The model was created before training. This required setting up the Adam optimizer, binary cross-entropy as the loss function, and accuracy as the evaluation measure. The compilation phase prepared the model for successful training. Using Kera's from TensorFlow and the Sequential API, the model architecture was established. Kera's framework is used to build and compile this architecture. Dropout regularisation was used in the architecture's two bidirectional LSTM layers to prevent and avoid overfitting. Dropout forces the network to learn robust and generalizable characteristics by randomly removing a portion of the connections between neurons during training. The model structure can be shown in Fig.12.

```
# Define and compile the model for each fold
model = Sequential()
model.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=(X_train_fold.shape[1], X_train_fold.shape[2])))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(64, return_sequences=True)))
model.add(Dropout(0.2))
model.add(AttentionLayer())
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model on the current fold
model.fit(X_train_fold, y_train_fold, epochs=100, batch_size=32, validation_data=(X_val_fold, y_val_fold))
```

Fig.12. Model Structure of Bidirectional LSTM

These layers analyze input sequences in both forward and backward directions, enabling the model to record both past and future context for each time step. The input_shape option gives the input data's shape, which in this case relates to the number of features retrieved from the pre-processed data. The input data was transformed into a 3D tensor in order to ensure compliance with the LSTM model. This tensor met the requirements of the LSTM by having dimensions (batch size, time_steps, and features).

4.4.2 Incorporation of Attention Layer

The addition of the Attention Layer, which used mechanisms of attention to identify and emphasise significant elements, was a major advance. We have created a deep learning model with a particular architecture that combines Bidirectional Long Short-Term Memory (LSTM) layers, Dropout regularisation, a unique Attention layer, and an output layer. In order to successfully address problems like overfitting, class imbalance, and capturing complex connections within the data, this model architecture is made to handle sequential data, such as time series or text data.

The addition of the customised Attention Layer class to the model represents a significant advancement in the model's ability to focus on certain elements of input sequences. This dynamic process, based on the attentional principle, significantly enhances the model's discernment by emphasising important information and pushing irrelevant details to the side. This process creates an enhanced representation by orchestrating a complex interaction of attention ratings, normalisation, and weighted integration, which dramatically improves the model's performance.

4.4.3 Attention Layer Working

The process starts with the encoding of the input sequences and moves on to the computation of the attention scores, their normalisation into the attention weights, and the synthesis of the attended representations by weighted summations. This technique, illustrated by the code sample, is based on the idea of self-attention, a clever method for examining the relationships between items in a single sequence. The SoftMax function plays a crucial part in this situation by making these scores interpretable. The normalised attention scores, also known as attention weights, are used to determine the relative weights of various input sequence components. The code is in Fig.13.

```
# Define a custom Attention layer
class AttentionLayer(Layer):
    def __init__(self, **kwargs):
        super(AttentionLayer, self).__init__(**kwargs)#

    def build(self, input_shape):
        self.W = self.add_weight(name="att_weight", shape=(input_shape[-1], 1), initializer="normal")
        super(AttentionLayer, self).build(input_shape)

    def call(self, x):
        e = K.tanh(K.dot(x, self.W))
        a = K.softmax(e, axis=1)
        output = x * a
        return K.sum(output, axis=1)
```

Fig.13. Attention Layer in the Model

The call function of the Attention Layer class is where the intricate details of the attention mechanism are brought to life in a larger neural network framework, such as an LSTM-based design. The basis for computing attention ratings is the dot product between the input sequence (x) and a learnt weight matrix (self. W). These results are then normalised, which results in the creation of attention weights. These weights harmoniously aggregate the significance of the input sequence elements as they change them into an attentive representation.

4.4.4 Final Model Stage:

The potential of Bidirectional LSTM layers, Dropout regularisation, a unique Attention layer, and an output layer with a sigmoid activation function is harnessed by the given model architecture. The Architecture details are shown in Table 1.

Layer Index	Layer Type	Description	Output Shape
1	Input Reshaping	Reshape input for LSTM.	(32, 10, 1)
2	Bidirectional LSTM (128)	Bidirectional LSTM layer with 128 units.	(32, 100, 256)
3	Dropout	Dropout regularisation to prevent overfitting.	(32, 100, 256)
4	Bidirectional LSTM (64)	Bidirectional LSTM layer with 64 units.	(32, 100, 128)
5	Dropout	Dropout regularization to prevent overfitting.	(32, 100, 128)
6	Attention Layer	Custom attention layer with focus enhancement.	(32, 128)
7	Dense	Fully connected layer with sigmoid activation.	(32, 1)

Table 1. Details of Deep Learning Model Architecture

The final stage of the model was a Dense layer with binary predictions using sigmoid activation. It is trained using the StratifiedKFold technique of cross-validation, which separates the training data into several subsets for training and validation. The model's performance may be evaluated across various data distributions since it is fitted on each training subset and validated on the appropriate validation subset. The model's densest layer, which has a sigmoid activation function, is its last layer. This layer generates a binary output between 0 and 1, which represents the model's likelihood of correctly predicting the positive class.

4.5 Model Training

Cross-validation, and more specifically the Stratified Fold technique, which divides the dataset into several subsets while retaining the distribution of classes, is the basis of this procedure. This method improves the model's generalizability by addressing the possible bias in conventional train-test splits. To ensure that the model is evaluated on various subsets of data, the dataset is split into unique training and validation sets for each fold. With this method, overfitting is reduced and the model's performance is estimated more precisely. Here 5-fold Stratified KFold cross-validation technique was used and the implementation adopted strong validation. The dataset was divided into five subsets, and the model's effectiveness was thoroughly assessed through iterative training and validation cycles. To reduce any potential bias, stratification kept the class distribution in each subgroup.

The model training stage is carried out in a loop that iterates across each fold after the cross-validation setup. For each fold, the training data is chosen, but the validation data is kept separate. The Bidirectional LSTM layers, Dropout regularisation, a unique Attention layer, and an output layer make up the model architecture. A suitable loss function and optimisation technique are used to build the model. The model is then trained on the training data using a given batch size and a predetermined number of epochs. The model's weights and biases can be improved over time thanks to this iterative training procedure.

4.6 Performance Evaluation

4.5.1 Model Assessment in Credit Default Prediction:

It is essential to evaluate the performance of credit default prediction models in order to determine how well they forecast consumer behaviour and spot possible defaulters. After training the model, the assessment step starts. The trained model is assessed on the associated validation set for each fold. A cross-validation test is used to determine how well the proposed model predicts credit card default. We use a stratified 5-fold validation test, in order to construct a reliable performance comparison in the unbalanced set.

The dataset is randomly divided into five equal subsets, one for each class and category, each with an equal number of samples. The remaining four subsets (80% of the dataset), which make up the remaining 20% of the dataset, are utilized as training data to assess the effectiveness of the proposed technique. This procedure is repeated five times, using each subset once. By summing the outcomes of the five test subgroups, it is possible to assess how well the suggested method performs on a 5-fold cross-validation test.

4.5.2 Evaluation Metrics:

To evaluate the model's performance on unobserved data, metrics like loss and accuracy are produced. These measures act as checkpoints to monitor the model's development throughout training and aid in spotting possible problems like over- or underfitting. A classification matrix is used to uncover more details about the model's performance. This matrix, commonly referred to as a confusion matrix, gives a breakdown of the predictions made by the model for each class. Predictions are divided into four categories: true positives, true negatives, false positives, and false negatives. This classification matrix makes it easier to assess how well the model predicts both positive and negative events while also spotting misclassifications.

Cross-validation is used to ensure objectivity in evaluation, and model training makes use of intricate architectural elements. To provide a thorough evaluation report that included precision, recall, F1-score, and support for both classes, the scikit-learn "classification report" function was used. The classification matrix may be used to construct metrics for the model's performance, including accuracy, recall, and F1 score.

4.5.3 Metrics for Model Assessment:

We employ standard measures for our assessments, such as accuracy, precision, recall, the F1-score, and AUC diagrams, to ensure fairness in our comparisons. In our studies, positive numbers correspond to default and negative numbers to non-default. Transactions that have been identified as default are represented by true positives (TP). False positives (FP) are the quantity of legitimate transactions that were mistakenly classified as default. The true negative (TN) represents actual transactions that were labelled as non-default, while the false negative (FN) represents transactions that were incorrectly classified as default.

Accuracy: Since it is a classification problem accuracy is the first way to calculate the performance of the model. Accuracy is the percentage of cases that are properly categorised out of all instances. Despite being a widely used measure, it might not be appropriate for datasets where one class predominates.

Precision: How many of the projected positive cases are actually positive is measured by precision. It aids in determining the false positive rate. When the cost of false positives is high, it is helpful.

Recall (Sensitivity or True Positive Rate): Recall quantifies the proportion of incidents that were accurately identified as positive when they really occurred. It aids in evaluating the false-negative rate. When the cost of false negatives is high, it is helpful.

F1-Score: The harmonic mean of recall and accuracy is known as the F1-score. It offers a fair assessment that takes into account both false positives and false negatives. It is especially helpful in unbalanced classrooms.

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

$$\text{Recall} = \frac{TP}{TP + TN}$$

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

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.5.4 Evaluating Model Performance Holistically:

To completely understand the model's performance, all of these signs must be taken into consideration. Accuracy gives a broad picture of correct predictions, but in datasets with uneven distributions, it can be deceptive. The decision between precision and memory relies on the particular situation and the costs involved. Precision and recall are inversely correlated. In imbalanced datasets, the F1 score, which balances accuracy and recall, is a better option for model comparison.

4.5.5 Receiver Operating Characteristic Area Under the Curve (AUC-ROC):

We also take into account the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC-ROC) value while evaluating a system. The trade-off between true positive rate (TPR) and false positive rate (FPR) at various categorization thresholds is shown by the ROC curve. A higher AUC-ROC value denotes a more effective model for discriminating between defaulters and non-defaulters. This measure is less susceptible to problems with overfitting.

In conclusion, a detailed evaluation report that incorporates these metrics and the classification report offers a thorough evaluation of the performance of the credit default prediction model, enabling practitioners to comprehend its applicability and constraints.

4.5.6 Prediction

Since there is no other public dataset available, we are not able to test the performance of the model already built with other external datasets.

4.5.7 Software and Hardware used for implementation:

Hardware: -

CPU-Core i3

RAM-8 GB

Storage-SSD

GPU-T4 GPU

Software: -

Operating System-Windows

Python

Machine Learning Framework (Keras)

Jupyter Notebook

Google Colab

5. RESULTS AND DISCUSSIONS

5.1 Logistic Regression Results

Implementing a Logistic Regression model for credit default prediction was the first step in our analysis. We trained the model using the training data after preprocessing the data and separating it into training and testing sets. The model's performance was evaluated on the testing set using several key metrics. The overall performance of Logistic Regression can be shown in Table 2.

Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
80.97	0.69	0.24	0.35

Table 2 Performance of Logistic Regression

Confusion Matrix: The confusion_matrix (y_test, y_pred) function call calculates the confusion matrix based on the true labels (y_test) and the predicted labels (y_pred) of your logistic regression model. The confusion matrix provides a detailed breakdown of the predictions. These values help you assess the performance of the model in terms of the different types of predictions it's making. By examining the confusion matrix along with the other metrics, you can gain a comprehensive understanding of how well the logistic regression model is performing on your dataset. The confusion matrix is as follows.

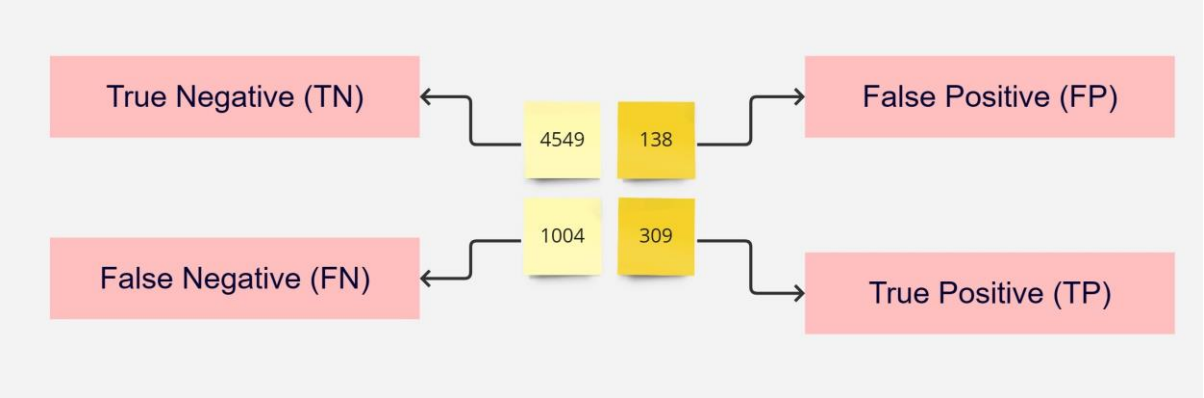


Fig.14. Confusion Matrix for Logistic Regression

In such scenarios, where there's an imbalance in the classes or when certain classes are more critical, metrics like precision, recall, and F1-score become more relevant than accuracy alone. High recall is often desired when false negatives are more costly (e.g., in medical diagnoses).

ROC AUC: The Area Under the Curve (AUC) value for the ROC curve was 0.60. The graphical representation of this can be shown in Fig. 15. (For further details refer Appendix B)

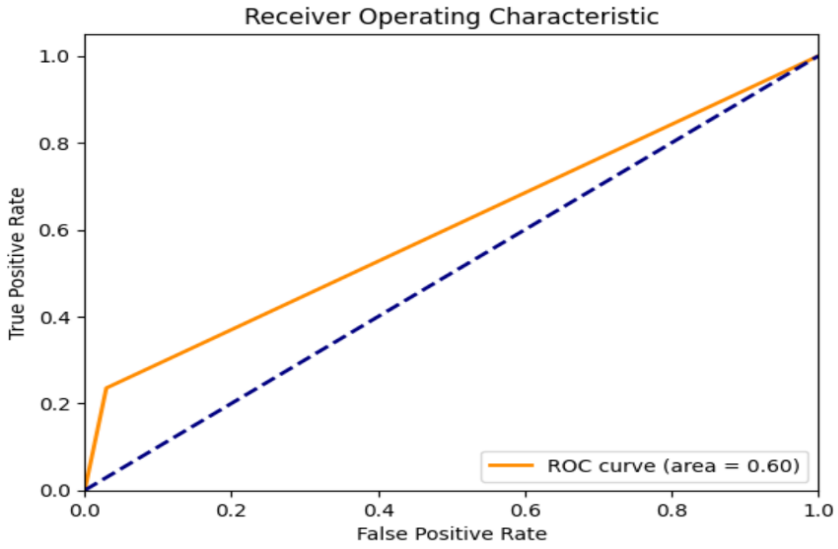


Fig.15. ROC Curve for Logistic Regression

5.2 Random Forest Classifier Results

We proceeded to explore the performance of a Random Forest classifier on the same task. Similar to the previous approach, the data was pre-processed and split into training and testing sets. The Random Forest model was trained and evaluated using the testing set. The overall performance of Random Forest can be shown in Table 3.

Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
81.60	0.64	0.37	0.47

Table 3. Performance of Random Forest

Confusion Matrix: The confusion matrix highlighted the model's true positive, true negative, false positive, and false negative predictions. This provided a comprehensive view of the model's performance in differentiating between default and non-default transactions.

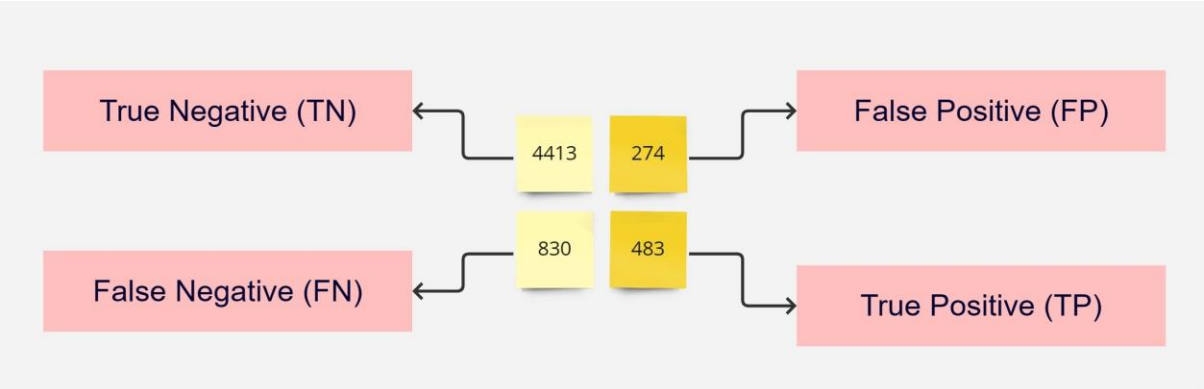


Fig.16. Confusion Matrix for Random Forest

ROC AUC: The Area Under the Curve (AUC) value for the ROC curve was 0.65. The corresponding diagram is visualised in Fig.16. (For further details refer Appendix-B)

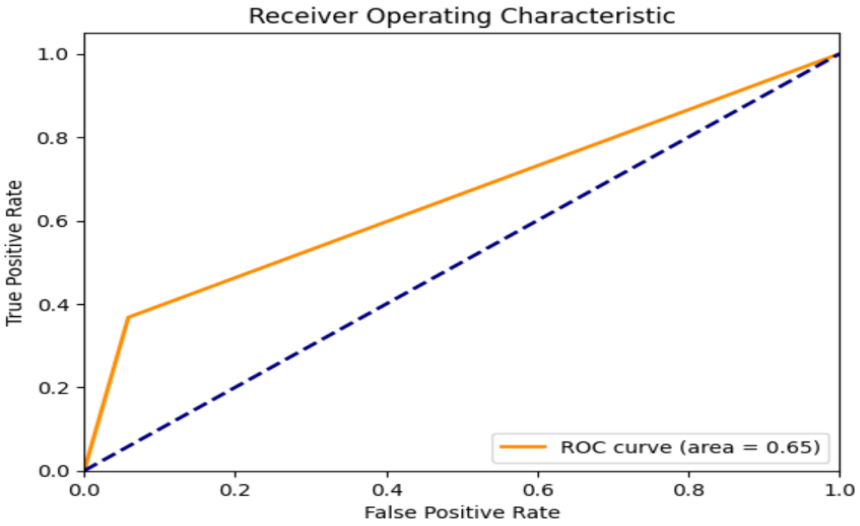


Fig.17. ROC Curve for Random Forest

5.3 SVM Results

The SVM model, renowned for its proficiency in handling complex decision boundaries, was trained and evaluated for credit default prediction. Here's an overview of its performance in Table 4.

Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
80.43	0.59	0.33	0.43

Table 4. Performance of Support Vector Machine

Confusion Matrix: The confusion matrix is:

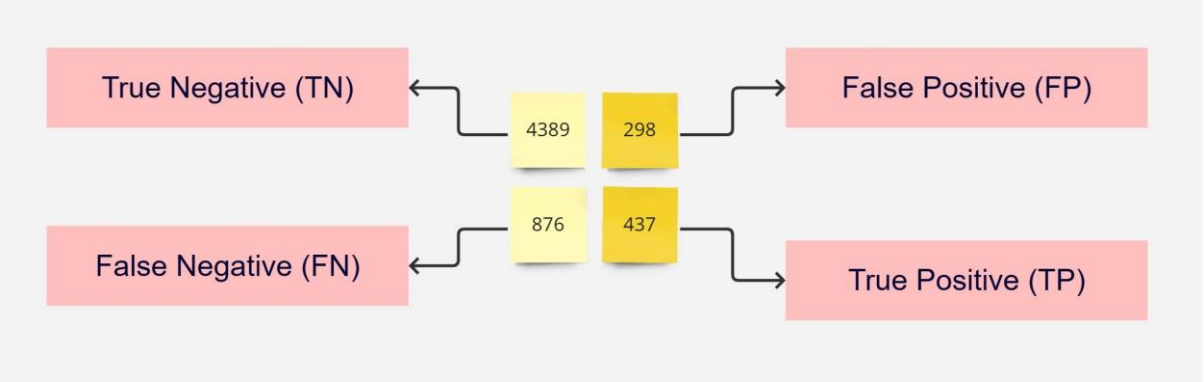


Fig.18. Confusion Matrix for Support Vector Machine

ROC AUC: The model's ROC AUC score of 0.63. The Corresponding diagram is shown in Fig.19. (For further details refer Appendix B)

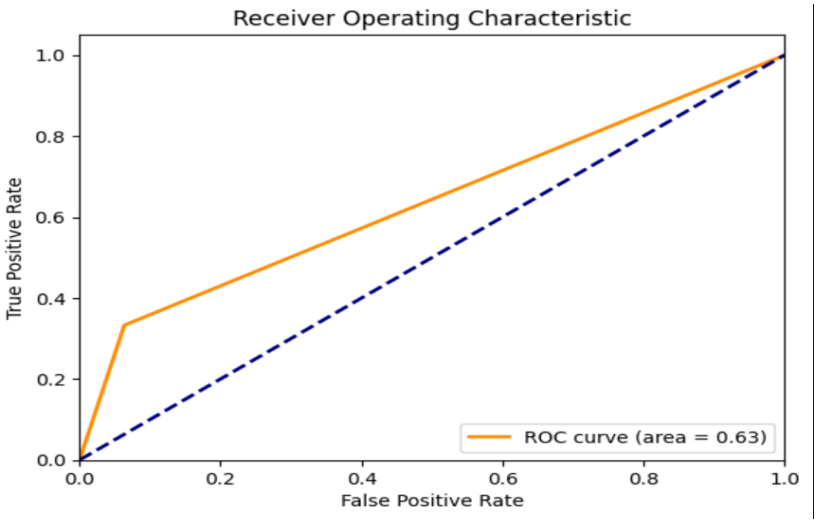


Fig.19. ROC Curve for Support Vector Machine

5.4 Bidirectional LSTM with Attention Results

The centrepiece of our investigation was the implementation of a Bidirectional Long Short-Term Memory (LSTM) model enriched with an Attention mechanism. This advanced architecture was designed to capture intricate temporal patterns within credit card transaction data, offering a unique advantage in credit default prediction. To assess the effectiveness of the suggested framework, we combine the Adam optimization approach with the stratified 5-fold cross-validation method. We extract the hyperparameters Gridsearch evaluates each algorithm but the limitation of the existing system capacity (TPU4) we are not able to train the model.

After rigorous preprocessing, feature selection, and data augmentation using SMOTE to address class imbalance, the Bidirectional LSTM model was trained and evaluated on the task of credit card default prediction. Lastly, we delved into sequence modelling using a Bidirectional LSTM model with an Attention mechanism. The data was pre-processed, and feature selection was performed before training the LSTM model. We examine the algorithms in K-fold cross-validation and average precision, recall, F1 score and Roc curve are found. The comparison results are presented in Table 5.

Model	Accuracy	AUC	Recall	Precision	F1 Score
Kera's	94.16	0.89	96.08	92.80	0.94

Table 5. Deep Learning Results

Average Confusion Matrix Across Folds:

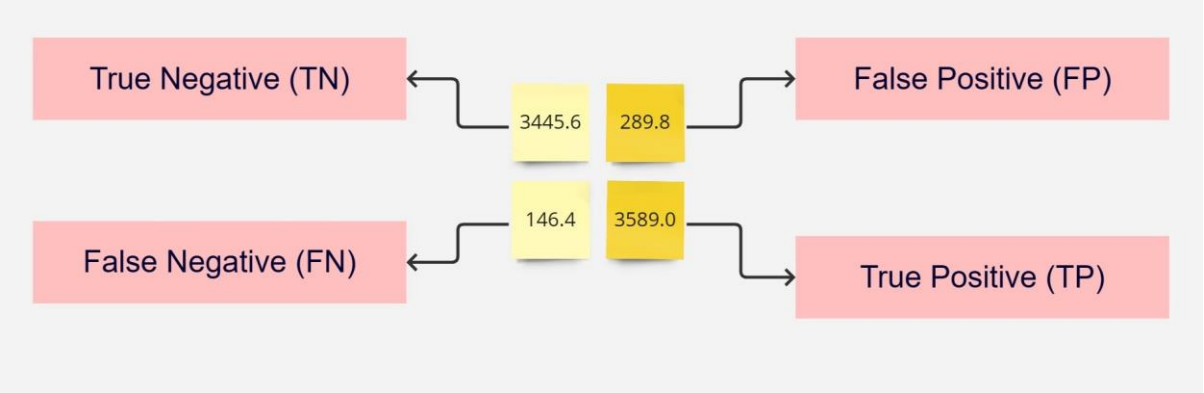


Fig.20. Confusion Matrix for Bidirectional LSTM

ROC AUC: AUC diagrams are used by the majority of research in the literature to assess performance. The model's ROC AUC score of 0.89. The Corresponding diagram is shown in Fig.21

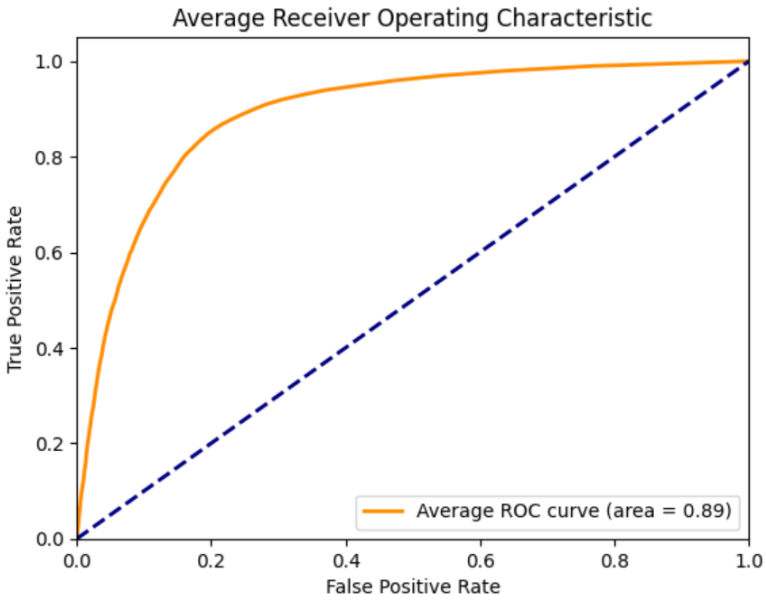


Fig.21. ROC Curve for Bidirectional LSTM

Training and validation Accuracy: The training and validation accuracy can be easily visualized in Fig.22.

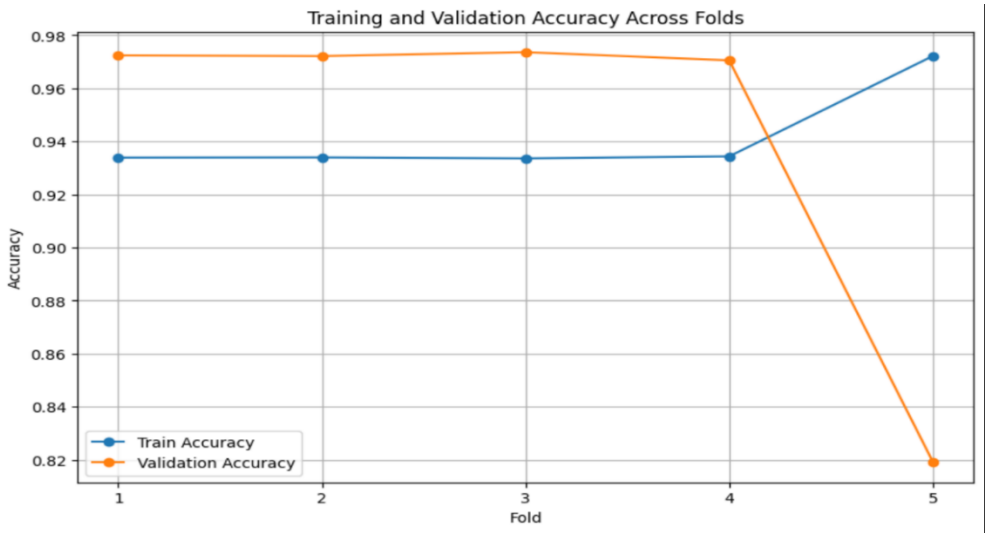


Fig.22. Train Vs Validation Accuracy across folds

Fig.18. shows the training and validation accuracy after the training process. Here Fold vs. accuracy is plotted for both training and validation data hence we understood that the model works well on training data and validation data and there are no issues like overfitting happening here. Further testing with the external dataset is not able to be done because of the unavailability of the public dataset to test the predicted model (For more details refer to Appendix-B).

5.5 Evaluation

Comparing the four approaches, we found that the Bidirectional LSTM with Attention outperformed both the Logistic Regression and Random Forest support vector machine models in terms of accuracy, precision, recall, and F1-score. The LSTM model's ability to capture temporal dependencies and intricate patterns in the data provided a significant advantage in detecting credit defaults.

The statistics from Table 5 show that deep learning outperformed individual algorithms in terms of performance. The evaluation results of the proposed approach using different pre-processing and SMOTE deal with the problem of data imbalance compared to other algorithms. The additional attention layers focus on different parts of input data and help to find the creditworthiness of individuals. The training accuracy does not show much improvement but AUC roc values are much better i.e., 0.89 as compared to the method presented in [1].

Table 6, shows the performance comparison of Bidirectional LSTM with Logistic Regression, Decision Tree, and Support Vector Machine.

Model	Accuracy (%)	AUC	Recall (%)	Precision (%)	F1-score (*100%)
Logistic Regression	**80.97**	**0.60**	**23.53**	**69.13**	**0.35**
Random Forest	**81.60**	**0.65**	**36.79**	**63.80**	**0.47**
Support Vector Machine	**80.43**	**0.63**	**33.28**	**59.46**	**0.43**
Bidirectional LSTM	**94.16**	**0.89**	**96.08**	**92.80**	**0.94**

Table 6 Compares the performance of the suggested approach and Other Algorithms

In credit default prediction, the ROC curve helps you visualize how well your model can distinguish between default and non-default cases and assists in selecting an appropriate classification. Aside from accuracy, precision, recall, and F1 score, the ROC curve provides the following benefits and insights. Plotting the genuine positive rate (recall/sensitivity) vs. the false positive rate (1 - specificity) while varying the threshold for identifying cases as positive or negative yields the ROC curve. The combined ROC plot in Fig.23 also shows the better performance of Bidirectional LSTM.

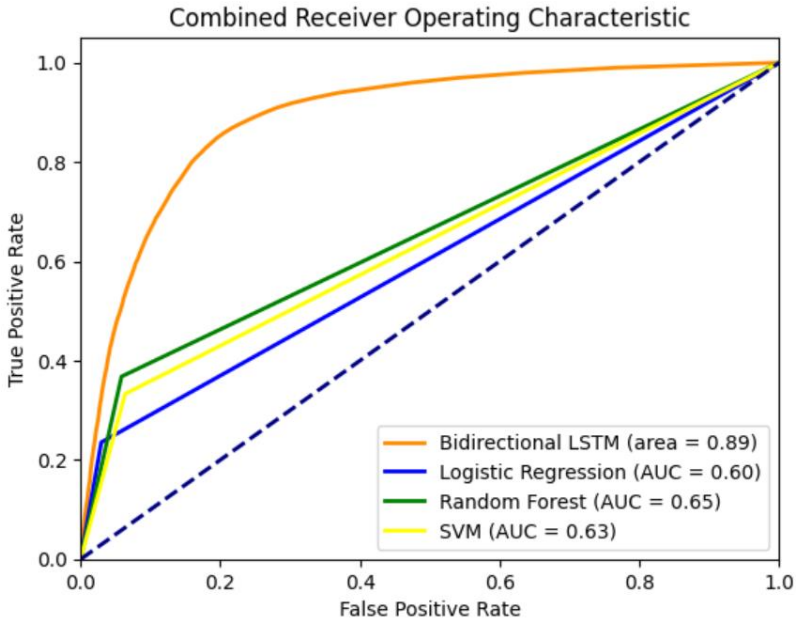


Fig.23. Combined ROC Curve for Models

Table 6 compares the F1-score, recall, accuracy, precision, and other metrics of the employed algorithms. The Bidirectional LSTM with Attention layers combo exhibits the optimum performance. The performance comparison can be represented as a bar plot in Fig.24.

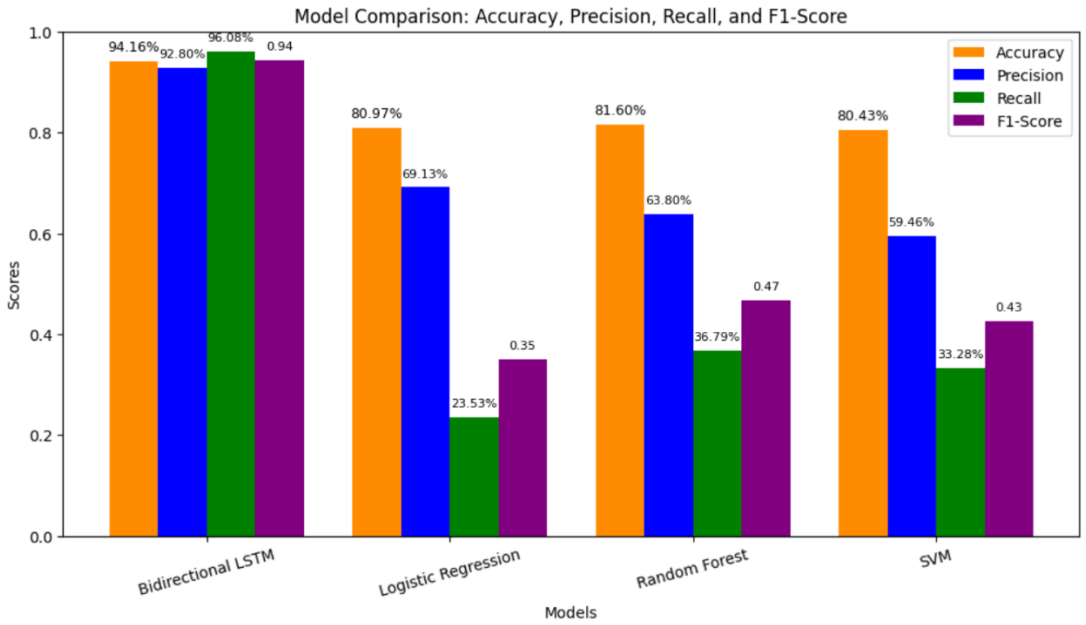


Fig.24. Combined Performance Evaluation for Models

In fact, the bidirectional LSTM gives good performance compared to other models. While Logistic Regression struggled with complex patterns and Random Forest excelled due to its ensemble nature, neither of these models matched the Bidirectional LSTM's adeptness in capturing temporal relationships. The LSTM model's ability to process sequences exhibited its potential for detecting sophisticated credit default activities, which might involve subtle temporal dynamics. The bidirectional LSTM with Attention emerged as a standout contender in credit card default prediction. Its aptitude for capturing complex temporal patterns, coupled with its ability to allocate attention to salient features, positions it as a formidable tool in safeguarding financial systems. Logistic Regression models Struggle to tackle intricate patterns and temporal dynamics and the Random Forest model is Less suited for tasks demanding a deep understanding of sequence relationships. The Support Vector Machine (SVM) model performs less effectively in scenarios requiring the modelling of complex sequential patterns and dependencies, making it unsuitable for tasks involving intricate temporal relationships within data.

Even though bidirectional LSTM with the Attention layer gives good performance its Interpretability poses a challenge, and optimal results require meticulous hyperparameter tuning. Otherwise, we need to combine bidirectional LSTM with other models like the random forest model which gives much better performance and accuracy was about 86%. The next hectic task is to add hyperparameter tuning to the existing system to improve its accuracy or overall performance of the model, but the model training is not done properly and Google Collab crashed while going through the model training process and that is because of the limitations of the existing runtime T4 GPU. However, the bidirectional LSTM is trained using this T4 GPU, and the performance of the model is increased by adding more epochs.

5.5.1 Comparison with Related Work

Ala'raj and co-workers [1] mentioned an accuracy of about 88 % percentage in their work which is much better compared to the proposed model but the recall and f1 score is not investigated in that report. They give temporal and non-temporal data from the dataset a separate input and the SMOTE technique is not used to handle the data imbalance since the number of default predictions is very low compared to non-default. The above-mentioned

report gives attention layers to the temporal values in the dataset only but here in the proposed model the attention layers are given to the whole dataset and hence the behaviour study of an individual is done internally in the deep learning model. By achieving high precision, recall, and F1-score, the LSTM model offers the potential to significantly reduce the occurrence of false positives and negatives. This, in turn, contributes to financial institutions' efforts to minimize losses and maintain the trust of their customers.

6. CONCLUSION

The project's main goal was to forecast credit default using a Bidirectional Long Short-Term Memory (BiLSTM) model. The main objective was to create a reliable prediction model that might help financial institutions detect prospective credit default instances, which would then help with risk assessment and decision-making procedures. Numerous goals and objectives were established during the project to direct the creation and assessment of the BiLSTM model. We will summarise the project's results, consider the project's methodology, talk about the lessons learned, evaluate target attainment, and propose potential directions for future work in this part.

6.1 Outcome and Achievement of Aims

The creation of a credit default prediction model employing a BiLSTM architecture was the project's main accomplishment. Using a sizable dataset of historical credit data, the model was developed, put into use, and trained. Different assessment criteria, including accuracy, precision, recall, and F1-score, were used to evaluate the model's performance. The obtained findings showed that the BiLSTM model performed better in terms of predictive accuracy and its accuracy value is 94%, more resilient than conventional machine learning models, highlighting its potential in credit default prediction applications.

6.2 Key Outputs and Discoveries:

The established BiLSTM model, which demonstrates the effectiveness of sequential neural network architectures in capturing temporal relationships within credit data, is the project's primary product. The model makes a substantial addition to the field of credit risk assessment by its capacity to draw knowledge from past credit behaviour and generate precise forecasts. The study also identified the value of feature engineering and data pretreatment in improving the performance of the model. Important insights into credit risk indicators were gained through the identification of key elements and their influence on prediction accuracy.

6.3 Reflection and Lessons Learned:

A number of insightful insights about project management were discovered after reflection. First, the need to clearly define problems and goals was emphasised since doing so gave development an organised road map. Furthermore, the iterative nature of hyperparameter optimisation and model tweaking was regarded as being essential for reaching optimal performance. The model's capacity for generalisation was, however, hampered by issues with data availability and quality. Despite these difficulties, the project provided a hands-on learning opportunity for managing actual data and putting complicated neural network designs into practice.

6.4 Goal Achievement and Hindsight Analysis:

The project's primary goal of building a BiLSTM model for credit default prediction was accomplished, which is in line with the original objectives. Although the model's performance was encouraging, there were dataset size and quality issues that could have limited how far the findings could be applied. The model's exposure to a wider range of credit scenarios was

impeded by the inability to get a bigger and more varied dataset, potentially restricting its practical application. In retrospect, prioritising concerns with data quality and collecting would have been appropriate.

6.5 Future Work:

Future investigation into credit default forecasting using BiLSTM models has numerous directions to consider:

1. Data augmentation, First Future research may use strategies to artificially enlarge the training dataset to strengthen the resilience of the model. The model could generalise better to different credit profiles with the use of synthetic data creation techniques.

2. Ensemble Techniques Performance could be enhanced by combining the forecasts of various credit risk models, including the BiLSTM. When using ensemble approaches, it is possible to improve prediction accuracy by utilising the advantages of many models. For example, combining Bidirectional LSTM with a Random Forest algorithm gives better performance.

3. Deployment and oversight: The model must be monitored and updated continuously after being deployed in a production setting. A system that can adapt to shifting credit patterns and guarantee consistency of performance may be designed in the future.

4. AMOTE Technique: When it comes to correcting class imbalance in machine learning datasets, AMOTE (Adaptive Minority Oversampling Technique) has an edge over SMOTE (Synthetic Minority Oversampling Technique). AMOTE, in contrast to SMOTE, takes noise into account, dynamically chooses neighbours, and adjusts to local density. It produces stronger generalisation and effectively manages data complexity, which is essential for complicated distributions. Because AMOTE is noise-adaptive, noise isn't amplified, making synthetic samples more accurate. AMOTE further lowers the danger of overfitting by maintaining the minority class’s natural structure. However, factors like dataset properties and issue domains influence the decision between AMOTE and SMOTE. To identify the strategy that best improves model performance on certain datasets, experimentation is essential.

Using the Bidirectional Long Short-Term Memory (BiLSTM) model, the credit default prediction project demonstrated the promise of deep learning in the field of credit risk assessment. The created model showed competitive predictive accuracy and emphasised the need for feature engineering and high-quality data. The project allowed for a greater comprehension of neural network designs, model training, and model evaluation. Although the project's main objectives were achieved, issues with data quantity and quality were identified. The above-mentioned future directions serve as a road map for the credit default prediction model's ongoing development and improvement, enabling it to be an effective tool for financial institutions in controlling credit risk.

7. REFERENCES

1. M. Ala'raj, M. F. Abbod, and M. Majdalawieh, "Modelling customers credit card behaviour using bidirectional LSTM neural networks," Journal of Big Data, vol. 8, no. 1, pp. 1-27, May 2021.
2. R. Anderson, "The credit scoring toolkit: theory and practice for retail credit risk management and decision automation," Oxford university press, 2007.
3. R. H. Hamilton and H. K. Davison, "Legal and ethical challenges for HR in machine learning," Employee Responsibilities and Rights Journal, vol. 34, no. 1, pp. 19-39, March 2022.
4. F. Louzada, A. Ara, and G. B. Fernandes, "Classification methods applied to credit scoring: Systematic review and overall comparison," Surveys in Operations Research and Management Science, vol. 21, no. 2, pp. 117-134, 2016.
5. K. Bastani, E. Asgari, and H. Namavari, "Wide and deep learning for peer-to-peer lending," Expert Systems with Applications, vol. 134, pp. 209-224, 2019.
6. S. S. Saha, S. S. Sandha, and M. Srivastava, "Deep convolutional bidirectional lstm for complex activity recognition with missing data," Human Activity Recognition Challenge, pp. 39-53, 2021.
7. C. Wang et al., "A deep learning approach for credit scoring of peer-to-peer lending using attention mechanism LSTM," IEEE Access, vol. 7, pp. 2161-2168, 2018.
8. M. Ala'raj et al., "A deep learning model for behavioural credit scoring in banks," Neural Computing and Applications, pp. 1-28, 2022.
9. V. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, no. 3, pp. 273-297, 1995.
10. F. Atiya and A. G. Parlos, "New results on recurrent network training: unifying the algorithms and accelerating convergence," IEEE transactions on neural networks, vol. 11, no. 3, pp. 697-709, 2000.
11. C. Wang et al., "A deep learning approach for credit scoring of peer-to-peer lending using attention mechanism LSTM," IEEE Access, vol. 7, pp. 2161-2168, 2018.
12. Y. Bengio, P. Frasconi, and P. Simard, "The problem of learning long-term dependencies in recurrent networks," IEEE international conference on neural networks, pp. 1183-1188, IEEE, 1993.
13. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp.
14. C. Y. Lai, R. C. Chen, and R. E. Caraka, "Prediction average stock price market using LSTM," ir. lib. cyut. edu. tw, 2019.
15. M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," IEEE Trans Signal Proces, vol. 45, pp. 2673-2681, 1997.
16. The dataset we used was available at https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

8. APPENDIX-A

Project Proposal Form

MILESTONE 02: Project Proposal

BSc & MSc Projects

Your first and last name

Merit Choorappadil Thomas

Your Student ID

CHO21538916

What degree programme are you on?

MSc Data Science

What is the working title of your project (this can be changed at a later date)?

Credit Default Prediction

What is the principal problem that your project aims to resolve?

A rise in using credit cards increases the chances of missing payments because the same individual has many credit cards from different banks leading to an increase in the probability of missing payments. As a result, financial institutions are under pressure to continuously improve the system which will detect early missing payment predictions in order to mitigate substantial financial losses associated with these card transaction activities. A perfect deep-learning model needs to predict these missing payments in future.

Describe your approach to solving the principal problem, and the technologies that will be used?

A data science model approach utilizing Long Short-Term Memory (LSTM), a deep learning technique, is anticipated as a perfect solution for the detection of credit default. Although several models are available for the detection of credit default, they have their own inherent drawbacks. My research would study all the existing models giving emphasis to LSTM model. The main goal would be to enhance the current detection methods and elevate accuracy, particularly when dealing with extensive amounts of data. To assess the effectiveness of the proposed enhanced model, a real dataset containing instances of credit card transactions would be employed. The performance of the improved model would be compared against an existing deep learning models, as well as other machine learning techniques. The improved model is anticipated to provide results with excellent accuracy with less time. Technologies such as Python, data processing libraries (e.g.,

Classify your project as a technology theme (i.e. How you want your project to be scrutinsed)

Artificial Intelligence & Machine Learning

If you selected "Other" above, please specify your theme below.

How will you test and evaluate your project?

A customer behavior prediction model utilizing deep learning techniques will be developed. The data will be divided into training and testing sets, and the model's performance on the testing data will be evaluated using metrics such as precision, recall, or accuracy. Cross-validation will be performed to validate the model's performance. The model will be fine-tuned for optimal results. A comparison will be made between the performance of the developed model and various other models and algorithms. Finally, the model will be tested in either a real-world or simulated environment.

Supervisor (First Marker)

Kimia Aksir

Second Marker (Second Supervisor)

Mohammad F Khan

List up to 3 aims of your project.
NOTE: An aim is an expected outcome of your project (e.g., issues it will address, how it might improve or enhance a situation for stakeholders, etc.)

1) The efficiency of predicting credit default is expected to be improved by by leveraging the strengths of various Machine Learning techniques in combination.

2) Our proposed improved model is anticipated to have the capability to effectively identify credit default of customer to the next month by capturing meaningful patterns within consumer behavior.

3) Preliminary investigations will be conducted to develop a credit default prediction model that solely relies on attention and transformer architecture, without employing recurrent networks for sequence processing.

List up to 4 key objectives of your project.
NOTE: Objectives are tangible tasks that you will complete. They are typically steps/activities that you must complete in order to deliver your project aims successfully.

1) The initial development of the project would be to incorporate feature selection and dimension reduction algorithms, for optimizing the process of learning classifiers. The problem with imbalanced data sets could be solved with the use of techniques like SMOTE.

2) As a next step, LSTM recurrent neural networks will be employed as a dynamic pattern recognition classifier which would construct a context of consumer spending behaviour

3) Fine-tune the model to optimize its ability to predict customers' credit card payment behaviour and utilization rates.Evaluate the model's ability to accurately identify high-risk accounts by comparing its predictions against historical default data or external credit risk indicators

4) Model Evaluation and Deployment

Evaluate the trained model using appropriate evaluation metrics such as accuracy, precision, recall, and F1 score.

List background/literature/technology review sources, that have been used to inform your project.

1.Research Papers:
"Long Short-Term Memory" by Sepp Hochreiter and Jürgen Schmidhuber.
"Bidirectional LSTM: A Deep Neural Network Architecture for Sequence Labeling" by Graves, Alex et al.
2.Journals and Publications:
IEEE Transactions on Neural Networks and Learning Systems
Journal of Machine Learning Research (JMLR)
Neural Networks
3.Books and Textbooks:
"Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
"Pattern Recognition and Machine Learning" by Christopher Bishop.
"Recurrent Neural Networks in Depth" by Aaron Palumbo and Charles Martin.
4.Online Platforms and Repositories:
ArXiv.org (preprint server for scientific papers)
Google Scholar (academic search engine)
GitHub (repository hosting various deep learning projects and code samples)

Describe any risks, ethical issues or other factors that are relevant to this project.

Data Privacy and Security: Working with credit card transaction data involves sensitive and personal information. Ensure that proper data anonymization and security measures are in place to protect customer privacy. Adhere to data protection regulations, such as GDPR or HIPAA, depending on the jurisdiction and context of the project. Safeguarding data from unauthorized access or breaches should be a priority.

Bias and Fairness: Deep learning models can be susceptible to biases present in the training data. Ensure that the dataset used for training the model is diverse and representative of the customer population to minimize biased predictions. Regularly monitor the model's performance and assess for any disparate impacts on certain demographic groups or protected classes.

Student and First Supervisor Project Sign Off

STUDENT: I agree to completing this project: ✔ **Date:** 29 / 06 / 2023

Student Name: Merit Choorappadil Thomas

Student Signature: Merit

SUPERVISOR: I approve this project proposal: ☐ **Date:** ☐ / ☐ / ☐

Supervisor Name:

Supervisor Signature:

NOTE: It is the supervisor's responsibility to approve this project as meeting the requirements for the module. This includes professional body requirements, programme requirements, and module requirements. By signing the form, you are agreeing that you have validated the suitability of the project.

4.2 Feature Importance and Outlier Removal

```
#Load liberaries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
# Assuming the target variable is the last column and other columns are features
X = data.drop(columns=['Y']).values
y = data['Y'].values   # Target
# List of feature names (replace with your actual feature names)
feature_names = ["X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9", "X10", "X11", "X12", "X13",
                 "X14", "X15", "X16", "X17", "X18", "X19", "X20", "X21", "X22", "X23"]
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Instantiate a Random Forest classifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
# Fit the classifier on the training data
clf.fit(X_train, y_train)
# Get feature importances
feature_importances = clf.feature_importances_
# Create a DataFrame to store feature names and their importance scores
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances})
# Sort the features by importance in descending order
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
# Print the important features
print(feature_importance_df)
```

```
    Feature  Importance
5        X6    0.099212
4        X5    0.066405
11      X12    0.060087
0        X1    0.059578
12      X13    0.054260
13      X14    0.051581
17      X18    0.051233
16      X17    0.050462
14      X15    0.050456
15      X16    0.050036
18      X19    0.047250
19      X20    0.046158
22      X23    0.045641
20      X21    0.043827
6        X7    0.043726
21      X22    0.043370
7        X8    0.024186
8        X9    0.023512
9       X10    0.021689
10      X11    0.020807
2        X3    0.020770
3        X4    0.013808
1        X2    0.011946
```

4.3 Data Loading and Splitting

```
In [7]: import pandas as pd


        # Assuming the dataset has a 'default' column indicating the target variable
        # Replace 'default' with the actual column name if it's different
        default_counts = data['Y'].value_counts()

        # Calculate the percentage of default and non-default instances
        total_instances = len(data)
        percentage_default = (default_counts[1] / total_instances) * 100
        percentage_non_default = (default_counts[0] / total_instances) * 100

        print(f"Percentage of Default: {percentage_default:.2f}%")
        print(f"Percentage of Non-Default: {percentage_non_default:.2f}%")

        Percentage of Default: 22.12%
        Percentage of Non-Default: 77.88%
```

Appendix-B

5.1 Logistic Regression Results

```
#Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

#Split the data into training and testing sets (80% for training, 20% for testing)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

#Create a Logistic Regression model
logreg_model = LogisticRegression(max_iter=1000)  # Increase max_iter to allow more iterations
```

+ Code + Text

```
#Train the model on the training data
logreg_model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
#Make predictions on the test data
y_pred = logreg_model.predict(X_test)
```

```
Accuracy: 80.97%
Precision: 69.13
Recall: 23.53
F1-Score: 35.11

Confusion Matrix:
[[4549  138]
 [1004  309]]
ROC AUC: 0.6029478889527856
```

	precision	recall	f1-score	support
0	0.82	0.97	0.89	4687
1	0.69	0.24	0.35	1313
accuracy			0.81	6000
macro avg	0.76	0.60	0.62	6000
weighted avg	0.79	0.81	0.77	6000

5.2 Random Forest Classifier Results

```
#Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
#Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Step 4: Train the Random Forest Model
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
Accuracy: 81.60%
Precision: 63.80
Recall: 36.79
F1-Score: 46.67

Confusion Matrix:
[[4413  274]
 [ 830  483]]
ROC AUC: 0.6547001469443363
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      4687
           1       0.64      0.37      0.47      1313

    accuracy                           0.82      6000
   macro avg       0.74      0.65      0.68      6000
weighted avg       0.80      0.82      0.80      6000
```

5.3 SVM Results Results

```
# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create an SVM classifier
svm = SVC(kernel='linear', random_state=42)
# Train the classifier
svm.fit(X_train, y_train)
SVC(kernel='linear', random_state=42)
# Make predictions on the test set
y_pred = svm.predict(X_test)
```

```
Accuracy: 80.43%
Precision: 59.46
Recall: 33.28
F1-Score: 42.68

Confusion Matrix:
[[4389  298]
 [ 876  437]]
ROC AUC: 0.6346227375195217
              precision    recall  f1-score   support

           0       0.83      0.94      0.88      4687
           1       0.59      0.33      0.43      1313

    accuracy                           0.80      6000
   macro avg       0.71      0.63      0.65      6000
weighted avg       0.78      0.80      0.78      6000
```

5.4 Bidirectional LSTM with Attention Results

```
Average Confusion Matrix Across Folds:
[[3445.6  289.8]
 [ 146.4 3589. ]]
Average Precision Across Folds: 0.9280093760849374
Average Recall Across Folds: 0.960805133453156
Average F1-Score Across Folds: 0.943824792009109
```

```
 Average Metrics for Class 0:
 Average Precision: 0.9572748455700555
 Average Recall: 0.9224135227562641
 Average F1-Score: 0.9391383580416702

 Average Metrics for Class 1:
 Average Precision: 0.9280093760849374
 Average Recall: 0.960805133453156
 Average F1-Score: 0.943824792009109

 Average Accuracy Across Folds: 0.9416094037507279
```

Appendix-C

3.8 Project Management Tool

Teamwork link

https://universityofroehampton15.teamwork.com/app/home/projects

Username: merit.thomas@gmail.com

Password: Merit@01

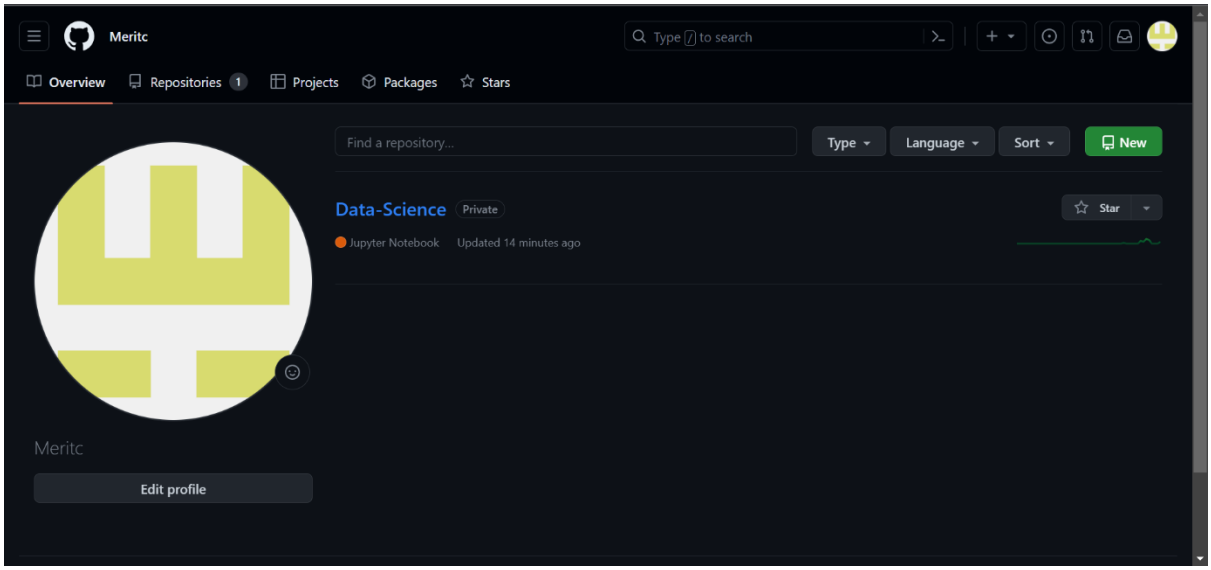
GitHub link

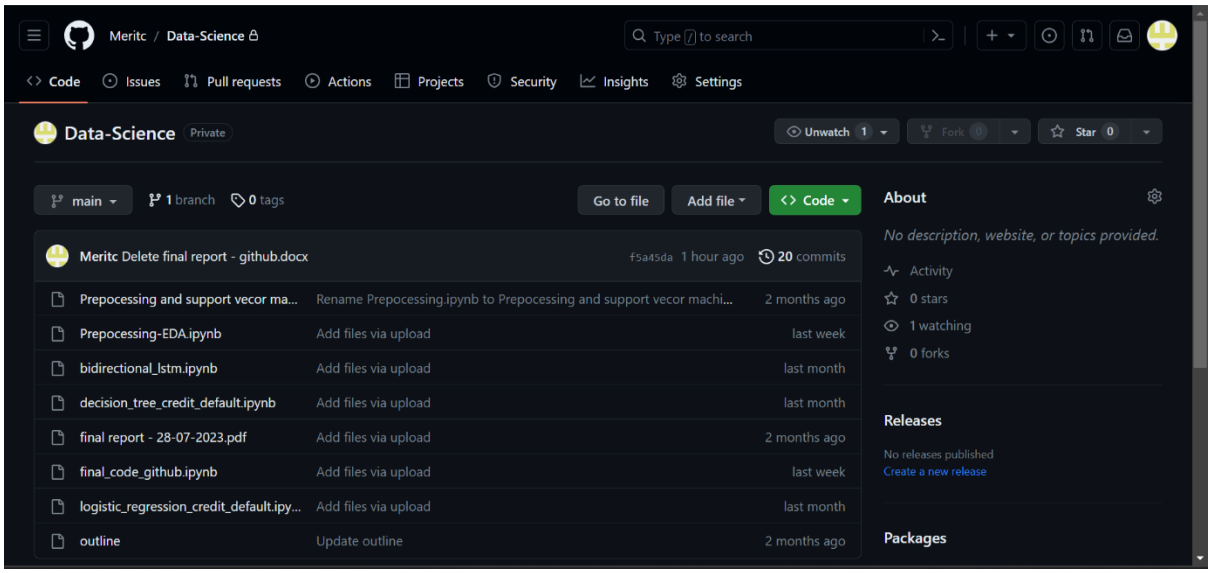
https://github.com/Meritc/Data-Science

Username: merit.thomas@gmail.com

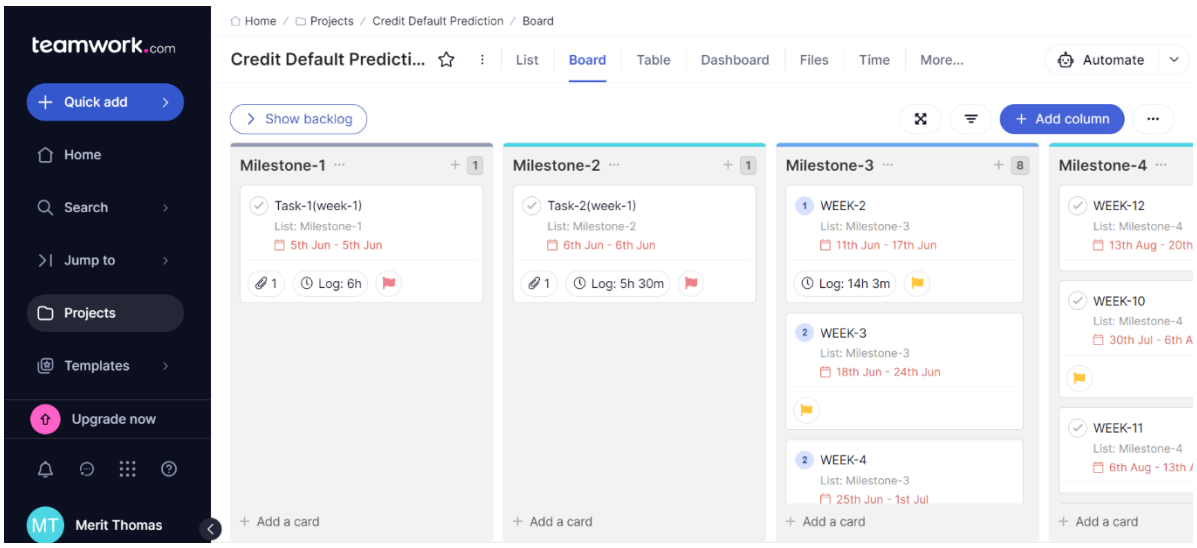
Password: Meritcthomas@01

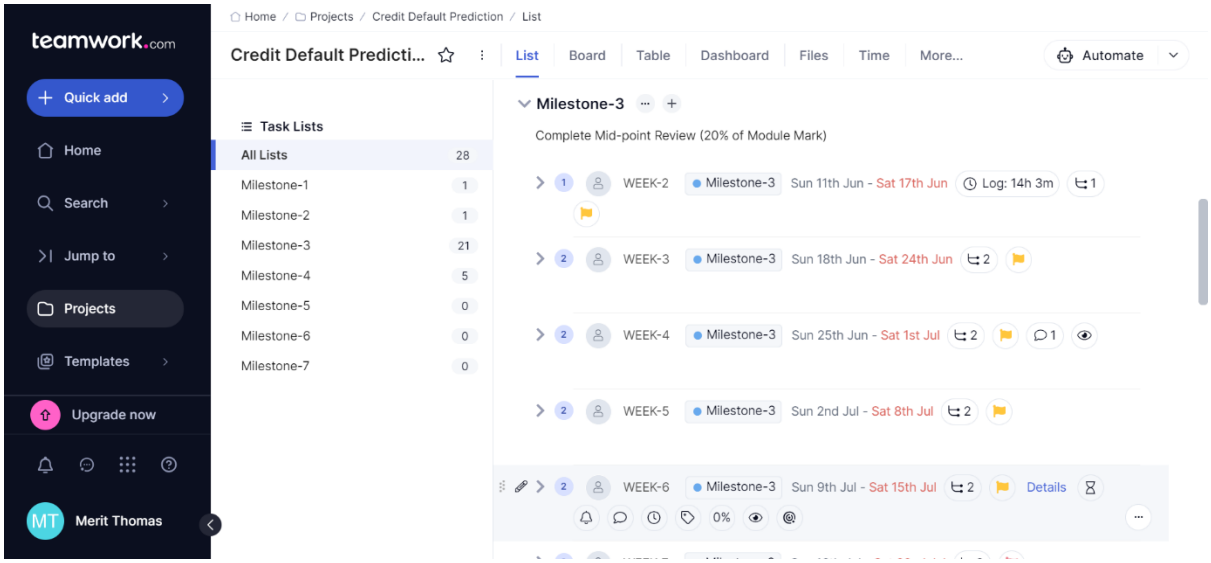
a)Github


Meritc
Type / to search
Overview
Repositories 1
Projects
Packages
Stars
Find a repository...
Type
Language
Sort
New
Data-Science
Private
Star
Jupyter Notebook
Updated 14 minutes ago
Meritc
Edit profile


Meritc / Data-Science
Type / to search
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
Data-Science
Private
Unwatch 1
Fork 0
Star 0
main
1 branch
0 tags
Go to file
Add file
Code
About
No description, website, or topics provided.
Activity
0 stars
1 watching
0 forks
Meritc Delete final report - github.docx
f5a45da 1 hour ago
20 commits
Prepocessing and support vecor ma...
Rename Prepocessing.ipynb to Prepocessing and support vecor machi...
2 months ago
Prepocessing-EDA.ipynb
Add files via upload
last week
bidirectional_lstm.ipynb
Add files via upload
last month
decision_tree_credit_default.ipynb
Add files via upload
last month
final report - 28-07-2023.pdf
Add files via upload
2 months ago
final_code_github.ipynb
Add files via upload
last week
logistic_regression_credit_default.ipy...
Add files via upload
last month
outline
Update outline
2 months ago
Releases
No releases published
Create a new release
Packages

b)Team Work


teamwork.com
Quick add
Home
Search
Jump to
Projects
Templates
Upgrade now
Merit Thomas
Home / Projects / Credit Default Prediction / Board
Credit Default Predicti...
List
Board
Table
Dashboard
Files
Time
More...
Automate
Show backlog
Add column
Milestone-1
Task-1(week-1)
List: Milestone-1
5th Jun - 5th Jun
Log: 6h
Milestone-2
Task-2(week-1)
List: Milestone-2
6th Jun - 6th Jun
Log: 5h 30m
Milestone-3
WEEK-2
List: Milestone-3
11th Jun - 17th Jun
Log: 14h 3m
WEEK-3
List: Milestone-3
18th Jun - 24th Jun
WEEK-4
List: Milestone-3
25th Jun - 1st Jul
Milestone-4
WEEK-12
List: Milestone-4
13th Aug - 20th
WEEK-10
List: Milestone-4
30th Jul - 6th A
WEEK-11
List: Milestone-4
6th Aug - 13th
Add a card


teamwork.com
Quick add
Home
Search
Jump to
Projects
Templates
Upgrade now
Merit Thomas
Home / Projects / Credit Default Prediction / List
Credit Default Predicti...
List
Board
Table
Dashboard
Files
Time
More...
Automate
Task Lists
All Lists 28
Milestone-1 1
Milestone-2 1
Milestone-3 21
Milestone-4 5
Milestone-5 0
Milestone-6 0
Milestone-7 0
Milestone-3
Complete Mid-point Review (20% of Module Mark)
WEEK-2 Milestone-3 Sun 11th Jun - Sat 17th Jun Log: 14h 3m 1
WEEK-3 Milestone-3 Sun 18th Jun - Sat 24th Jun 2
WEEK-4 Milestone-3 Sun 25th Jun - Sat 1st Jul 2 1
WEEK-5 Milestone-3 Sun 2nd Jul - Sat 8th Jul 2
WEEK-6 Milestone-3 Sun 9th Jul - Sat 15th Jul 2 Details
0%