UNIVERSITY OF BIRMINGHAM

School of Computer Science
College of Engineering and Physical Sciences

MSc. Project

Machine Learning & Deep Learning Approaches to Predict Credit Card Default

Submitted in conformity with the requirements
for the degree of MSc. Artificial Intelligence & Computer Science
School of Computer Science
University of Birmingham

Sarathkumar Padinjare Marath Sankaranarayanan
Student ID: 2359859
Supervisor: Dr.Kashif Rajpoot

September 2022

Abstract

The material contained within this report has not previously been submitted for a degree at the University of Birmingham or any other university. The research reported within this report has been conducted by the author unless indicated otherwise.

Keywords Credit Card Default Prediction, Ensemble Learning

Sarathkumar Padinjare Marath Sankaranarayanan

Declaration

The material contained within this report has not previously been submitted for a degree at the University of Birmingham or any other university. The research reported within this report has been conducted by the author unless indicated otherwise.

Signed Sarathkumar Padinjare Marath Sankaranarayanan

Sarathkumar Padinjare Marath Sankaranarayanan

"You have to learn the rules of the game.
And then you have to play better than anyone else"
Albert Einstein

MSc. Project
Machine Learning & Deep Learning Approaches to Predict Credit Card Default

Sarathkumar Padinjare Marath Sankaranarayanan

Contents

Table of Abbreviations

SVM	Support Vector Machine
ANN	Artificial Neural Network
GBDT	Gradient Boosting Decision Tree
GRU	Gated Recurrent Unit
LGBM	Light Gradient Boosting Machine
XGBoost	Xtreme Gradient Boosting Machine
GRU	Gated Recurrent Unit
CV	Cross Validation
SMOTE	Synthetic Minority Oversampling Technique
RAM	Random Access Memory
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
CFS	Correlation Based Feature Selection
AUC	Area Under the Curve
PCA	Principle Component Analsys
RNN	Recurrent Neural Network
RF	Random Forest Classifier
KNN	K-Nearest Neighbours
RBF	Radial Basis Function
DT	Decision Tree Classifiers
GOSS	Gradient based One-side Sampling
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
MSE	Mean Squared Error
BPTT	Backpropogation Through Time
BRNN	Bidirectional Recurrent Neural Network
LSTM	Long Short-term Memory
LS-SVM	Least Square SVM

List of Figures

List of Tables

1 Introduction

This section will introduce the user to definitions of terms relevant for understanding the problem, discuss the motivation behind the problem, the aim & approach taken to solve the problem, and the structure of this report.

1.1 Definitions

1.1.1 Credit Card Statement Date

The credit card statement date is the date on which the statement/bill is generated every month. Any transaction conducted on the card post billing date will reflect in the next month's credit card statement.

1.1.2 Delinquent Account

A credit card account is considered delinquent if the customer has failed to make the minimum monthly payment for 30 days from the original due date.

1.1.3 Delinquency Rate

The percentage of credit card accounts within a financial institution's portfolio whose payments are delinquent.

$$DelinquencyRate = \left(\frac{NumberOfDelinquentCreditCardAccounts}{TotalNumberOfCreditCardAccount}\right) * 100 \tag{1}$$

1.1.4 Credit Card Default

The customer is considered as defaulting customer in the event of nonpayment of the due amount in 120 days after the latest statement date.

1.2 Motivation

Delinquency rates & credit card default rates are directly proportional. According to the figure 1, the delinquency rates were at an all-time high just before the recession started in 2008; moreover, this was the same time when more & more customers began to default on credit card payments.

Predicting credit defaults is essential for managing risk in the consumer lending industry. Credit default prediction enables lenders to make the best possible lending decisions, improving customer satisfaction and fostering strong company economics.

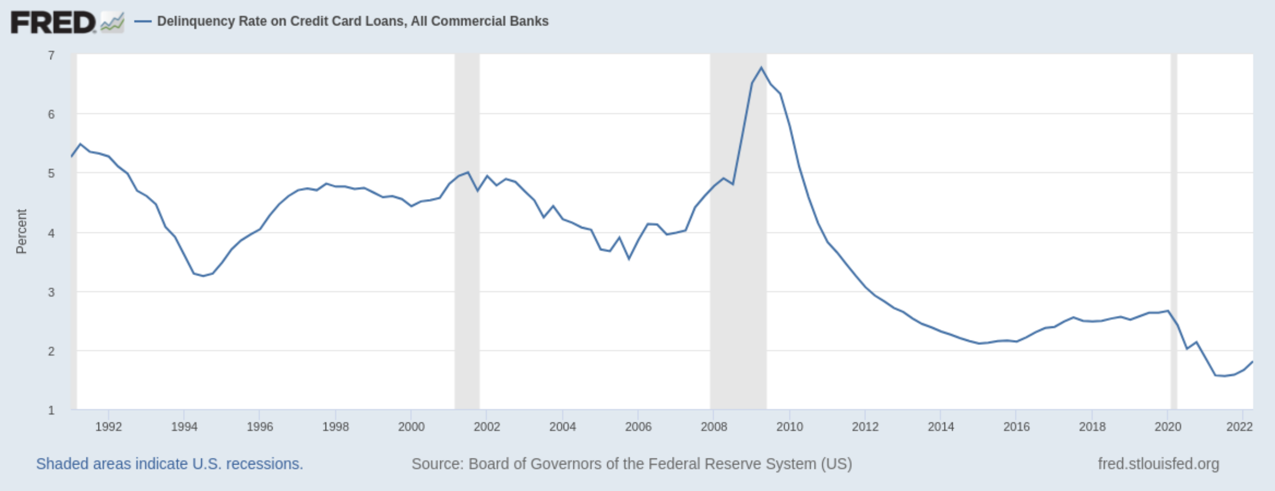


Figure 1: Delinquency rate on credit card loans for the period 1992-2022(Board of Governors of the Federal Reserve System (US) 2022).

Existing models can be used to manage risk. However, developing models that perform better than those in use is feasible.

1.3 Aim & Approach

The objective of this project was to explore different machine learning algorithms & deep learning architectures on the American Express default prediction dataset(American Express 2022) to predict if a customer will default on the payment in the future. The project work started by developing a model using classic machine learning algorithm Support Vector Machine (SVM) followed by creating multiple models using Random Forest Classifier & Gradient Boosting Decision Tree (GBDT) algorithms. Artificial Neural Network (ANN), Gated Recurrent Unit (GRU) & Custom ensemble model created by model combining ANN, GRU and GBDT were created as part of exploring deep learning architectures. Finally created a lean model, using less features & optimzed parameters using GBDT which provided comparable performances to the previously explored models.

1.4 Structure of Report

The remainder of the report is structured as follows: in Section 2 background information on different machine learning & deep learning algorithms along with metrics explanation is provided. Then in Section 3 a literature review related to the credit card default prediction research is given. Section 4 & 5 provides detailed explanations on the dataset, tools & software used in the project, and methodology followed for creating the models. Model evaluation results and the comparison is given in Section 6. Finally Section 7 discusses the conclusion of the project.

2 Background Knowledge

This section provides the reader with the required background information on the machine learning algorithms, deep learning architectures, data preprocessing techniques & model evaluation metrics. The explanations provided in this section are at intutive level only without going deeper into the mathematical formulation.

2.1 Support Vector Machine (SVM)

SVM is a reliable classification and regression machine learning algorithm that maximizes the accuracy of a model without overfitting the training data. There are 4 main components to the SVM model, Hyperplane, Support Vectors, Margin, Kernel function. Hyperplane refers to the decision boundary of the model, this could be a line if the data is 2 dimensional, plane if the data is 3 dimensional, or hyperplane if the data is n dimensional. Support Vectors refers to the data points that are nearest to the hyperplane and these points are more difficult to classify. As shown in figure 2, the SVM algorithm tries to find a hyperplane which maximizes the distance between the support vectors and the hyperplance and hence reducing the overfitting on the training set. SVM machinel learning technique uses a concept of Soft Margin to remove the issues that may arise due to outliers in the dataset.Additionally, if the decision boundary is non-linearly separable, SVM transforms original data to map into new space using the Kernel function. Linear Kernel, Polynomial kernel & Radial Basis Function (RBF) are different kernel functions that can be used in SVM model. There are multiple implementations of the SVM technique is available; however, in this project libsvm(Chang & Lin 2011) & liblinear (Fan et al. 2008) is used as these implementations provide higher performance on large datasets.

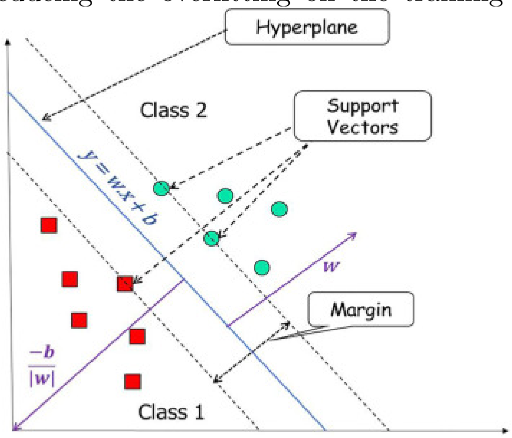


Figure 2: Components of SVM (Rani et al. 2022)

2.2 Ensemble Learning

Ensemble methods are highly effective compared to the traditional machine learning techniques and considered state-of-the-art approach for solving many challenges (Sagi & Rokach 2018). The idea behind ensemble learning technique is to train multiple base learners/models and combine the predictions from each learner to make the final prediction. As shown in figure 3, ensemble techniques can be mainly categorized into 2 types Homogeneous & Heterogeneous methods. In homogeneous methods all the base learners uses the same machine learning technique; on the other hand, heterogeneous methods may use different types machine learning techniques as the base learner.

2.2.1 Bagging

Bagging is a Homogeneous Ensemble Method where the base learners are trained in parallel on the complete training set or a subset of training set based on the configuration. As depicted in figure 4, initially creates multiple dataset through random sampling with replacement, then train the mutiple learners in parallel. Finally combine output from all learners using either taking average or using majority vote strategy based on the problem is being solved.

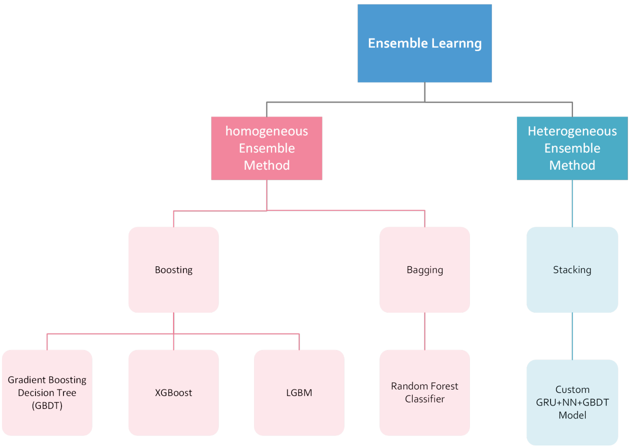


Figure 3: Ensemble Learning Techniques & Implementations

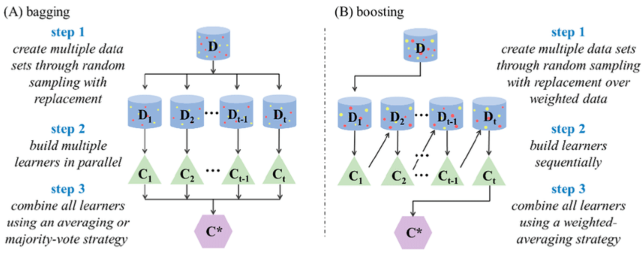

(A) bagging
step 1
create multiple data sets through random sampling with replacement
step 2
build multiple learners in parallel
step 3
combine all learners using an averaging or majority-vote strategy
D
D_1 D_2 ⋯ D_{t-1} D_t
C_1 C_2 ⋯ C_{t-1} C_t
C*
(B) boosting
step 1
create multiple data sets through random sampling with replacement over weighted data
step 2
build learners sequentially
step 3
combine all learners using a weighted-averaging strategy
D
D_1 D_2 ⋯ D_{t-1} D_t
C_1 C_2 ⋯ C_{t-1} C_t
C*

Figure 4: Illustrations of bagging and boosting ensemble algorithms(Yang et al. 2019).

2.2.2 Boosting

Boosting is a Homogeneous Ensemble Method where the base learners are trained sequentially and the predictions from individual learners are combined to make the final prediction. The model starts by creating a base learner which performs slightly better than the random prediction, then subsequent learners try to improve on the prediction made by the previous learner. The process ends when the improvement made by the new learner is less than the threshold. As depicted in figure 4, the boosting methods uses a weighted averaging strategy to combine the predictions from the individual learners.

2.2.3 Stacking

Stacking is a Heterogeneous Ensemble Method where the output from the base learners is passed through another learner to make the final prediction. In Bagging & Boosting methods the final prediction was made using taking average, majority voting, or weighted average; however the Stacking models uses a learner to make the final prediction. Figure 5 shows a sample Ensemble Stacking Model. Model A, B, and C are base learners which are trained parallely. The output from Model A, B & C is then passed through a Generalizer/Meta Learner which predicts the final output.

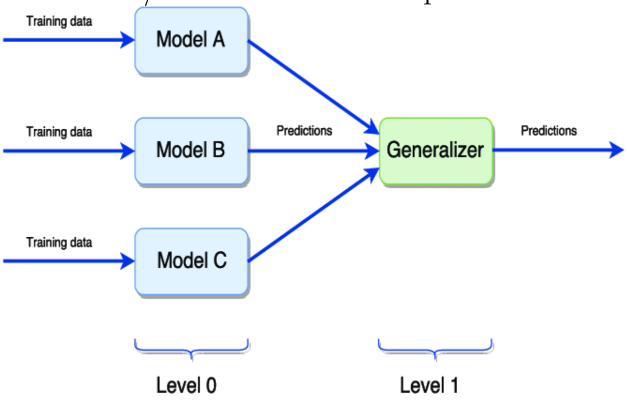


Figure 5: An example scheme of stacking ensemble learning Divina et al. (2018).

2.3 Random Forest Classifier (RF)

Random Forest Classifier (RF) (Breiman 2001) is an ensemble bagging method technique which uses Decision Tree Classifiers (DT) to create base learners. DT are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. RF creates multiple DT and trains each one of them with a random subsample of the dataset. The predictions from each DT is then combined by either taking average or by using majority voting strategy.

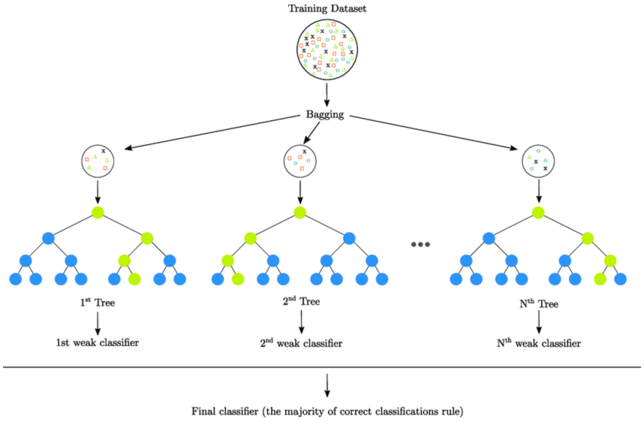


Figure 6: An example of Random Forest Classifier (RF) Sapountzoglou et al. (2020).

2.4 Gradient Boosting Decision Tree (GBDT)

Gradient-boosted decision trees are a machine learning technique for optimizing the predictive value of a model through successive steps in the learning process. Each iteration of the decision tree involves adjusting the values of the coefficients, weights, or biases applied to each of the input variables being used to predict the target value, with the goal of minimizing the loss function (the measure of difference between the predicted and actual target values). The gradient is the incremental adjustment made in each step of the process; boosting is a method of accelerating the improvement in predictive accuracy to a sufficiently optimum value.

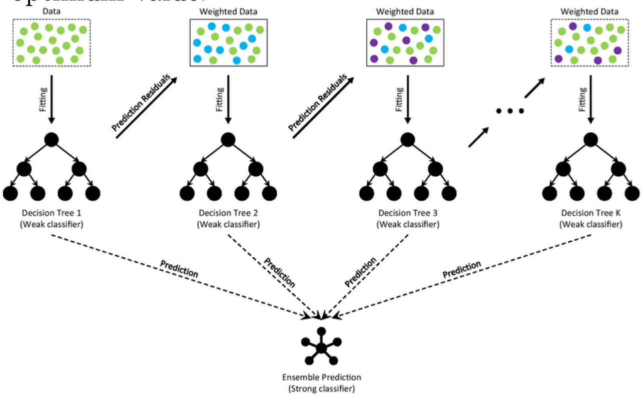


Figure 7: Architecture of Gradient Boosting Decision Tree (GBDT) Deng et al. (2021).

2.4.1 Xtreme Gradient Boosting Machine (XGBoost) & Light Gradient Boosting Machine (LGBM)

Xtreme Gradient Boosting Machine (XGBoost)(Chen & Guestrin 2016) and Light Gradient Boosting Machine (LGBM)(Ke et al. 2017) are two mostly used Gradient Boosting Decision Tree (GBDT) framework/libraries which performs much better than the basic GBDT models (Machado et al. 2019). Xtreme Gradient Boosting Machine (XGBoost) & LGBM frameworks differ in how the individual weak learners are constructed. As shown in figure 8, the XGBoost uses Level Wise Tree Growth strategy while the LGBM uses Leaf Wise Growth Strategy. Gradient based One-side Sampling (GOSS) technique is used by the LGBM to split the nodes while generating the individual weak learners; on the other hand, the XGBoost uses Pre-Sorted and historgram based algorithms for splitting nodes.

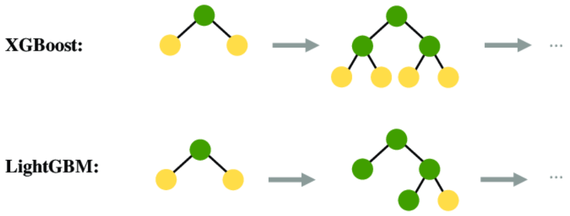


Figure 8: XGBoost Level wise tree growth and LGBM Leaf wise tree growth.Rezazadeh (2020).

Both frameworks require the features to be numerical and does not support text features; thus; categorical encoding must be done before training LGBM & XGBoost models. In addition, both these libraries are well optimized for parallel processing and hence can be used for large datasets.

2.5 Artificial Neural Network (ANN)

ANN are a subset of Machine Learning algorithms. The architecture and name of ANN is inspired by how the human brains works, specifically, how biological neurons signal to one another(IBM 2022*a*). Figure 9 represents the architecture of a ANN in general. The network consists of an input layer, multiple hidden layers and an output layer. There are multiple nodes, also known as neurons, in each layer. The neurons are the building blocks of the ANN network. Each neuron connects to all the neurons in the next layer; moreover, neurons in the input layer are basically the input features.

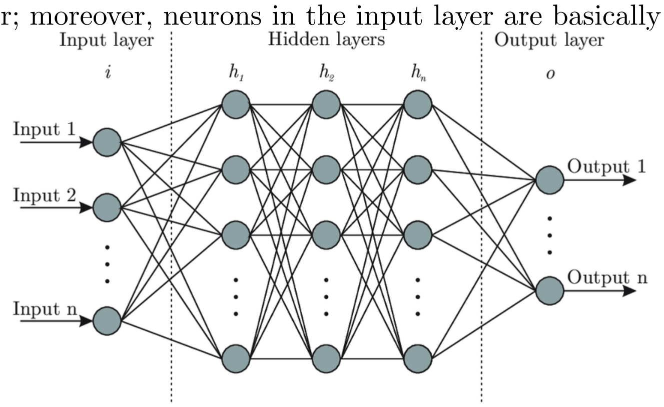


Figure 9: Architecture of Artificial Neural Network (ANN)Bre et al. (2018).

Neurons are the basic building block of an ANN. As shown in figure **??**, each neuron at least consists of a learning function and an activation function. Learning function can be thought of as a simple logistic/linear regression model and the activation function can be though of as a gate-keeper which decides how much influence this neuron should have in the next layer. Commonly used activation functions are Sigmoid, ReLU and tanh.

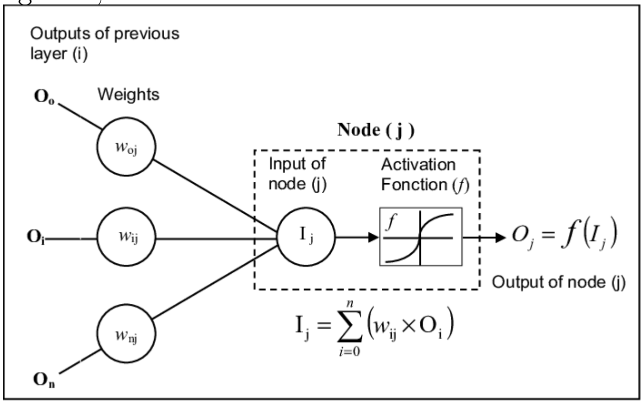


Figure 10: Representation of a Neuron in Artificial Neural Network (ANN).Ghedira & Bernier (2004)

While training ANN models, a cost function is assigned to evaluate the performance of the model, Mean Squared Error (MSE), Binary Cross Entropy are some of the cost functions which are generally used. Finally the weights learning function of each neuron is adjusted through an algorithm called Backpropogation based on the result of the cost function. The algorithm ends when the cost function converges (figure 11) and the improvement made to the performance is less than the threshold.

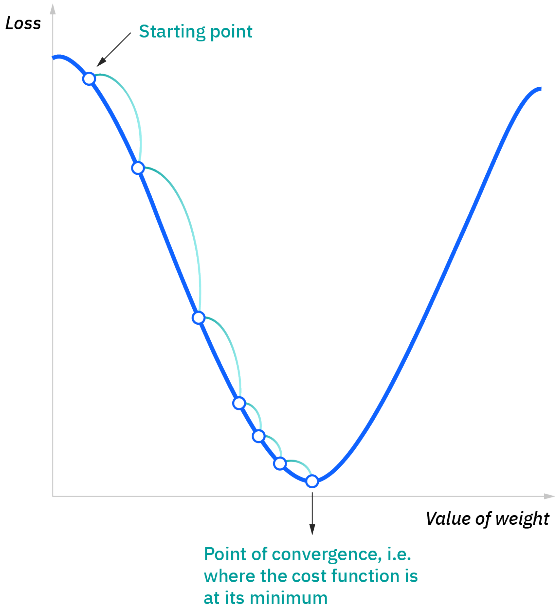


Figure 11: Representation of Convergence of Cost Functions(IBM 2022*a*)

2.6 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a type of ANN which uses sequential data or time series data. RNN has the capability of using previous inputs in the sequence to influence the current input and output, in other words, the RNN has a memory of past events/data which it uses to calculate the current output.Figure 12 shows comparison of a normal feed forward neural network (Tradition Neural Network) and a RNN. In a Feed Forward Neural Network the signals flow only in one direction from input to output; however, In an RNN output of a layer is added to the next input and fed back into the same layer.Traditional Deep Learning networks assumes no dependency between the inputs and outputs, on the other hand, the output of RNN depends on the previous inputs in the sequence.

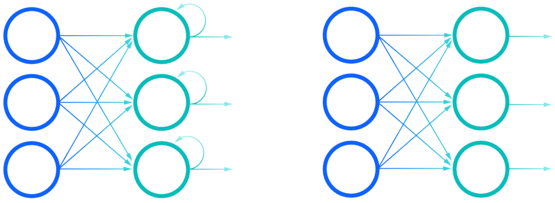


Figure 12: Comparison of Recurrent Neural Networks (on the left) and Feedforward Neural Networks (on the right)(IBM 2022*b*).

Consider the Credit Card Default problem with 12 month data, the Rolled RNN represents the entire Neural Network, and the Unrolled RNN represents each layer in the Neural Network or each time step. In credit card default problem, X_0 represents the data for the first month, X_1 for the second month and so on. RNN uses Backpropogation Through Time (BPTT) algorithm and gradient descent to learn the weights of the model while training.

There are 3 variants of the RNN architectures as shown in figure 14. While RNN uses previous inputs to influence the current output, BRNN use future inputs as well to improve the accuracy of the RNN architecture. Long Short-term Memory (LSTM) is variant of RNN proposed Sepp Hochreiter and Juergen Schmidhuber in their paper (Hochreiter & Schmidhuber 1997). If the previous input which is influencing the current input is not in recent past, RNN might not be able predict the current output accurately. LSTM tries to solve this issue using cells in the hidden layers of the neural network. Cells have three different gates, input, output and forget gate, these gates control the flow of information which is needed to predict the output in the network. Gated Recurrent Unit (GRU)

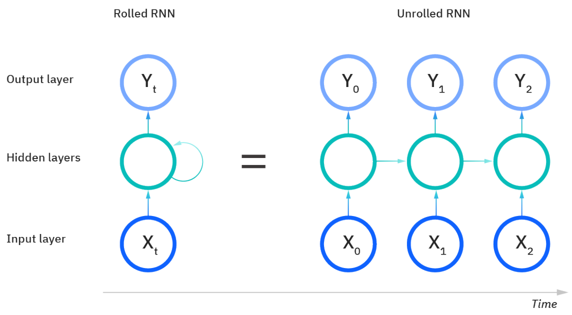


Figure 13: RNN rolled and unrolled (IBM 2022*b*).

is similar to LSTM and tries to resolve the issue of Short-term memory issue of RNN. GRU uses hidden states to regulate the information in the network and uses two gates, reset, update gate, to control the flow of information through network (Cho et al. 2014).

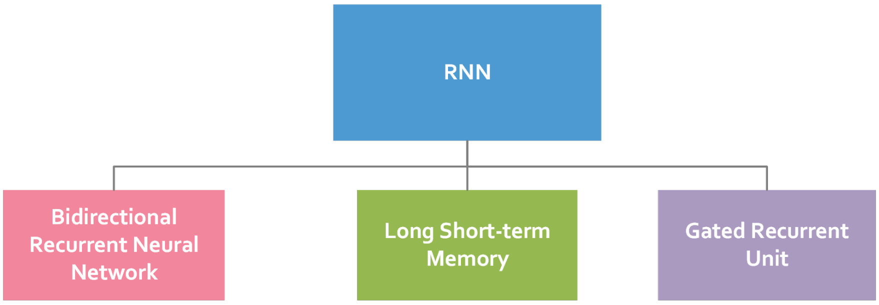


Figure 14: Variant RNN Architectures.

2.7 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) is a variant of RNN network which tries to solve the short-term memory issue of RNN networks. Figure 15 represents the structure for GRU unit. Input to the current unit is the output hidden state from the previous input ($\mathrm{h}_{\mathrm{t-1}}$);furthermore, GRU has two gates, update(z_{t}) and reset(r_{t}) gate. The algorithm uses reset gate to calculate the candidate output state (h_{t}'). Finally, update gate z_{t}, previous hidden state $\mathrm{h}_{\mathrm{t-1}}$ and candidate output hidden state h_{t}' are used to calculate the final output hidden state h_{t}.

Update gate is calculated using equation 2 which is highlighted in figure 16 (a) of update gate. Update gate is formed adding the result of multiplication of input x_{t} with its own weight W^{z} and previous hidden state , $\mathrm{h}_{\mathrm{t-1}}$ with it's own weight U^{z}. Finally applying Sigmoid activation function to make the value between 0 and 1. Update gate determines how much of the information from the previous time steps needs to be passed to the future.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{2}$$

Reset gate is calculated using equation 3 which is highlighted in figure 16 (b). Reset gate may seem similar to update gate by looking at the formula;;however, the important thing to note is that the weights are different for update & reset gate and hence it learns different characteristics of the network. Reset gate determines how much of the memory from past inputs to forget.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{3}$$

Calculation of new candidate output hidden state is shown in equation 4 and 16 (c). Both the input and previous output hidden state is multiplied by the respective weights before taking sum. Additionally, the product of previous hidden state and the weight is further multiplied (Hadamard

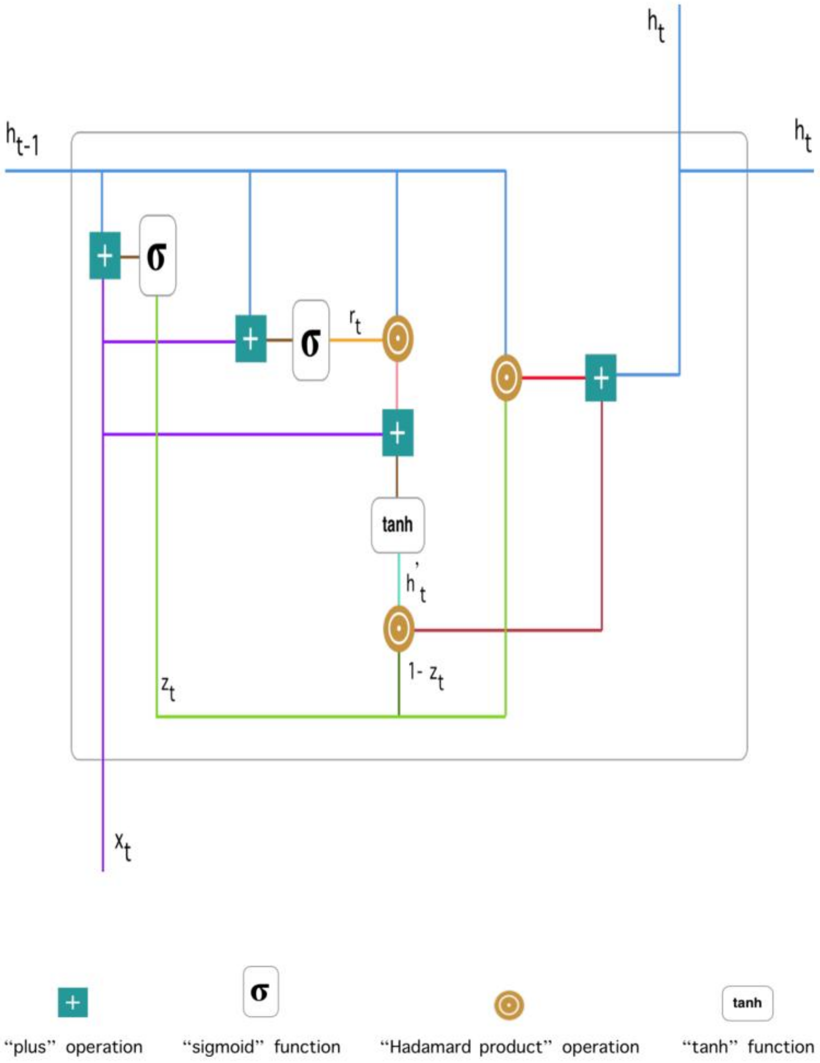


Figure 15: GRU Architecture (Simeon Kostadinov 2017)

Product) with the reset gate, this operation determines how much of the previous information to be passed over to the current output hidden state.

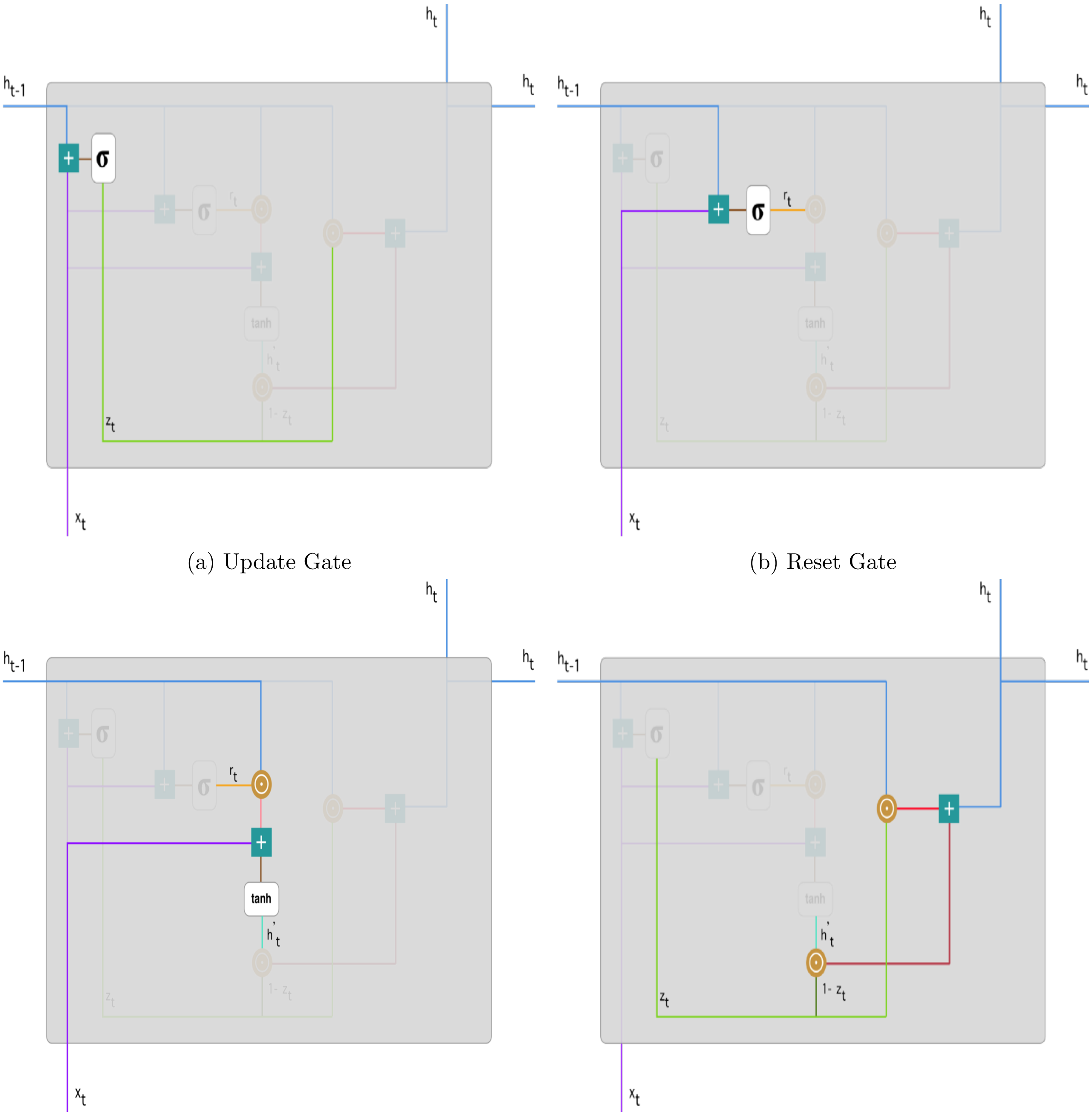
$$h_t^{'} = \tanh(Wx_t + r_t \odot Uh_{t-1}) \tag{4}$$

Finally the update gate, previous output hidden state and the candidate current output hidden state is used to calculate the final current output state as shown in equation 5 and figure16 (d).

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'} \tag{5}$$

2.8 Feature Selection

While building a Machine Learning model, all of the features might not contribute equivalently to the model's prediction performance, some of the features might impact the model's prediction performance adversely. This problem becomes more prominent on high dimensional data. The process of identifying the important features which improve the performance of the model is called Feature Selection. Two feature selection methods which were used in this project are Select From Model & Sequential Feature Selection.



(a) Update Gate

(b) Reset Gate

(c) Candiate Output State

(d) Final Output State

Figure 16: GRU Architecture Stepwise (Simeon Kostadinov 2017)

2.8.1 Select From Model

In this method, the training dataset is first trained on a machine learning model which is computationally less expensive and provides acceptable level of performance, this model can be considered as a filter. Then remove the features whose feature weights are less than the threshold value in the filter model. Finally, use the filtered dataset for training the main model. In this project, the SVM is used as the filter model for training the main LGBM model.

2.8.2 Sequential Feature Selection

As shown in figure 17, sequential feature selection starts with a subset of dataset and then an estimator chooses the best feature to add or remove to the dataset based on the Cross Validation (CV) score obtained by adding/removing each feature. The process ends once the algorithm reaches the exit criteria. Exit criteria could be either the count of features or until improvement in performance gained by adding/removing features is not greater than a threshold. There are two types Sequential Feature Selection techniques, Forward Selection & Backward Selection. In Forward Selection, the

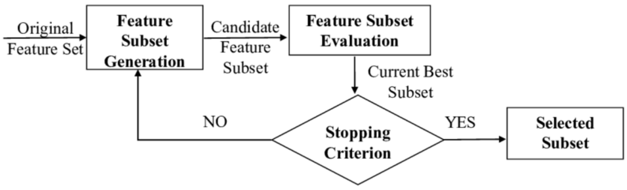


Figure 17: Sequential Feature Selection Process(Beyan 2015)

initial feature subset contains only one feature, then additional features are added based on CV score until the exit criteria is met. On the other hand, in Backward Selection technique, the initial feature subset contains all the features, then the algorithm removes the features based on CV score until the exit criteria is met.

2.9 Encoding Categorical Features

As discussed in section 2.4.1, some of the machine learning techniques does not support categorical text variables. Two most common methods used to encode categorical features are Ordinal Encoder & One-hot encoder. Ordinal encoder is better than the One-hot encoder in memory efficiency.

2.9.1 Ordinal Encoder

In this method, the feature is transformed into a numerical value by assigning numbers to each distinct category. For eg: in Scikit Learn library the categorical variables low, medium, high will be transformed to 0, 1, 2; moreover, a default number can be assigned to categories which are not seen in the training dataset.

2.9.2 One-hot Encoder

In this method, the categorical feature column is transformed into n different features each representing one distinct category. For eg: a feature X with low,medium,high as distinctive categorical values, will be transformed into 3 features X_low, X_medium, X_high. X_low will have a value of 1 if the X='low' for the record, likewise for the other features.

2.10 Data Oversampling

A classification data set with skewed class proportions is called imbalanced. Classes that make up a large proportion of the data set are called majority classes. Those that make up a smaller proportion are minority classes. Data sampling is a method used to overcome / reduce the effect of Class Imbalance on the performance of the model. There two types of Data Sampling, Over Sampling & Under Sampling. Over Sampling techniques boost the minority class entries by introducing new records in minority class; on the other hand, the Under Sampling methods eliminates entries from majority class and makes the majority & minority class entries to similar proportion.

In this project, Data Oversampling techniques are used as the dataset is large. Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al. 2002) & KMeans SMOTE (Last et al. 2017) are two different Oversampling methods which were explored in this project.

2.10.1 Synthetic Minority Oversampling Technique (SMOTE)

SMOTE method introduces new data points in the minority class by finding nearest neighbors and adding new data point along the line of nearest neighbor. Figure 18 shows the SMOTE process, in this picture green points are minority class & blue points are majority class. The algorithm first selects a data point from the minority class, then finds the K nearest neighbors among the minority class. Finally one of the K nearest neighbor is choosen randomly and a new synthetic minority class data point (red) is added along the straight line connecting the selected data point and the choosen nearest neighbor.

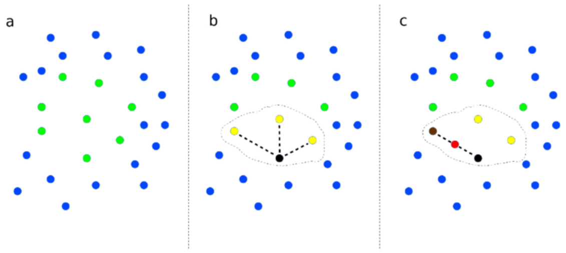


Figure 18: SMOTE algorithm (Schubach et al. 2017)

2.10.2 KMeans SMOTE

In KMeans SMOTE method, the minority class is first passed through a KMeans clustering model before applying the SMOTE algorithm. Clustering before applying SMOTE helps to eliminate the problem of oversampling the outliers in the minority classes;moreover, clustering also helps to apply the SMOTE algorithm for non-linearly separable data. As shown in figure 19, the SMOTE algorith is applied separately on each cluster.

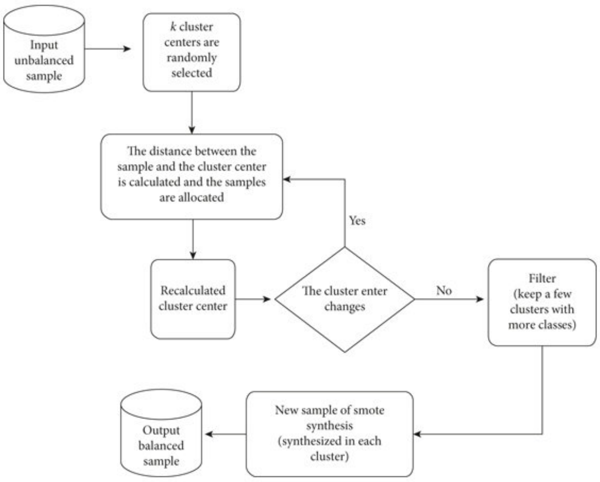

Input unbalanced sample
k cluster centers are randomly selected
The distance between the sample and the cluster center is calculated and the samples are allocated
Yes
Recalculated cluster center
The cluster enter changes
No
Filter (keep a few clusters with more classes)
Output balanced sample
New sample of smote synthesis (synthesized in each cluster)

Figure 19: KMeans SMOTE Flowchart (Chen & Zhang 2021)

2.11 Cross Validation (CV)

Cross Validation (CV) is a model evaluation technique where a subset of the training dataset is removed and kept for evaluation and optimization of the model. Its a common practice to split the dataset into 3 set, Training, Validation & Test set. Training set is used to build the model; on the other hand, Validation set is used to optimize the model parameters built on the Training set. Test set is used to evaluate the performance of the final model;importantly, it is assumed that the Test set has no influence on the model creation & optimization and it is unseen while the model is trained and optimized.

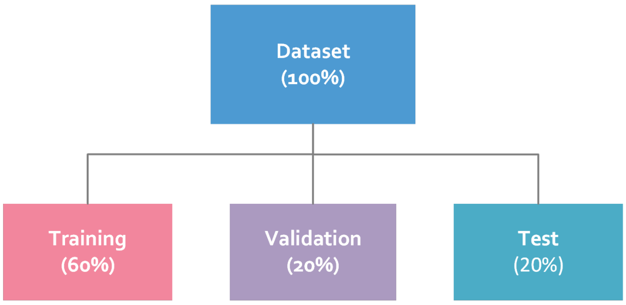


Figure 20: A sample Train,Validation & Test Split

Figure 21 shows different Cross-Validation techniques, all these technique differs on how it selects/creates the validation set. For eg: Leave One Out Cross-Validation technique selects one data record from the Training set as the Validation set (without replacement) and the rest is used for training, this process is repeated n times (number of data records in Traning set). The model evaluation metric collected in each iteration and the average of collected metric taken to get the final value of metric. Section 2.11.1 & 2.11.2 will present couple of CV techniques used in this project.

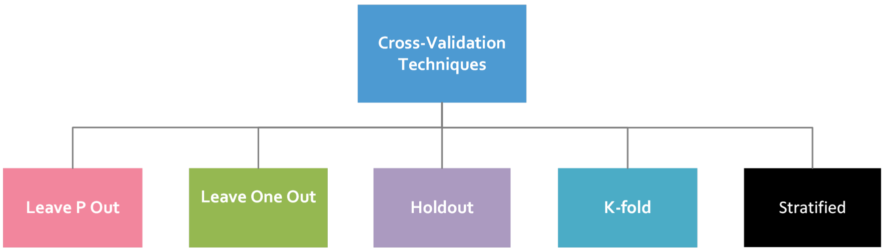


Figure 21: Different Cross-Validation Techniques

2.11.1 Holdout CV

In this method, the Training set is split randomly into two set, one split is used to train the model and other is used as the Validation set to evaluate the model. Generally, the training split is larger than the Validation split. This method is commonly used if the dataset is large and balanced. This is because if the dataset is large, other cross-validation techniques will be more expensive to run;also, if the dataset is large, the model can generalize well using the training set itself and leaving out few entries for validation will not cause huge decline in performance.

2.11.2 K-Fold CV

In this method, the Training set is split into K equal partitions, and then one partition is taken as validation set and the remaining partitions used to build the model. This process is repeated for every partition, ie, the process is repeated K times, each time taking a different partition as the validation set. This method is commonly used if the dataset size is moderate or to boost the performance of the model by training on the entire Training set. However, K-Fold CV is computationally expensive as the model will be trained 5 times with different subset of data.

2.12 Hyperparameter Tuning

Every Machine Learning technique provides choices/parameters while creating model architecture to tune the algorithm to specific use cases. For eg: in Random Forest Classifier (RF) algorithm, the number of weak learners is a hyperparameter, a user can try different values for number of weak learners to see which value provides best performance for the model. Another example is number of layers in an Artificial Neural Network (ANN), user can try different number of layers to tune the performance of the model. The process of trying different values for hyperparameters to find the best model architecture is known as Hyperparameter Tuning. The entire Training set, including the validation set, is trained on the model using best hyperparameters found using Hyperaparameter Tuning process to get the final model. Figure 22 shows different techniques used for the Hyperparameter tuning process.

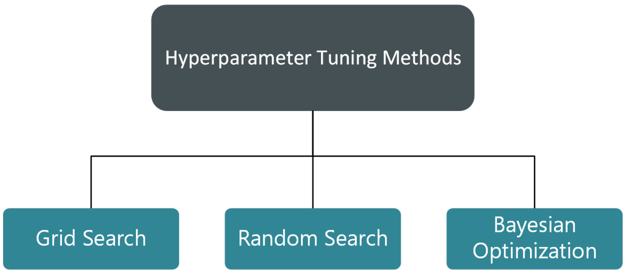


Figure 22: A sample Train,Validation & Test Split

2.12.1 Grid Search

In this method, the model is trained using each possible combination of hyperparameters and the best model is the one which scores the highest value for the chosen metrics on the Validation set. The choices for hyperparameters are discrete; for example, X & Y are two hyperparameters for the model Z, choices for X = [1, 2, 3] & Y=[4, 5, 6], then Grid Search algorithm will train the model on every possible combination of X & Y and will find the best combination of X & Y which provides the highest performance. Grid Search CV is an API in Scikit Learn Libray which uses K-Fold Validation technique to assess the performance of the model while performing Grid Search algorithm for hyperparameter tuning.

2.13 File Format - Parquet

The file format of the dataset has a significant role in the overall model building process. The Primary Dataset used in this project was initially hosted in Google Drive (Cloud Storage) as a csv file and the dataset size was 16GB. Transferring the dataset into Google Collab runtime environment took lot of time and effected overall model building time. Due to this, each iteration of the experiment took more time which was impacting overall progress of the project. In order to overcome this issue, the dataset was converted to Parquet file format which helped to reduced the dataset size to 7GB.

In 2013, Twitter & Cloudera announced a new open-source columnar storage format, Parquet. Instead of storing the millions of records in row wise, the Parquet format stores the data column wise. Due to this, the Parquet format is able to compress the data better than the other file formats. Moreover, the data retrieval in columnar storage is also more efficient for the analytical usecases.

2.14 Summary

In this section, intuitive level explanation of all the techniques and algorithms used in this project was provided. The next section will discuss some of the previous studies conducted on Credit Card Default Prediction task.

3 Literature Review

3.1 Introduction

This section discusses the current techniques used to predict the credit card default. Since the dataset used for these studies differ, a direct comparison of results is not possible. However, an overall comparison of different techniques and its efficiency in predicting the credit card default will be discussed wherever possible.

3.2 Previous Work

(Sayjadah et al. 2018) developed logistic regression, rpart decision tree & Random Forest Classifier models on dataset generated from credit card operations. The dataset contains 30000 records and 24 features. A Correlation Based Feature Selection (CFS) technique was used to reduce the dimensionality of the dataset and 30% of the dataset was used as the test set for evaluating the performance. (Sayjadah et al. 2018) found that the Random Forest Classifier scored highest Area Under the Curve (AUC) metric among the all models.

(Widyadhana & Prastyo 2021) developed Logistic Regressiion, SVM, ANN & Random Forest Classifier models to preduct credit card default on dataset containing 1000 records and 11 features. The dataset contains credit card data from cardholders from the territory of Indonesia;moreover, the authors used Principle Component Analsys (PCA) to do feature selection also. The dataset is split into 70:30 Train/Test set and the AUC metric was used to compare the results. The authors found that the Random Forest Classifier outperformed all other models by far and provided a 80% AUC score.

(Alam et al. 2020) investigated different approaches to solve the credit card default prediction problem with a specific focus. The study experimented the models on 3 different dataset, Taiwan Clients Dataset (30000 records) (Yeh & Lien 2009), Sought German Clients Credit Dataset(1000 Records) & Belgium Clients Credit Dataset (300000 records). The credit card default prediction dataset are inherently imbalanced as only a small fraction of customers default on credit card. The authors employed different data under/over sampling techniques and evaluated performance on multiple credit card default dataset. They found that GBDT classifier when used with KMeans SMOTE provided the best results and the models performed significantly better on balanced datasets compared to imbalanced datasets on all the datasets.

(Faraj et al. 2021) research shows that ensemble methods consistently outperform Neural Networks and other machine learning algorithms in terms of F1 score. (Faraj et al. 2021) uses the same dataset as the (Sayjadah et al. 2018) which has 30,000 records and 24 features. The authors found that XGBoost provided maximum F1 score compared to Neural Networks, Random Forest Classifier and custom ensemble stacking model. Authors also concludes that the performance of XGBoost model did not improve on balanced datasets. (Emil Richard Singh & Sivasankar 2019) also compared the ensemble bagging,boosting techniques on the same data used by (Faraj et al. 2021) and found that the Ensemble Boosting techniques provided highest predictive accuracy.

Study conducted by Yang & Zhang (2018) compares the performance of the SVM, ANN, XGBoost, and LGBM algorithms on the Taiwan Credit Card dataset. The authors used 10-Fold cross-validation to optimze the model and F1-Score metric was used to compare the performance of the models. Similar to the other studies (Faraj et al. 2021) (Emil Richard Singh & Sivasankar 2019), the authors found that the Ensemble Boosting techniques provided the highest performance on the dataset. Among the Ensemble Boosting frameworks, LGBM provided the best F1-Score.

(Hsu et al. 2019) approached the credit card default prediction from a different perspective and proposed a model where dynamic features (time dependent features) were first passed through a Recurrent Neural Network (RNN) network to extract the time dependent features. Then the extracted dynamic features were concatenated with the static features and trained on a Random Forest Classifer. The dataset contained 30,000 samples credit card payment history with 23 features (5 static feature, 18 dynamic features). The authors compared the results of the proposed model with the SVM, Logistic Regression and KNN models and found that proposed method outperformed the others and provided a AUC score of 78%.

A recent study (Wu et al. 2022) developed a graph attention network method, CDGAT, for credit card default prediction. The dataset used for this study was obtained from Industrial and Commercial Bank of China (Macau) Limited (ICBC (Macau)) and contains transaction data between 2015 to 2019. The authors used RNN to encode the customer transaction using and Amount bias Sampling technique to obtain the neighborhood embedding . The proposed CDGAT was compared with the Logistic Regression, SVM, LGBM and ANN models on AUC score. The proposed method performed significantly better than the other models.

(Gao et al. 2021) proposed XGBoost-LSTM method to predict the credit card default. In this approach, LSTM is used to embed the time series transaction data, which is then combined with the basic data of the customer before training the same using XGBoost algorithm. The authors identified that the XGBoost-LSTM performed better than the XGBoost technique on the Recall metric. The dataset used for this study was obtained from a small-scale commercial bank (Gao et al. 2021).

Chen & Zhang (2021) proposed ANN-KMeans SMOTE technique to predict the credit card default. This method uses the KMeans SMOTE algorithm to solve the Dataset Imbalance issue and uses RF to extract the feature importance. The feature importances are then substituted to the initial weights of the ANN network. The authors compared SVM, DT and RF with the proposed model and found that the proposed model performed slightly better than the SVM model on F1-Score. Authors also observes that the increase in performance gained by substituting initial weights with the feature importance obtained from RF is not obvious.

(Lawi & Aziz 2018) conducted a study comparing performance of the models SVM,Least Square SVM (LS-SVM) (Suykens & Vandewalle 1999) and an ensemble model based on LS-SVM on Taiwan Clients Dataset (30000 records) (Yeh & Lien 2009). The results shows that the LS-SVM performed better on Sensitivity metric than the SVM model. The LS-SVM Ensemble did not improve the Sensitivity of the prediction from the LS-SVM model; however, the LS-SVM model provided the best score on Specificity.

3.3 Summary

In conclusion, the ensemble boosting models generally provided better performance than the classic machine learning and deep learning techniques on Credit Card Default prediction problem. The data under/over sampling had mixed performances depending on the dataset, some performed better with under/over sampling and some did not. Out of data under/over sampling techniques, KMeans SMOTE performed better. Some studies used feature selection techniques in the data preprocessing techniques which improved the model efficiency. Most of the studies on the credit card default problem uses Taiwan Clients Dataset (30000 records) (Yeh & Lien 2009) which only contains 30,000 records and the data is from period 2004-2005; however, this dataset does not completely represent today's industry scenario and the challenges where banks have millions of records which can aid the credit card default prediction. Also the consumer behavior might have changed over the time;hence, study of Credit Card Default problem on large scale & recent data has significance in the current landscape. American Express Default Predition dataset (American Express 2022) contains 5 Million records of 400,000 unique customers from the period 2018-19, and exploring the different techniques on such large scale dataset will help to consolidate the understanding gained from these papers.

4 Materials

4.1 Primary Dataset

The primary dataset contains 190 aggregated profile features of 458913 American Express customers at each statement date for 13 months. Features are anonymized and normalized, and fall into the following general categories:

- D_* = Delinquency variables
- S_* = Spend variables
- P_* = Payment variables
- B_* = Balance variables
- R_* = Risk variables

This dataset(American Express 2022) was released as part of the "American Express - Default Prediction" hosted in Kaggle by the American Express team.

4.2 Secondary Dataset

The secondary dataset was derived from primary dataset by applying the below mathematical aggregate operations to the numerical features.

- Minimum
- Maximum
- Mean
- Last Value
- Standard Deviation

Aggregate for the categorical features were taken by the applying below operations.

- Last Value
- Count
- Unique Value Count

The secondary dataset contains 920 features and 458913 records.

4.3 Tools & Software

The primary programming language used for the implementation of this project is Python version 3.7. Data analysis and manipulation is done using Pandas(1.3.5), seaborn(0.11.2) & Dask(2.12.0) packages. Scikit Learn(1.0.2) package is used for create, train & evaluate machine learning models. ANN & GRU models were created using Tensorflow (2.8.2).Google colab was used to train the model in cloud and Github was used as the version control & project management software.

5 Methodology

5.1 Introduction

This section will first provide a brief overview of the overall strategy of the experiments performed as part of this project followed by providing detailed explanation on the data preprocessing techniques used. Then in subsequent sections each experiment/model will be presented along with the model specific explanations & details.

5.2 Overview of Methodology Followed

Figure 23 represents a overview of methodology in general followed for conducting experiments. The dataset was first split into chunks and stored in different files in parquet format to optimize the memory usage. Then, dataset was preprocessed to remove invalid values and encode categorical text variables to numerical values. Followed by data pre-processing, the dataset was split into Training & Test set, this ensures that none of the entries in test set will have an influence in model training and model selection process. Then the dataset was enhanced using oversampling techniques to resolve the class imbalance issue;in addition, feature selection techniques were used to eliminate the features from the dataset which were less important and hence contribute very little to model.

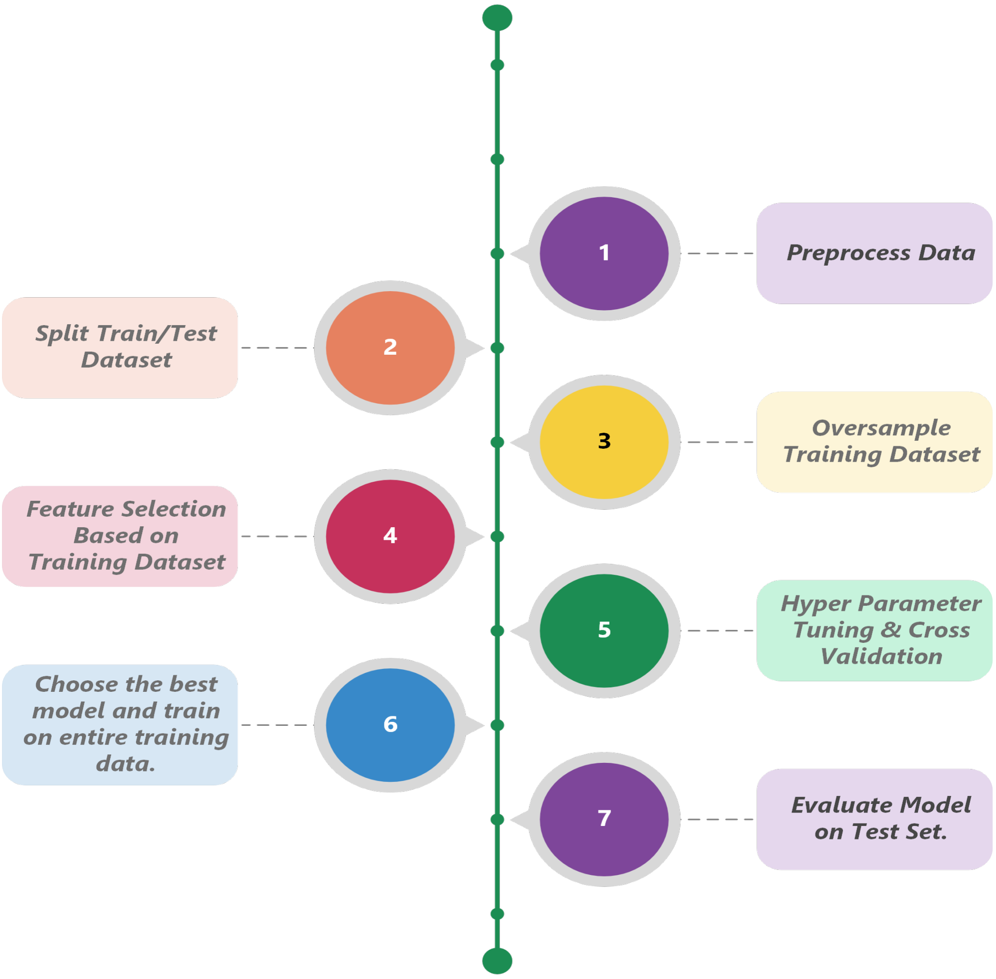

1
Preprocess Data
2
Split Train/Test Dataset
3
Oversample Training Dataset
4
Feature Selection Based on Training Dataset
5
Hyper Parameter Tuning & Cross Validation
6
Choose the best model and train on entire training data.
7
Evaluate Model on Test Set.

Figure 23: Methodology followed for the experiments

After the feature selection, the model was created and then passed through a Hyperparameter tuning pipeline which helps to find the best parameters for the model which would give highest cross validation score. Finally the entire training dataset was trained on the best model found using hyperparameter tuning and the model was evaluated using the test set set aside at the beginning of the experiment.

5.3 Data Preprocessing

This section discusses the common preprocessing techniques used in all experiments conducted as part of the project. The model specific data preporcessing techniques used will be discussed in respective sections describing the model.

5.3.1 Default Values

NaN & NULL values in the dataset was replaced by Zero and if a column contains all values same, it was removed from the dataset. -1 was used as the default value for the categorical variables. The categorical variables were encoded using Ordinal Encoder before passing to the model training pipeline.

5.3.2 Normalization

The primary dataset from American Express is already normalized and all the values lies between zero to ten, hence none of the data normalization techniques were used to preporcess the data.

5.3.3 Handling Memory Issue

Google Colab provides 24 GB of Random Access Memory (RAM) in the virtual environment, though the Primary Dataset is 16 GB, the pandas library was unable to load the complete data into memory due to memory leakage issue in the framework. Dask library, which uses multiple Pandas dataframe under the hood, was used to overcome the memory. Dataframe API in Dask library splits the dataset into multiple chunks and loads each chunk on a need basis only (Lazy Loading), this ensured that the complete 16 GB dataset could be loaded even at a low memory of 4GB.

Additionally, the primary dataset was loaded using Dask framework and split the dataset month wise, ie one file for each month. The month wise files were saved in parquet format which helped to reduce the total size of the dataset from 16GB to 7GB. Similarly the primary dataset was also split customer wise, ie 1-50000 customers data in one file, 50001-100000 customers data in second file etc. These files were later used to build the Secondary Dataset.

5.4 Model 1 - Support Vector Machine (SVM)

The SVM model was created with parameters Regularziation Term = L2 Norm(Squared Error Loss), Alpha = 0.0001, Loss='hinge'(soft-margin), tolerance=0.001. The primary dataset was used to train the model and the model converged after 28 iterations. Early stopping was used to prevent overfitting of the model and 10% of the data from training set used as the validation set. Stochastic gradient descent was used to optimize the objective function, this ensured that even though the dataset contains millions of records, the training is able to proceed and finish in reasonable time. 20% of the Secondary Dataset was used as test set to evaluate the performance of the model.

5.5 Model 2 - Random Forest Classifier

The Random Forest Classifier model uses 100 Decision Trees trained in parallel on the primary dataset. Each decision tree uses a different subset of Primary Dataset with maximum number of records in a database set to 600,000. Gini impurity metric is used to measure the quality of the split while building decision tree. Finally the model predicts the target variable by taking mean of all the predictions from the 100 individual decision trees. 20% of the Secondary Dataset was used as test set to evaluate the performance of the model.

5.6 Model 3 - Gradient Boosting Decision Tree (GBDT)

GBDT model was created using 100 Decision Trees trained sequentially on the primary dataset. Friedman Mean Squared Error (MSE) is used to measure the quality of a split; additionally, model was set to use only 60% of the data for constructing each decision trees to avoid memory leakage issue. 10% of the training set was set for validation purpose; furthermore, the parameters were set to stop the training if the validation score does not improve to avoid overfitting. Loss function for the training was set to Deviance. 20% of the Secondary Dataset was used as test set to evaluate the performance of the model.

5.7 Model 4 - Xtreme Gradient Boosting Machine (XGBoost)

XGBoost model was created using training 100 base learners on the Secondary Dataset and each base learner is constructed using 80% of the training dataset. Instead of using complete features to construct the base learner, parameters were set to use only 60% of features, this helped to eliminate the memory leakage/overflow issues while training. Moreover, L2 regularization parameter was set to 0.9 to reduce the overfitting of the model. 20% of the Secondary Dataset was used as test set to evaluate the performance of the model.

5.8 Model 5 - Light Gradient Boosting Machine (LGBM)

LGBM model was trained on Secondary dataset and the 100 base learners were constructed using the entire features & training set. Tradition Gradient Boosting Decision Trees were used as the boosting type and learning rate was set to 0.1. 20% of the dataset were set aside as the test set for evaluating the model. Maximum depth is not set as to allow trees of any depth.

5.9 Model 6 - Artificial Neural Network (ANN)

Figure 24 depicts the architecture of the custom ANN model developed. The primary dataset was first split into training & test dataset, followed by oversampling the training dataset using KMeans SMOTE to make the percentage of defaulting & non defaulting customers equal. Then the training dataset was trained using the custom ANN model. The first & second layer uses Rectified Linear Unit (ReLU) as the activation function, however the final layer uses Sigmoid as the activation function. Adam optimizer was used to optimze the objective binary cross entropy loss function. The trained model was tested and evaluated on the test set.

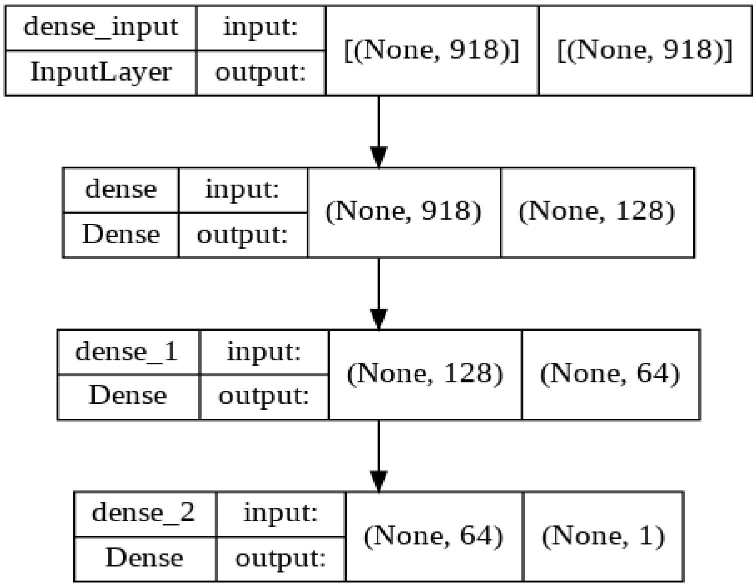


Figure 24: Custom Neural Network Architecture

5.10 Model 7 - Gated Recurrent Unit (GRU)

Figure 25 represents the architecture of the GRU based model for predicting credit card default. The primary dataset is used for training this model;furthermore, the tanh function is used as the activation function and sigmoid is used as the recurrent activation function for the GRU layer. Dropout of 10%, recurrent dropout of 50% added to reduce the overfitting problem. The output of the GRU layer is then fed to dense layer followed by another dense layer with activation sigmoid for making final prediction. Optimizer used is Adam & the loss function is Binary Cross Entropy. A dataset generator was created to provide input to the model in chunks, this helped to eliminate the memory issues while training. The final model contains 49165 trainable parameters.

5.11 Model 8 - Ensemble Stacking Model using ANN + GRU + GBDT

Figure 26 partially depicts the architecture of the custom ensemble stacking model. The primary dataset is first trained using GRU layer followed by a dense layer. In parallel, the secondary dataset is trained using a 2 dense layers. Then the output of these two parallel legs were combined to form the concatenation layer. The output of concatenation layer is then trained using a GBDT model to get the final prediction. Adam optimizer was used to optimze the objective binary cross entropy loss function. Instead of loading complete dataset into memory and train the entire dataset in one go, a dataset generator was written to return chunks of data for training, this helped to eliminate the memory overflow issues.

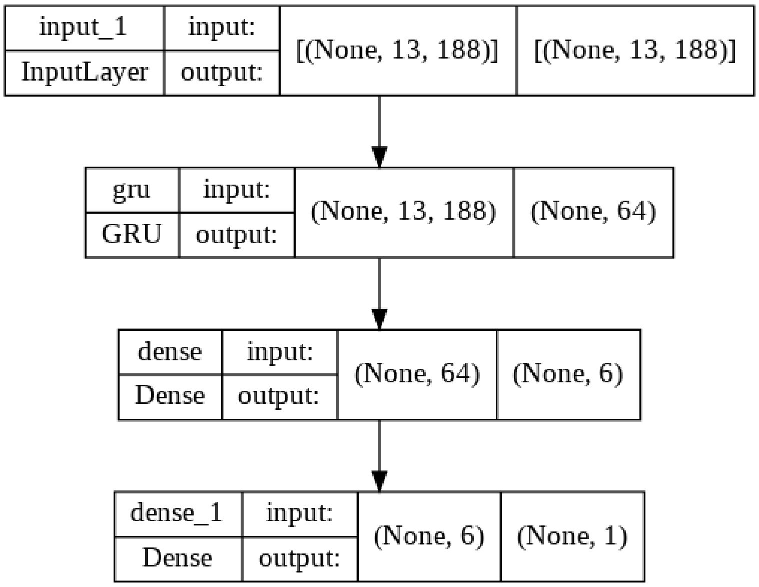


Figure 25: GRU Model Architecture

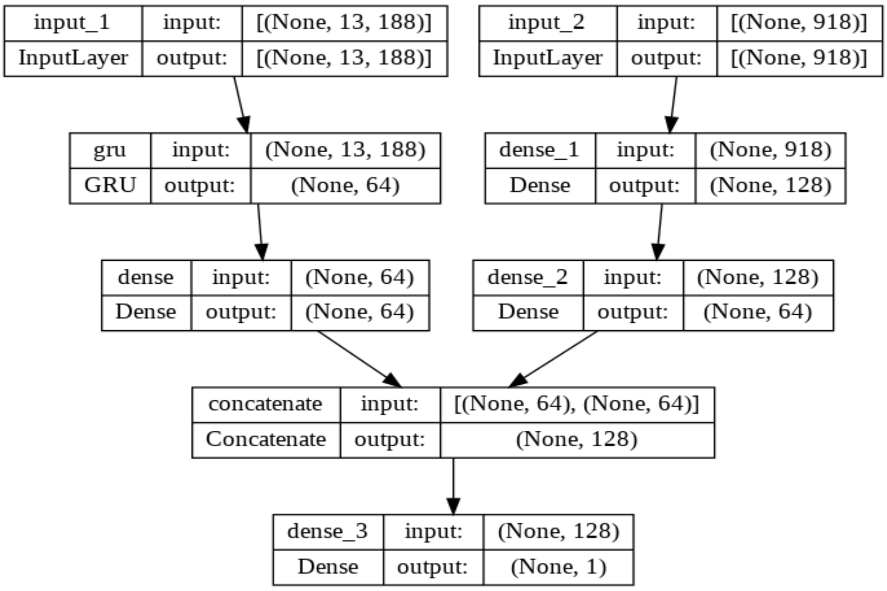


Figure 26: Custom Ensemble Stacking Model Architecture

5.12 Model 9 - Lean LGBM Model

Finally, a lean model was created using LGBM model which peformed on par with the other models but with less resources & data. Firstly the 20% of the dataset was set aside as test set and remaining 80% for training purpose. Training dataset was then oversampled using KMeans SMOTE method which resulted in the number of defaulting customers & non defaulting customers to become equal. Secondly the the training dataset was trained using a simple SVM model to extract the feature importances; in addition, the features with feature importance weight less than the mean of importance of weights were discarded. This helped to reduced the feature count from 920 in the secondary dataset to 279. This modified dataset was then passed through a Grid Search CV pipeline to choose the best parameters for maximum depth & maximum number of leafs for base learner trees, and boosting type. Cross Validation (CV) Recall score was used as the metric to choose the best model. Finally, a LGBM model was created using the best model parameters found using GridSearchCV and trained the same on the complete training dataset. The final model was tested and evaluated on the test set.

5.13 Summary

Firstly, this section presented overall methodology followed for the experiments followed by the data preprocessing techniques used. Handling of invalid feature values, normalization & handling memory leakage and memory overflow issue were discussed in the Data Preprocessing section. Secondly, the 9 different models created to solve the problem were discussed. Initially SVM model & Random Forest Classifier model were discussed followed by more advanced machine learning techniques such as GBDT, XGBoost & LGBM. Then 3 models using deep learning techniques such as Neural Network, GRU and Ensemble Stacking model were presented. Finally a lean model was developed and presented which used less features for training, was computationally less expensive and was explainable model. Before training the final model, the model was passed through a data oversampling pipeline, feature selection pipeline. The model was also tuned using GridSearchCV method to find the optimal hyper parameters. In all the experiments 20% of data was set aside before training for testing & evaluation purposes.

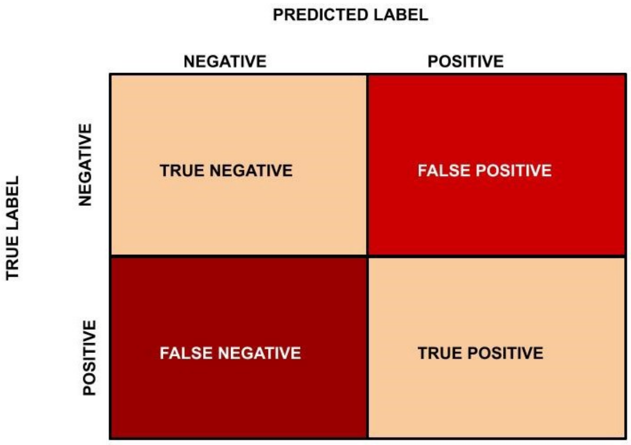
6 Results & Discussions

6.1 Introduction

Firstly this section presents the Metrics used to evaluate the model and the test result of the experiments conducted based on the methodology described in section 5. Secondly the main highlights or achievements of the project is discussed. Finally, limitations of this project as well as the direction for the future works are reviewed.

6.2 Metrics

Figure 27 represents the Confusion Matrix associated with binary classification problem. There are 4 important terms related to confusion matrix, True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). TP represents the predictions where the truth label and the predicted labels are both positive; on the other hand, FP represents the predictions where the predicted label is positive but the truth label is negative. TN represents the predictions where both the predicted label and the truth label are Negative. Predictions where the predicted label is negative and the truth label is positive is called FN.



		PREDICTED LABEL	
		NEGATIVE	POSITIVE
TRUE LABEL	NEGATIVE	TRUE NEGATIVE	FALSE POSITIVE
	POSITIVE	FALSE NEGATIVE	TRUE POSITIVE

Figure 27: Confusion Matrix of A Binary Classification Problem(Harikrishnan N B 2019)

There are multiple metrics which can be used to evaluate the performance of the model based on the confusion matrix, metrics which are relevant to this project is discussed in below sections.

6.2.1 Accuracy

Accuracy is defined as the ratio of total number of correct predictions and total number of samples. However, the Accuracy metric might be misleading on imbalanced dataset;for example, in a dataset where the minority class is only one percent of the overall dataset, 99% accuracy can be achieved by predicting every data point as the majority class. Equation 6 shows the formula for calculating Accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

6.2.2 Recall

Recall represents the ratio of total number of positive predictions and the total number of positive samples. Intuitively, Recall provides an idea of how many positive samples where correctly identified by the model;for example, in a credit card default prediction dataset, recall is how many of the defaulted customers were identified correctly by the model. Equation 7 shows the formula for calculating Recall.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

6.2.3 Precision

Precision is defined as the ratio of correct positive predictions and the total number of positive predictions. Precision provides insight into how many of the predicted positive samples where actually correct.Equation shows the formula for calculating Precision.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

6.2.4 F1-Score

F1-Score combines both Precision & Recall into a single metric; in other words, F1-Score is the harmonic mean of Precision and Recall metrics. F1-Score has shown to be a good metric on the imbalanced datasets classification problems. F1-Score better represents the performance of the model on the dataset compared to the other metrics. Equation 9 shows the formula for calculating the F1-Score.

$$Precision = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{9}$$

6.3 Test Results

Table 1 depicts the result of running the models discussed in section 5 on the test set. 4 metrics, F1 Score, Recall, Accuracy & Precision is used to compare the performance of the models. Though 4 metrics are tracked, Recall & F1 Score are the metric which is used for Hyper Parameter Tuning, Early Stopping and Cross Validation across all experiments. This decision was taken based on observation that Accuracy will not be a good metric as the dataset is imbalanced; furthermore, the F1 score & Recall provides a better indictation on how well the model is able to predict the defaulting clients correctly.

Model	F1 Score	Recall	Accuracy	Precision
SVM	74.66	75.57	87.24	73.78
RF	73.81	72.78	87.13	74.86
GBDT	74.33	74.13	87.25	74.52
XGBoost	80.22	80.11	89.79	80.34
LGBM	81.01	81.11	90.13	80.91
ANN	81.03	81.81	90.09	80.27
GRU	79.81	79.96	90.67	79.65
GRU+ANN+GBDT	80.09	79.71	90.87	80.49
Lean LGBM	80.95	80.68	90.15	81.23

Table 1: Performance Comparison of Various Models.

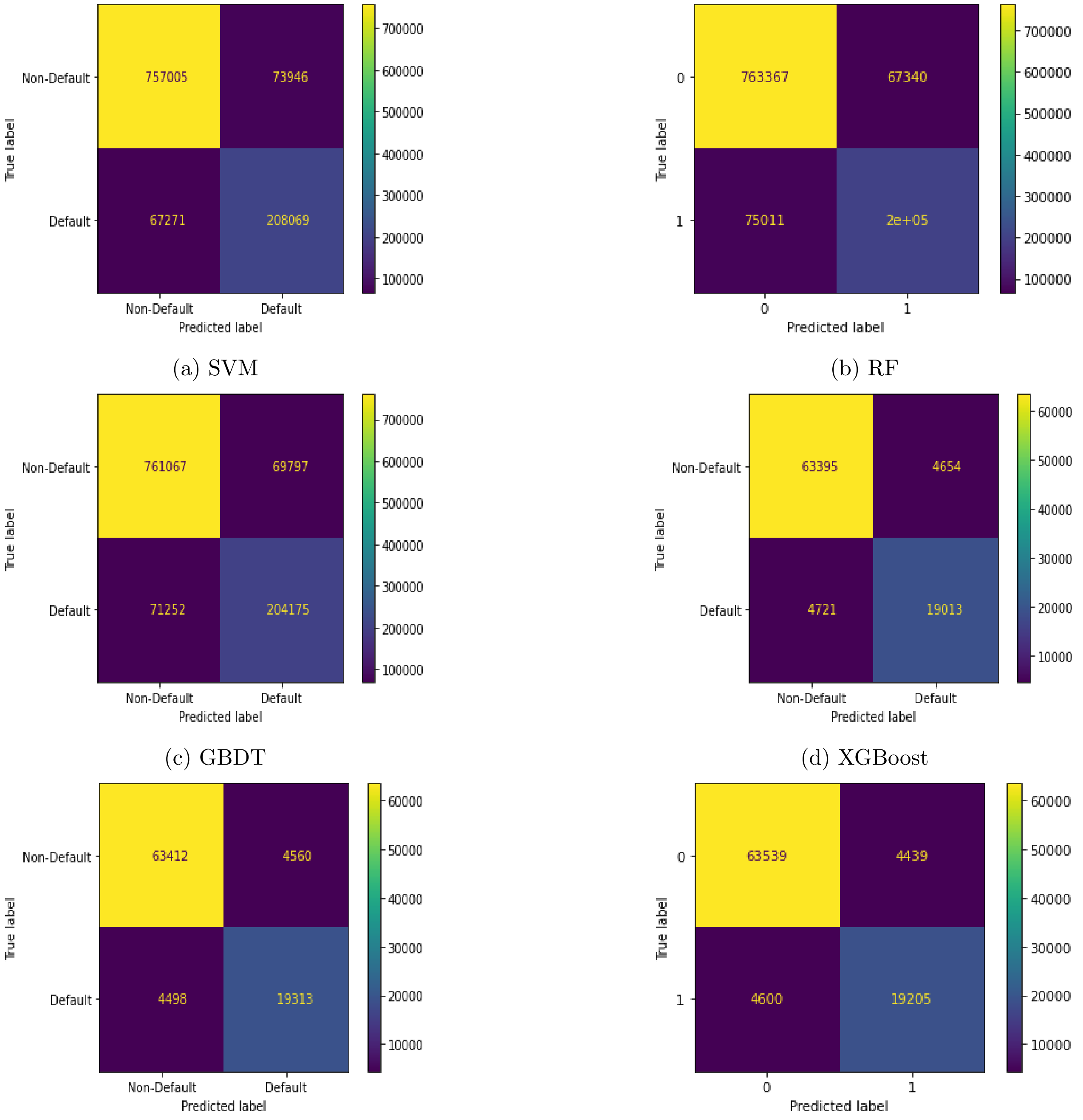
The SVM provided an accuracy of 87.24% & F1 score of 74.66 which was impressive considering the model took very less time to train compared to other models; thus, SVM model was used in Lean LGBM model as part of feature selection pipeline. RF model performed worst on F1 Score, Recall & Accuracy among all the models experimented as part of this project. This reduced performance of RF could be because the algorithm is not able to generalize well with 100 decision trees on a large dataset with high dimensionality. GBDT model provided similar results to SVM model, however, GBDT provided better Precision while SVM provided better Recall scores. Ensemble boosting techniques XGBoost & LGBM performed really well the test set such that F1 score & Recall increased almost by 6% compared to SVM model. LGBM model performed slightly better than XGBoost model in all the metrics and LGBM model took less time to train compared to XGBoost.

ANN model provided the highest F1 Score & Recall among all the models, however, the difference in F1 Score of LGBM and ANN model is only 0.02. ANN model provided better Recall score while LGBM provided better Precision score. GRU+ANN+GBDT model provided best accuracy score of 90.87 among all the models. GRU provided the second best accuracy score;moreover, considering that GRU model has 49,165 trainable parameters while the GRU+ANN+GBDT has 178,945, the GRU model is able to extract the features better than GRU+ANN+GBDT with less parameters. The table 2 shows the trainable parameter count for the deep learning models explored.

Model	Trainable Parameters
ANN	125,953
GRU	49,165
GRU+ANN+GBDT	178,945

Table 2: Trainable parameters comparison of Deep Learning Models

The Lean LGBM model provides a F1 Score of 80.95 which is only 0.08 less than the best performing ANN model; futhermore, the Lean LGBM provided the best Precision score among all the models. Lean LGBM model uses only 1/3rd of the features used by the ANN model and still provides comparable performance. The figure 28 represents the confusion matrix of all the Machine Learning models explored during this project.



(a) SVM

(b) RF

(c) GBDT

(d) XGBoost

(e) LGBM

(f) Lean LGBM

Figure 28: Confusion Matrix of Machine Leaning Models

6.3.1 Feature Importance

The table 3 represents the top 5 features based on the importance from model Lean LGBM. This is another advantage of Lean LGBM model over the ANN model, the Lean LGBM model is explainable. However, the features in the Primary Dataset & Secondary Dataset is anonymized, hence it is not possible to know the exact real life meaning of these features.

Feature	Importance Score
D_39_last	226
P_2_last	214
B_4_last	191
B_3_last	162
B_1_last	144

Table 3: Important Features Extracted Based on Lean LGBM model (Secondary Dataset)

6.4 Achievements

As mentioned in 3, most of the previous work on credit card default prediction was based on dataset containing records less than 100,000. However the Primary Dataset used in this work has around 5 Million records of 400,000 unique customers. Most of the previous works concluded that the Ensemble Boosting classifier such as XGBoost & GBDT performed better than the traditional machine learning algorithms, this is reconfirmed on large dataset with the work of this project. On the other hand, this work helped to identify that the difference in peformances between the Deep Learning Network & Ensemble Boosting Algorithms reduced on large dataset, however it the differences were significant in the smaller datasets. Finally, it was reconfirmed that the models performed better on balanced data and KMeans SMOTE is a suitable candidate for fixing the class imbalance.

6.5 Limitations & Future Work

6.5.1 Limitations

Training using some of the traditional machine learning algorithms, such as Non Linear SVM, K-Nearest Neighbours (KNN) etc, were not possible due to the large dataset size and the related time complexity. Feature selection using Sequential Feature Selection technique was tried to select 50 features from the Secondary Dataset;however, due to the large number of features and the large dataset size, Sequential Feature Selection did not complete even after 20 hours. Thus, due to the time constraint on the project as well as the cost related to training on Google Colab, Sequential Feature Selection was dropped from experiments and instead Select From Model technique were used.

Additionally, though multiple data over/under sampling techniques, such as SVMSMOTE, BorderlineSMOTE, Cluster Centroids etc, were tried, none of them worked as some throw memory overflow issue and others did not complete due to higher time complexity. Since the features of the dataset is anonymized, applying domain knowledge to feature selection process was not possible; moreover, features identified as important in section 6.3.1 can not be made sense in real life as the features are anonymized. Finally, the data types auto-detected by the Pandas framework while loading dataset is not optimal because even though some of the features contains values between 0 - 1 with precision up to 10 decimal points, still Pandas chooses float64 as the datatype by default. This causes the in memory size of the dataset to quickly exceed the available memory and making impossible to try some of the machine learning algorithms.

6.5.2 Future Work

Scikit Learn (Pedregosa et al. 2011) provides limited support for parallel computing, yet this can be explored to make certain computations faster and it might help to overcome some of the computation related limitations mentioned in previous section. Since the data types auto-detected by the Pandas framework while loading dataset is not optimal, one can explore individual features and manually find an optimal datatype of the feature and save this new dataset in parquet format. This will help to reduce the overall size of the dataset and will make loading the dataset to memory much easier.

Attention based Transformer networks recently shown to perform well for time series based classification tasks (Wen et al. 2022), (Lim et al. 2021), (Cholakov & Kolev 2021), this could be explored on the Primary Dataset to extract dynamic features or could be used as part of the Ensemble Stacking model as one of the learners. TabNet (Attentive Interpretable Tabular Learning)(Arik & Pfister 2021) architecture on classification problems with tabular dataset provides promising performance;thus, this architecture could be explored on the Primary/Secondary dataset to check if provides better performance than the models explored in this project.

In order to overcome some of the challenges related to feature selection mentioned in the previous section, one can explore the idea of stacking the different feature selection techniques in sequential manner, so that less expensive feature selection technique is applied first and then more expensive techniques applied later in the feature selection pipeline.

6.6 Summary

In summary, this section initially discussed the evaluation results of different model on the test set. It was identified that the ANN network scored the highest on F1 Score & Recall metrics while GRU+ANN+GBDT scored the highest score on accuracy metric. The Lean LGBM model performed on par with the best scoring models while using less features;also, the Lean LGBM scored highest on Precision metrics. Top 5 important features were identified from the Lean LGBM model result and the same was presented. Then the achievements of the experiments, such as reconfirming understanding from prior studies on huge industry scale dataset and identifying that the performance of Ensemble Boosting models is on par with the deep learning architecture performance on large datasets. Finally, the limitations related to the data sampling, feature selection, huge dataset size were discussed along with proposing future works that could be conducted on the dataset & models.

7 Conclusion

The objective of this project was to explore different machine learning algorithms & deep learning architectures on the American Express default prediction dataset(American Express 2022) to predict if a customer will default on the payment in the future. 8 different models (5 Machine Learning & 3 Deep Learning models) were explored in this project;subsequently, it was identified that Artificial Neural Network (ANN) networks performed best in terms of F1 Score metrics. However, ANN model only performed slightly better then the best performing machine learning model, LGBM;more importantly, the increase in performance was almost negligible considering the complexity of deep learning networks to traditional machine learning models and the non-explainability of ANN models.Thus, we can conclude that the LGBM is a better choice for credit card default prediction in a industry environment, this is because the explainability of Machine Learning model could become useful for the organizations which require to provide explanations regarding the model to Regulatory Authorities.

Based on the above conclusion, an improved & efficient model, Lean LGBM, was proposed which uses 1/3rd of the features (hence less dataset size) of the Secondary Dataset to provide performance on par with the LGBM & ANN models. This Lean LGBM model uses Feature Selection Techniques, Data Oversampling Technique(KMeans-SMOTE) and Hyper Parameter tuning to obtain high performance with minimal computations.

In conclusion, as reported in various studies (Alam et al. 2020), (Faraj et al. 2021), (Emil Richard Singh & Sivasankar 2019), the Ensemble Boosting techniques such as XGBoost & LGBM provides best performance in terms of F1 Score & computation for credit card default prediction problems.

References

Alam, T. M., Shaukat, K., Hameed, I. A., Luo, S., Sarwar, M. U., Shabbir, S., Li, J. & Khushi, M. (2020), ‘An investigation of credit card default prediction in the imbalanced datasets’, *IEEE Access* **8**, 201173–201198.

American Express (2022), ‘American Express Default Precition Dataset’, Available: `https://www.kaggle.com/competitions/amex-default-prediction/data`.

Arik, S. Ö. & Pfister, T. (2021), Tabnet: Attentive interpretable tabular learning, *in* ‘Proceedings of the AAAI Conference on Artificial Intelligence’, Vol. 35, pp. 6679–6687.

Beyan, C. (2015), ‘Detection of unusual fish trajectories from underwater videos’.

Board of Governors of the Federal Reserve System (US) (2022), ‘Delinquency Rate on Credit Card Loans, All Commercial Banks [DRCCLACBS]’, Available: `https://fred.stlouisfed.org/series/DRCCLACBS`. Data retrieved from FRED, Federal Reserve Bank of St. Louis;.

Bre, F., Gimenez, J. M. & Fachinotti, V. D. (2018), ‘Prediction of wind pressure coefficients on building surfaces using artificial neural networks’, *Energy and Buildings* **158**, 1429–1441.

Breiman, L. (2001), ‘Random forests’, *Machine learning* **45**(1), 5–32.

Chang, C.-C. & Lin, C.-J. (2011), ‘Libsvm: a library for support vector machines’, *ACM transactions on intelligent systems and technology (TIST)* **2**(3), 1–27.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), ‘Smote: synthetic minority over-sampling technique’, *Journal of artificial intelligence research* **16**, 321–357.

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* ‘Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining’, pp. 785–794.

Chen, Y. & Zhang, R. (2021), ‘Research on credit card default prediction based on k-means smote and bp neural network’, *Complexity* **2021**.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), ‘Learning phrase representations using rnn encoder-decoder for statistical machine translation’, *arXiv preprint arXiv:1406.1078* .

Cholakov, R. & Kolev, T. (2021), ‘Transformers predicting the future. applying attention in next-frame and time series forecasting’, *arXiv preprint arXiv:2108.08224* .

Deng, H., Zhou, Y., Wang, L. & Zhang, C. (2021), ‘Ensemble learning for the early prediction of neonatal jaundice with genetic features’, *BMC medical informatics and decision making* **21**(1), 1–11.

Divina, F., Gilson, A., Goméz-Vela, F., García Torres, M. & Torres, J. F. (2018), ‘Stacking ensemble learning for short-term electricity consumption forecasting’, *Energies* **11**(4), 949.

Emil Richard Singh, B. & Sivasankar, E. (2019), Enhancing prediction accuracy of default of credit using ensemble techniques, *in* ‘First International Conference on Artificial Intelligence and Cognitive Computing’, Springer, pp. 427–436.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R. & Lin, C.-J. (2008), ‘Liblinear: A library for large linear classification’, *the Journal of machine Learning research* **9**, 1871–1874.

Faraj, A. A., Mahmud, D. A. & Rashid, B. N. (2021), ‘Comparison of different ensemble methods in credit card default prediction’, *UHD Journal of Science and Technology* **5**(2), 20–25.

Gao, J., Sun, W. & Sui, X. (2021), ‘Research on default prediction for credit card users based on xgboost-lstm model’, *Discrete Dynamics in Nature and Society* **2021**.

Ghedira, H. & Bernier, M. (2004), The effect of some internal neural network parameters on sar texture classification performance, *in* 'IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium', Vol. 6, IEEE, pp. 3845–3848.

Harikrishnan N B (2019), 'Confusion Matrix', Available:
`https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd`. Last visted on 23-09-2022.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hsu, T.-C., Liou, S.-T., Wang, Y.-P., Huang, Y.-S. et al. (2019), Enhanced recurrent neural network for combining static and dynamic features for credit card default prediction, *in* 'ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)', IEEE, pp. 1572–1576.

IBM (2022*a*), 'What are Neural Networks', Available:
`https://www.ibm.com/ae-en/cloud/learn/neural-networks`. Last visted on 23-09-2022.

IBM (2022*b*), 'What are recurrent neural networks?', Available:
`https://www.ibm.com/cloud/learn/recurrent-neural-networks`. Last visted on 23-09-2022.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y. (2017), 'Lightgbm: A highly efficient gradient boosting decision tree', *Advances in neural information processing systems* **30**.

Last, F., Douzas, G. & Bacao, F. (2017), 'Oversampling for imbalanced learning based on k-means and smote', *arXiv preprint arXiv:1711.00837* .

Lawi, A. & Aziz, F. (2018), Classification of credit card default clients using ls-svm ensemble, *in* '2018 Third International Conference on Informatics and Computing (ICIC)', IEEE, pp. 1–4.

Lim, B., Arık, S. Ö., Loeff, N. & Pfister, T. (2021), 'Temporal fusion transformers for interpretable multi-horizon time series forecasting', *International Journal of Forecasting* **37**(4), 1748–1764.

Machado, M. R., Karray, S. & de Sousa, I. T. (2019), Lightgbm: An effective decision tree gradient boosting method to predict customer loyalty in the finance industry, *in* '2019 14th International Conference on Computer Science & Education (ICCSE)', IEEE, pp. 1111–1116.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.

Rani, A., Kumar, N., Kumar, J. & Sinha, N. K. (2022), Machine learning for soil moisture assessment, *in* 'Deep Learning for Sustainable Agriculture', Elsevier, pp. 143–168.

Rezazadeh, A. (2020), 'A generalized flow for b2b sales predictive modeling: An azure machine-learning approach', *Forecasting* **2**(3), 267–283.

Sagi, O. & Rokach, L. (2018), 'Ensemble learning: A survey', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **8**(4), e1249.

Sapountzoglou, N., Lago, J. & Raison, B. (2020), 'Fault diagnosis in low voltage smart distribution grids using gradient boosting trees', *Electric Power Systems Research* **182**, 106254.

Sayjadah, Y., Hashem, I. A. T., Alotaibi, F. & Kasmiran, K. A. (2018), Credit card default prediction using machine learning techniques, *in* '2018 Fourth International Conference on Advances in Computing, Communication & Automation (ICACCA)', IEEE, pp. 1–4.

Schubach, M., Re, M., Robinson, P. & Valentini, G. (2017), ‘Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants’, *Scientific Reports* **7**.

Simeon Kostadinov (2017), ‘Understanding GRU Networks’, Available: `https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be`. Last visted on 23-09-2022.

Suykens, J. A. & Vandewalle, J. (1999), ‘Least squares support vector machine classifiers’, *Neural processing letters* **9**(3), 293–300.

Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J. & Sun, L. (2022), ‘Transformers in time series: A survey’, *arXiv preprint arXiv:2202.07125* .

Widyadhana, K. N. & Prastyo, D. D. (2021), ‘Credit card default prediction using machine learning’.

Wu, J., Zhao, X., Yuan, H. & Si, Y.-W. (2022), ‘Cdgat: a graph attention network method for credit card defaulters prediction’, *Applied Intelligence* pp. 1–15.

Yang, S. & Zhang, H. (2018), ‘Comparison of several data mining methods in credit card default prediction’, *Intelligent Information Management* **10**(5), 115–122.

Yang, X., Wang, Y., Byrne, R., Schneider, G. & Yang, S. (2019), ‘Concepts of artificial intelligence for computer-assisted drug discovery’, *Chemical reviews* **119**(18), 10520–10594.

Yeh, I.-C. & Lien, C.-h. (2009), ‘The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients’, *Expert systems with applications* **36**(2), 2473–2480.

8 Appendix One: Code

8.1 Directory Structure

8.2 Running the Provided Code