Credit card default prediction

Amith Nair, BTech

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Dr. Bahman Honari

August 2022

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Amith Nair

August 19, 2022

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Amith Nair

August 19, 2022

Credit card default prediction

Amith Nair, Master of Science in Computer Science
University of Dublin, Trinity College, 2022

Supervisor: Dr. Bahman Honari

Around the world, credit cards have become one of the most common payment methods. Customers can now pay at a restaurant, movie theatre, or grocery store with a simple tap. Banks reward the use of credit cards by giving points for each unit of money spent. However, there is no guarantee to the banks that credit user will pay back their debts on time. This puts a huge amount of risk on the banks. The research tries to distinguish good customers from bad customers by analyzing the historical data of a customer. Two main approaches are used for this purpose. The first approach focuses on looking at the aggregated information of the customer while the second approach looks into the trends present in the customer's historical information. Both these approaches provide the probability of a customer defaulting. Later the predictions obtained from both these approaches are averaged by assigning weights to each approach. The research has used models like random forest and recurrent neural networks.

Acknowledgments

I would like to thank Dr. Bahman Honari for his continued guidance during the past few months. The feedback and insights that he provided were invaluable.

I would also like to thank my Mom and Dad for always being my pillars of support.

Amith Nair

University of Dublin, Trinity College
August 2022

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modern financial systems have simplified the use of money for the customer without the need for carrying lots of cash. In the older days, a customer would have to carry cash for his/her every need. This brought along with it issues like security and also the need to exchange cash for the required change. All of these issues have been solved with the advent of using a credit card. A customer now can just conveniently tap and pay the required amount to any vendor be it a restaurant, grocery shop, or a theatre. In addition to these, banks encourage the use of credit cards by giving reward points and cashback based on the usage of the cards. All of these benefits have led to the widespread use of credit cards around the world. 28% of all payments in the world are made through credit cards.

A credit card service provided by a bank is much similar to the concept of loans. The bank gives credit to the customers by giving loans up to a certain limit. The loan is given with the expectation that it will be paid back within a given period of time. While giving these credits to customers, financial institutions like banks are put at a lot of risks. The bank carries the risk of losing money in case the customer does not pay back the amount given to them.

What is credit card default?

Usually a customer is expected to pay back the money borrowed using a credit card within a month. If at least a minimum set amount is not paid within the due date, the bank declares the account as delinquent which means that the customer has broken the terms of their contract. A late fee is then added to the original loan amount. If the customer still does not pay back the amount usually within 6 months after the expected due date, the customer is said to have defaulted. This affects the credit score of the customer and in most cases the loan amount is written off by the bank. This leads to the bank ending

up in a loss.

In 2021, the average credit card debt for a customer was around $ 5,221. With each year the number of credit card users keep on increasing and along with it the credit card defaults also increase, thereby increasing the losses of banks.

In order to reduce these losses, it becomes necessary to identify the different types of customers they cater. By distinguishing a good customer from a bad customer, banks can identify the risk associated with giving credits to each customer and thereby set limits to reduce their losses. This classification of good customers from bad customers can be done by using data analytics and machine learning on the historical data of a customer.

Why do customers default?

The common expectation is that a customer defaults because of some present financial crisis they might be in. However this is not always true.

- Gerardi et al. (2018) showed that there is a strategic motive in about 38% of the defaults. These customers default even if they have the ability to pay off the their credit loans. Majority of these strategic defaulters are driven by emotions. White (2010) suggests that they tend to be anxious and depressed about their financial futures and angry at their lenders and the government for refusing to help them out.

- Gerardi et al. (2018) also discovered that household-level income shocks, such as those caused by unemployment and disability, play a significant influence. Unemployment and disability created in situations like the COVID 19 pandemic have huge effects on ability of customers to pay back their credit loans.

Thus credit card default can be predicted by analyzing the following:

- Aggregated historical data of a customer which gives an idea of identifying if a customer is strategically trying to default.

- Looking at trends in the payment history of a customer with time which gives insights of whether the customer is going through a financial crisis.

This research focuses on credit default predictions by focusing on the above factors. Generally only a small percent of customer do actually default. Hence the data used for modelling credit card defaults are mostly skewed. The research thus also looks into various ways to address the data imbalance issue while modelling the defaults.

1.0.1 Dissertation Structure

Chapter 2 looks into some of the previous papers written on tackling the problems specified in introduction. Chapter 3 provides an in-depth understanding of the various models used in this research. Chapter 4 contains the methodologies that were followed right from cleaning the data to getting predictions. Chapter 5 contains a comparison of the different models based on various evaluation metrics. The final chapter 6 concludes the dissertation along with discussion of some future work that could be done.

Chapter 2

Literature Review

The literature review is conducted by breaking down the research problem into smaller sub-problems and doing extensive research for each sub-problem.

2.1 Common Models used for predicting credit defaults

Venkatesh and Jacob (2016) proposed the use of simple machine learning models like logistic regression, naive Bayes and random forest to predict if the customers will default or not. They found that the Random forest model had the best performance among all the models they tried. However, they trained their models on a lower number of features. and considered features like the payment history, marital status, age and education of a person for their modelling. They also only considered looking into payment history as a singular value rather than trying to identify patterns\trends present in it.

Zhou et al. (2019) proposed the usage of different ensemble learning models for prediction of default in the P2P market. The data they used contained 1138 feature variables and 15000 records. They performed feature selection using regularisation methods to reduce the number of features in the data. They then evaluated and compared the performance of various models. Metrics like area under the curve and F1-score were taken into consideration for the comparisons. Although the results achieved by this method was quite promising, there were no measures taken to address the problem of imbalance in the data.

Perera (2019) used the J48 Decision Tree algorithm to predict the credit risk of leasing customers in Sri Lanka. The analysis was performed on a data containing 24 columns of 8235 customers. The evaluation metrics used for this analysis were AUC score and kappa coefficient.

All of these research have a common conclusion that a decision tree model seems to have the best performance for predicting credit defaults.

2.1.1 Dimensional Reduction Methods

Financial data used for credit analysis generally contains a lot of feature variables.The data used by Zhou et al. (2019) contained 1138 columns. Some of these features might not be relevant to our research question. With increase in the dimensions, relevant data becomes sparse and thus the amount of data required to build a good model becomes quite high. Hence it becomes important to reduce the dimension of the data.

There are two common methods used for dimensional reduction which are Principal Component Analysis and Non matrix factorization. To understand Principal Component analysis, it is important to understand the concept of eigen vectors. In a linear transformation, a matrix A can be decomposed into its eigen vector v and eigen value *lambda* as follows:

$$A.v = \lambda.v$$

Here λ is a scalar value and the equation suggests that eigen vector is a vector which changes by a value of λ when a linear transformation by A is performed on it. Principal component analysis(PCA) transforms the data from higher dimension space to a low dimension space in such a way that maximum variance is captured by the data in the low dimension space. Mathematically, it involves decomposing the covariance matrix of the data into its eigen vectors and eigen values. The eigen vectors that correspond to the maximum eigen values are then selected. PCA was first coined by the Pearson (1901) and then later expanded by Hotelling (1933).

Non negative matrix factorization breaks down the input matrix A into two non negative matrices X and Y, such that the product of X and Y is approximately equal to A.

$$A \approx X.Y$$

Non negative matrix factorization was initially used in chemometrics and was first used by Lawton and Sylvestre (1971).

Vikram et al. (2019) performed a performance evaluation of these dimensional reduction techniques by varying the workloads. They performed dimensional reduction on various standard datasets like MNIST (Deng (2012)) and evaluated the time taken and mean square error(MSE) for each technique. It was observed that PCA was much faster

than Non negative matrix factorization and also gave better MSE scores as the data increased.

2.2 Identifying temporal relationships in the data

Temporal relationships refer to the relation between events occurring over time. Credit default of a customer can be modelled as a function of their financial habits. These financial habits can be observed by looking at temporal relationships in bill payments and other factors. Although temporal relationships are widely used in the domain of Natural language processing and time series forecasting, not much work has been done to use these in credit defaults.

Neural networks are the most common models used to classify temporal relations in the data. Neural networks consists of various nodes that are connected to each other. Neural networks tries to mimic how a human brain works. Similar to how neurons fire in the human brain, the nodes in the neural network sends a signal to its corresponding nodes when the input received to it crosses a certain threshold. A detailed explanation of neural networks is provided in the later section.

Do and Jeong (2016) proposed a neural network using convolution to identify temporary relations in a sentence. The sentence was broken down to words and each word was encoded by a unique number. In the convolution layer, a filter of size n runs through the sequence of data and performs convolution with the data in the sequence. This helps in extracting key features from the sequence of data. However the convolution neural network was not able to find relationships between the data and thus gave poor performance.

Lawrence et al. (2000) suggested using a recurrent neural network to find temporal relations in a sequence of data. Recurrent neural network is type of neural network which keeps a memory of the previous data in the sequence by maintaining its state. Steve demonstrated the application of recurrent neural networks to identify temporal relations in sentences for classifying it as either grammatically correct or wrong. It was observed that the recurrent neural networks network was able to learn the grammar and proved to be quite promising.

Another research conducted by **?** involved using a recurrent neural network network to identify anomalies in stock markets by studying the movement of stock prices. The performance of the network was compared with a convolutional neural network. The recurrent neural network proved to be much better than the convolutional neural networks.

2.3 Working with Imbalanced data

Imbalanced data can lead to the model being biased towards the majority class. In order to build a robust model, the imbalance in the data needs to be handled. The most common methods for handling imbalanced data are re-sampling methods and cost sensitive classifiers.

Random under-sampling involves dropping records of the class with majority till the data becomes balanced. However this can lead to information loss. Mani and Zhang (2003) proposed an updated random under sampling technique called Near miss where the records of the majority classes were undersampled based on the distance between the point of a majority class and a minority class. It was observed that using this method of undersampling led to increase in the prediction performance of the minority classes.

Random over sampling is another technique in which the records of the minority class are randomly duplicated till the data becomes balanced. However this can lead to over-fitting of the model being trained because of the huge amount of duplicate records in the data. Synthetic minority oversampling technique (SMOTE) is an improvement to the random over-sampling technique proposed by Chawla et al. (2002). Instead of randomly duplicating the records of the minority class, this method created synthetic records of the minority class to balance the data. For any record of the minority class, it calculates the nearest neighbour. It then multiplies the difference between this two points by a factor between 0 and 1. This value is then added to the initial record to create a synthetic record. This process is repeated till the data becomes balanced.

$$x_{synthetic} = x + (x - x_{neighbor}) * \beta$$

Here x is the record of the minority class, $x_{synthetic}$ is the new synthetic record, $x_{neighbor}$ is the nearest neighbour of x and β is the multiplying factor. However this method was seen to over amplify the data containing no information and also noise leading to reduction in the model accuracy.

Cost sensitive classifiers as demonstrated by Pazzani et al. (1994) are another method to handle data imbalance. In this process, rather than trying to balance the data, it penalizes the miss classification of minority class records during the evaluation phase. The penalty given to a particular class is generally in inverse proportion to the frequency of that class in the data.

Technique	**Pro**	**Con**
Random Under Sampling	Data Balance achieved	Information Loss
Near Miss under sampling	Prediction of minority classes improved	Accuracy reduced
Random Over Sampling	Data Balance achieved	Causes over-fitting
SMOTE over sampling	Data Balance achieved with no overfitting	Amplifies noise
Cost sensitive classifier	Improved representation of the minority class with almost no effect on accuracy	Less effective in achieving data balance

Table 2.1: Summary of methods to handle data imbalance

Chapter 3

Background

3.1 Decision Tree

A Decision Tree is a tree like model in which the leaf nodes represent the various output classes. Figure 3.1 shows the visualization of a Decision Tree. The top node is called the root node and every node between the root node and the leaf node is called as Decision Node. The classification rules used in a Decision Tree is represented by the path from the root node to the leaf node. At each Decision node, the tree splits into multiple nodes based on the conditions set. The conditions are set based on the association of the feature variables in the data to the target variable.

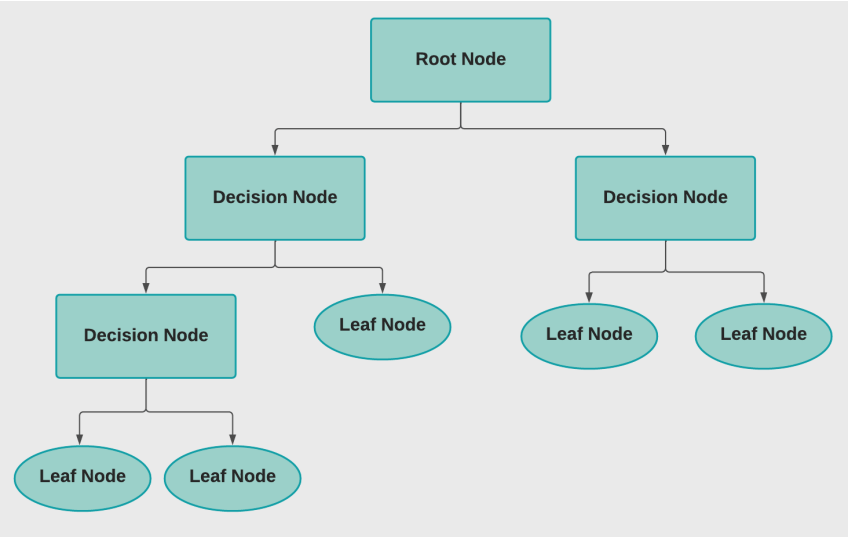

Figure 3.1: Visualization of a Decision Tree Nik (2022)

Levels in the Decision Tree:

The maximum number of splits that a Decision Tree makes while going from the root node to the leaf node is called as levels in a Decision Tree. In figure 3.1, the number of levels is 3. The number of levels in a decision tree can affect it's accuracy. If the number of levels are too less, then lesser number of input variables are being considered and thus the accuracy is less. On then other hand, if the number of levels is too high, then a lot of input variables are being considered and this decision tree might not be able to generalize well on data it has not seen before. The accuracy therefore decreases. Ideally the number of levels should be somewhere in between to maximize accuracy.

To identify the association between feature variables and the target variables, factors like Entropy, Information Gain, and Gini Index are taken into consideration.

Entropy:

Entropy is defined as the amount of information held by a variable in the data. For a data X with P(x) being the probability of event x happening, the entropy $H(X)$ is defined as:

$$H(X) = \sum_{x} P(x) log \frac{1}{P(x)}$$

$$H(X) = -\sum_{x} P(x) log P(x)$$

$$H(X) = -E[log P(X)]$$

Thus a lower value of entropy specifies that the data has less information and thus less surprise associated with the target variable.

Information Gain:

Information gain is used to identify which feature in the data is able to provide maximum information about the target. Considering Y as a target variable and X being a feature variable, the information gain obtained by adding X to predict Y is given as:

$$Information\ Gain(X) = H(Y) - H(Y|X)$$

While building a decision tree, the feature variable which gives the maximum information gain is used at the root node. For the next decision node, the next variable providing maximum information on top of the information already gained by the first variable is used.

Gini Index:

Gini Index is a measure of impurity in the data. For a variable Y with probability function $P(Y)$, the gini index is given as:

$$Gini\ index = 1 - \sum_{y}[P(y)]^2$$

The impurity in the data is maximum when the probability of every event occurring in the data is equal. Thus the Gini Index is maximum in this case. The Gini Index is minimum (0) when the probability of an event is 1. In this case, the data is said to be pure.

For building a decision tree, the feature variable which is able to split the data into subsets with minimum impurity is considered first. This feature variable is used in the root node. The Gini Index of the resultant splits is given as:

$$Gini\ index(X_k) = \sum_{i=1}^{m} \frac{n_i}{n} Gini(Y|X_k)$$

Here X_k is the feature variable which splits the data in children with minimum impurity

The children of the splits are then further split based on the next variable which results in maximum reduction of the impurity in its children. Thus the goal is to create children with minimum overall impurity.

3.1.1 Random Forest:

Traditional Decision Trees suffer from the problem of over-fitting where the model starts to remember the training data. Such models are not able to generalize well on data that it has not seen before. A solution to this problem is ensemble learning. Ensembles consists of multiple Decision trees which are trained independently from each other. The individual decision trees outputs a target variable and the final output is decided on the class which receives the most votes. Inspired by the works of Amit and Geman (1997), Breiman (2001) developed an ensemble method called Random Forest which not only performed bagging but also selected random features from the data for individual trees. This method was seen to considerably reduce overfitting in the models.

Bagging:

In bagging, the individual decision trees are trained on uniform samples taken from the data with replacement. Example: If the data consists of values (1, 2, 3, 4, 5, 6), then one of the samples could be (1, 1, 2, 3, 4, 4). As can be seen the numbers 1 and 4 are repeated

twice. Since individual trees are training on different subsets of data, the variance of the predictions is reduced and thus prevents over-fitting.

Feature Randomness:

The individual trees in Random Forest are trained on a different set of feature variables from the data. This increases the amount of variation in the model making each of the trees more independent. By supplying different columns to each of the individual decision trees, the model also becomes immune to missing data. If there are missing values in the data, the decision tree might only consider records which contain values for the classification. However in random forest, since for some models, the features containing missing values are not considered, these models can use the records which had missing values. This makes Random forest models more robust.

3.2 Principal Component Analysis

Principal Component Analysis is a technique used for reducing the dimension of the data. It is used in exploratory data analysis when there is a huge amount of columns present in the data. In such cases, visualizing the data in a 2-dimension or 3-dimension graph becomes extremely difficult.

3.2.1 Key Definitions:

- **Vector Space:** A vector space is a set of vectors which is multiplied by a scalar value and added together.
- **Basis:** A basis of a vector space is a set of vectors which can represent any element in the vector space as a linear combination of the elements of the basis. The coefficient of each of these elements in this combination is called as a component.
- **Change of Basis:** A change of basis is referred to as the conversion of representation of the vector space from a particular basis to another basis.

  $$x_{new} = C * x_{old}$$

  Here x_{old} represents the vector represented in the older basis.x_{new} represents the vector represented in the new basis and C represents the change of basis matrix.
- **Principal Component:** In a co-ordinate space, the principal component is a set of unit vectors where the i^{th} unit vector is in a direction that minimizes the mean square error within the data. Each of the unit vectors are orthogonal to the remaining unit vectors.

The process of obtaining the principal components while performing a change of basis is called as Principal Component Analysis. The amount of information contained in a principal component decreases with each component. This means that the first principal component contains the maximum information and the last principal component contains the least information. Usually while performing dimension reduction, only the first few principal components are taken into consideration. However this can lead to information loss.

In mathematical terms, the first principal component captures the maximum variance in the data while the last principal component captures the lowest variance of the data.

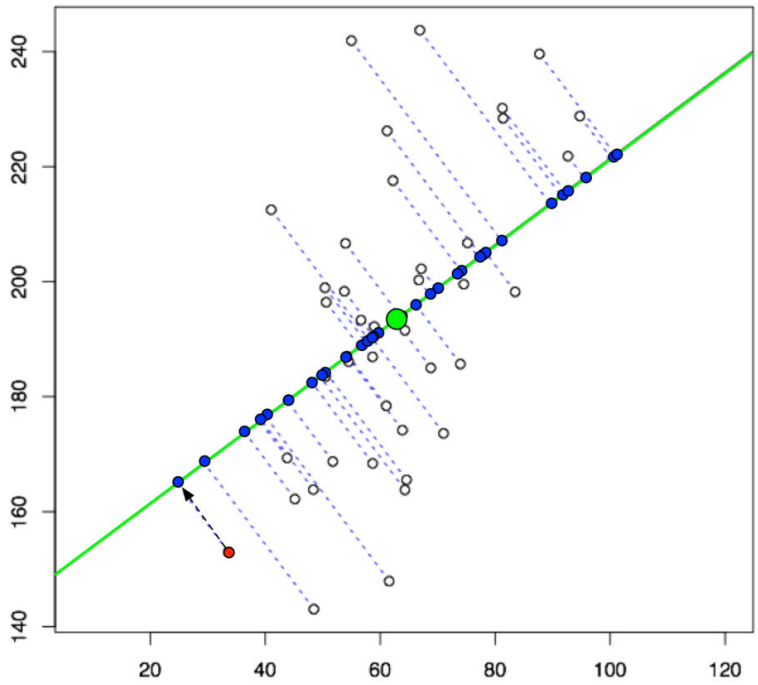

Figure 3.2: Graphical Representation of obtaining the first component of a 2 dimensional data. The green line represents the first component Seb et al. (2022)

3.2.2 Calculating the principal components:

Considering a data matrix X_{old} of p columns that which undergoes a change in basis to X_{new}. The transformation is performed by a change of basis matrix $C = (c_1, c_2....c_p)$.

Calculating the first Principal Component:

As mentioned in the above text, the first principal component captures the maximum variance in the data. In figure 3.2, the maximum variance is in the direction of the green line and hence the first principal component is in this direction as well.

The first weight vector c_1 which maps the data into the first principal component can then be given as:

$$c_1 = arg\ max_{||c||=1} \sum (x^2_{new_1})$$

$$c_1 = arg\ max_{||c||=1} \sum (x_{old}.c)^2$$

In matrix form, it can be written as follows:

$$c_1 = arg\ max_{||c||=1} \sum \{c^T X_{old}^T X_o l dc\}$$

Since c_1 is a unit vector:

$$c_1 = arg\ max \sum \frac{c^T X_{old}^T X_{old} w}{c^T c}$$

Calculating the remaining Principal Components:

The k_{th} component is obtained by subtracting first $k-1$ component from X_{old}.

$$\hat{X}_{old} = X_{old} - \sum_{i=1}^{k-1} X_{old} c_{(s)} c_{(s)}^T$$

The weight vector c_k is then obtained by maximizing the variance for this matrix

$$c_k = arg\ max \sum \frac{c^T \hat{X}_{old}^T \hat{X}_{old} c}{c^T c}$$

3.3 Neural Networks

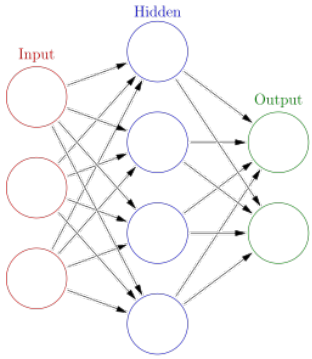

Figure 3.3: Architecture of a Neural Network

Figure 3.3 shows the architecture of a neural network. A neural network consists of layers of nodes that are connected to each other. It consists of a input layer of nodes through which input is fed to the network. The output or the predictions are obtained from the output layer which consists of multiple nodes based on the number of classes in the target. In between the input and the output layers, there can be multiple hidden layers. The most common type of Neural networks are the feed-forward neural networks in which the information moves only in the forward direction i.e. from the input layer to the output layer.

A node can receive input from more than one nodes in the previous layer. Each node in the hidden layer of a neural network is assigned weights. Any input received to these nodes are multiplied with the weights and summed up. If the value obtained is greater than a threshold value, then it is forwarded to the next layer. The weights assigned to the nodes are random at start. The network gradually learns the weights for each the nodes during the training process. Methods like gradient descent are used to optimize these weights.

Neural networks can learn linear and non-linear relationships between the data and also allows for parallel processing on the GPU. This allows for faster training on huge data sets.

The feed forward neural network is not capable of identifying temporal relationship between the data. If we pass a sequence of data to this network, for each value of the sequence, there will not be any memory of the previous values in the sequence. Since our goal is to identify the temporal patterns in these records, we need a model which identifies the relationship between the various values of a sequence and identify patterns within them to output the probability of whether the customer will default. A Recurrent neural network is a perfect example of such a model.

3.3.1 Recurrent Neural Networks

A recurrent neural network is generally used for time series analysis or for data that could be represented in the form of a sequence. Recurrent neural networks maintain memory which allows them to keep information of previous inputs in the sequence to predict the final output.

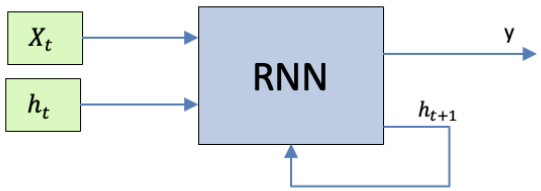

Figure 3.4: A Recurrent Neural network Node

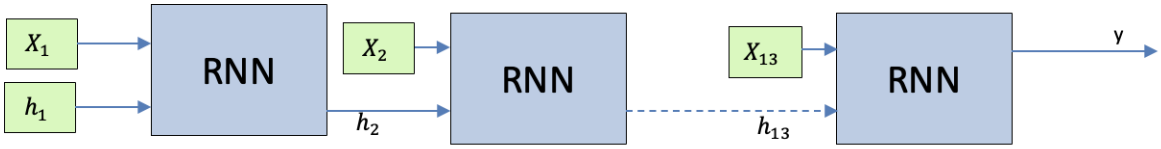

Figure 3.5: Expanded Recurrent Neural network Node

As can be seen in figure 3.5, a output state h_t from the first RNN blocks is used as feedback in the next RNN block. In this way, the RNN block memorizes the previous input in the sequence. Each RNN block has a weight assigned for the input data X_t and the output state h_t.

$$h_{t+1} = f(X_t, h_t, w_x, w_h, b)$$

Here w_x and w_h are the weights assigned for the input data and the feedback data respectively. b is the bias parameter.

Chapter 4

Methodology

4.1 Methodology Introduction

The research starts with analyzing the data by visualizing the distribution of data for each column for customers who have defaulted and customers who have not defaulted. Various hypotheses are made based on the visualizations and appropriate models are used to capture the relationship between the feature variables and the target. The initial approach is obtain a model which looks into the aggregated information of a customer for prediction. The next approach is developing a model to look into trends in the data. The final approach is to use the insights gained from both these models together for prediction.

4.2 Data

The data used for this analysis is provided by **?** which contains real customer data. To mask this sensitive information, the values are normalized, and the column names are given anonymous values. The number of customers in the data is 458913 and is recorded over a period of 13 months from March 2017 to March 2018.

File containing feature variables:

Every row in this file contains data of a customer for a particular month. Thus, one single customer can have a maximum of 13 records. There are certain cases in which the customer has stopped using the credit card service in between the window of observation. There are also some cases in which a customer starts a credit card service in between the 13 months' time frame. In both these cases, the number of records for a customer becomes less than 13.

This file contains 190 columns.

- **Customer_ID:** This is a unique key for each customer
- **S_2:** Date column which represents the date for which the corresponding column values are recorded.
- **Numerical Columns:** The data contains 177 numerical columns.
- **Categorical Columns:** The data contains 11 categorical columns.

File containing target variables:

Unlike the file containing feature variables, the target variable file contains consolidated information of whether the customer has defaulted or not in the given observation period of 13 months.

- **Customer_ID:** This is a unique key for each customer
- **Target:** This column represents if the customer has defaulted or not. A target value of 1 represents that the customer has defaulted and a target value of 0 represents that the customer has not defaulted.

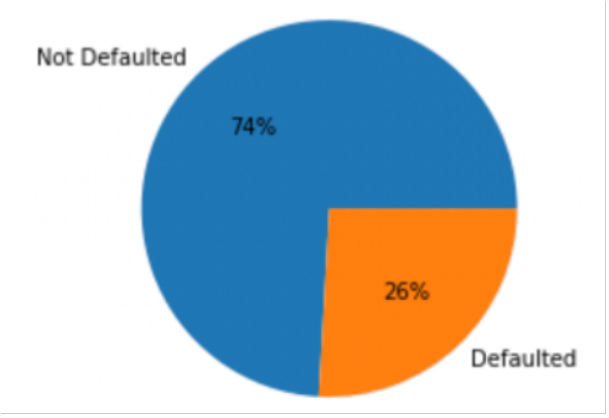

Figure 4.1: Distribution of the target column which shows if the customer has defaulted or not

Figure 4.1 shows the distribution of the target column. As can be seen, the distribution of the data is quite imbalanced. Only 26 % of the customers have defaulted

4.2.1 Data Preparation

The file available for this data is provided in the form of csv. The total file size of the csv is 16 GB. Reading the file for analysis takes a lot of time which would slow down the time taken for its analysis. Hence the csv files were converted into parquet files.

Parquet Files:

Parquet is a open source columnar file format which is used to speed up the retrieval of files. Traditional file formats store tabular data in the form of rows.

No.	**Animal**	**Type**
1	Dog	4
7	Cat	8
2	Mouse	6

Table 4.1: Sample Data

1	Dog	4	7	Cat	8	2	Mouse	6

Table 4.2: Row wise Data storage

1	7	2	Dog	Cat	Mouse	4	8	6

Table 4.3: Columnar Data storage

Table 4.2 shows the row wise storage of the data in table 4.1 and table 4.3 shows the storage of the data in a columnar format. In a row wise storage format, the encoding applied while storage is generic to all columns. However in a columnar storage, since all records belonging to a certain column are close to each other, it allows for appropriate encoding of each column which leads to lower file size and thus faster file retrieval.

4.2.2 Data Imputation

Figures 4.2 and 4.3 show the percentage null values in each column of the data. As can be seen a lot of columns have more than 60% null values. These columns were removed from the data. For columns with null values less than 60%, a simple imputer was used. For numerical values, the null values were replaced by the median of the values in each column. For categorical columns, the null values were replaced by the mode of the column values.

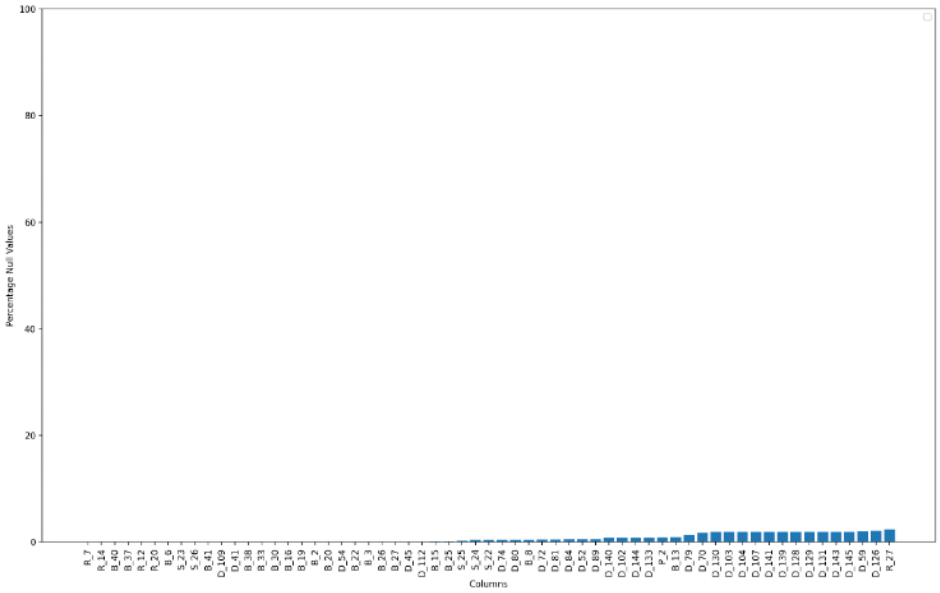

Figure 4.2: Percentage null values in each column: Part a

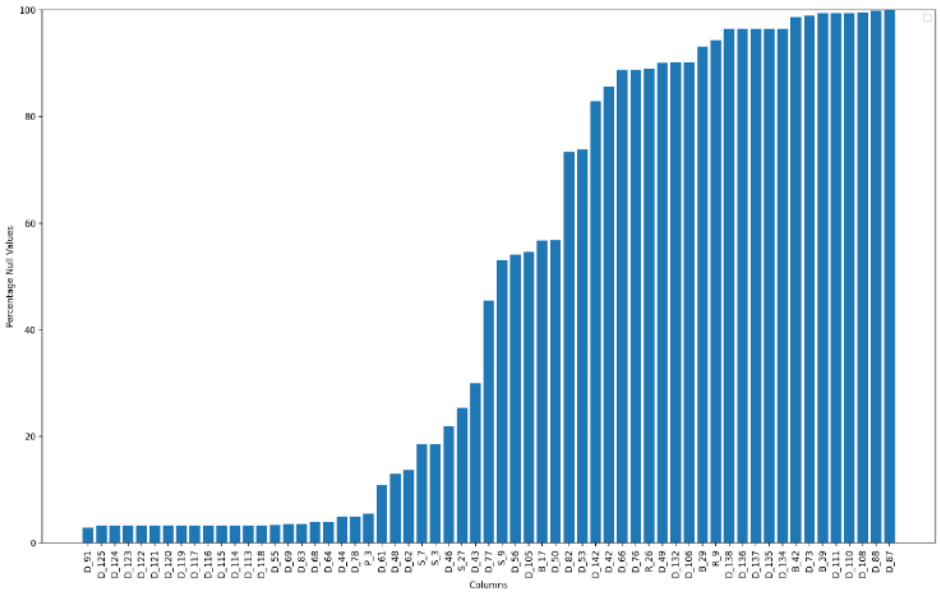

Figure 4.3: Percentage null values in each column: Part b

4.2.3 Splitting Data for training and testing

The entire data was split in such a manner that 80% of the customer data was used for training and the remaining 20% were used for testing. The test data was used for evaluation and comparison of the models.

4.3 Predictions using Aggregated Data

A mean of all the variables for each customer was taken. Before trying to fit a model, it is important to visually analyse if there is a relationship between the aggregated data and the target column. Since each customer has at most 13 records, the mean of the column values were taken per customer. It is very difficult to visualize the effect of these aggregated variables as the data contained more than 3 columns. Hence a dimensional reduction using Principal Component Analysis was performed to reduce the number of columns to 3.

4.3.1 Visualizing Principal Components of the research dataset:

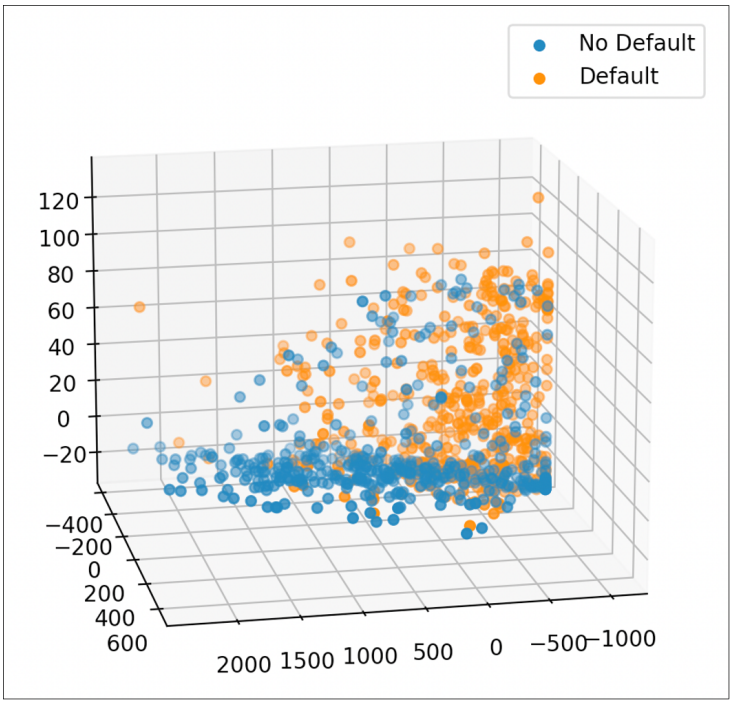

Figure 4.4: 3d Scatter plot of the data obtained by performing dimensional reduction on the research dataset

A small sample of the data was taken and plotted on a 3d graph to observe if different clusters are formed for customers who have defaulted and customers who have not

defaulted.

As can be seen in figure 4.4, there are clearly two separate clusters formed for customers who have defaulted and customers who have not. This means that the data obtained by aggregating the records for each customer can be used to distinguish good customers from bad customers.

4.3.2 Using Random Forest model for predictions

A random forest model was fit using the data that was obtained after aggregation. Gini Impurity was used as a criteria to measure the purity of individual splits.

As shown in 4.1, 74% of the data corresponds to customers who have not defaulted. Thus the data is imbalanced. This can lead to the output of the random forest model being biased. To mitigate this issue, various methods to balance the data were tried out

- Cost sensitive Classification/Balanced weights classification: A penalty is given for incorrectly classifying an record. The penalty assigned is in inverse proportion to the frequency of that class in the data. Thus the miss-classification of a record belonging to the minority class has a higher penalty. The weights associated with class j is given as:

  $$weight_j = \frac{total\ samples}{number\ of\ classes * number\ of\ samples\ of\ class\ j}$$

  In our data, class 1 constitutes of 26% of the data and class 0 constitutes 74% of the data.

  $$weight_0 = \frac{100}{2 * 26} = 1.923$$

  $$weight_1 = \frac{100}{2 * 74} = 0.675$$

- Near miss under sampling: A package provided by Imbalanced-learn is used to perform near miss under sampling.

- SMOTE over sampling: A package provided by Imbalanced-learn is used to perform SMOTE over sampling.

Another important factor while using a random forest model is hyper parameter tuning. In random forest, choice of the depth of individual trees is quite crucial. If the depth is too less, then it can lead to under fitting. If the depth is very high, then it can lead to

over fitting. Hence we need to find the optimum depth parameter. The optimum depth parameter is obtained by using random search algorithm.

Random Search algorithm

A range of depth values is provided to the random search algorithm. The algorithm then randomly selects a value from the range and evaluates the model. This process is repeated till a given number of iterations. At each iteration, if the model gives the best performance so far, then that value is stored.

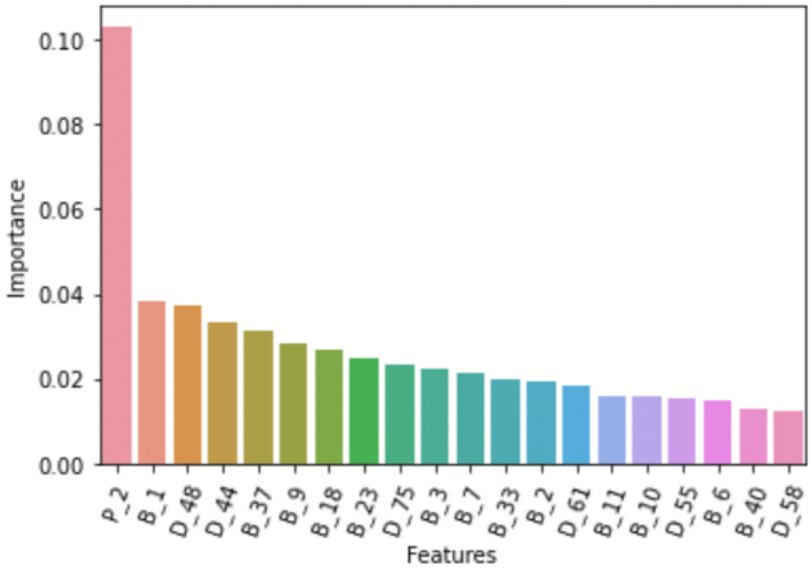

Figure 4.5: Feature importance of the key features

Once the imbalance in the data is taken care of, a random forest model is fit on the data. Importance of individual features in the model were obtained. Figure 4.5 shows that the feature P_2 has the most importance.

4.4 Prediction using trends present in the data

The defaulting behaviour of a customer can be said to be a function of his/her spending habits. Customers who default have a different spending pattern than customers who do not default. These patterns in the data can be used to predict if the customer will default or not.

4.4.1 Visualizing different patterns in data:

To visualize the above hypothesis, a new column called time_rank was added to the data. This column was populated by values ranging from 1 to 13 based on the data column S_2 in the data. Thus for a customer, the first month's record had the time_rank column populated as 1, the second month's record as 2 and so on. The entire data was then divided into two subsets. One containing the customers who defaulted and other containing customers who did not default. For each of these subsets, a linear regression model was fit with the time_rank column as the independent variable and each column in the data as a dependent variable. This was done for all the columns. Since we had 188 columns in the data and 2 different subsets, a total of 376 models were fit. The linear models obtained was then plotted on a graph for each column to visualize the different patterns for the defaulted customers and the non defaulted customers.

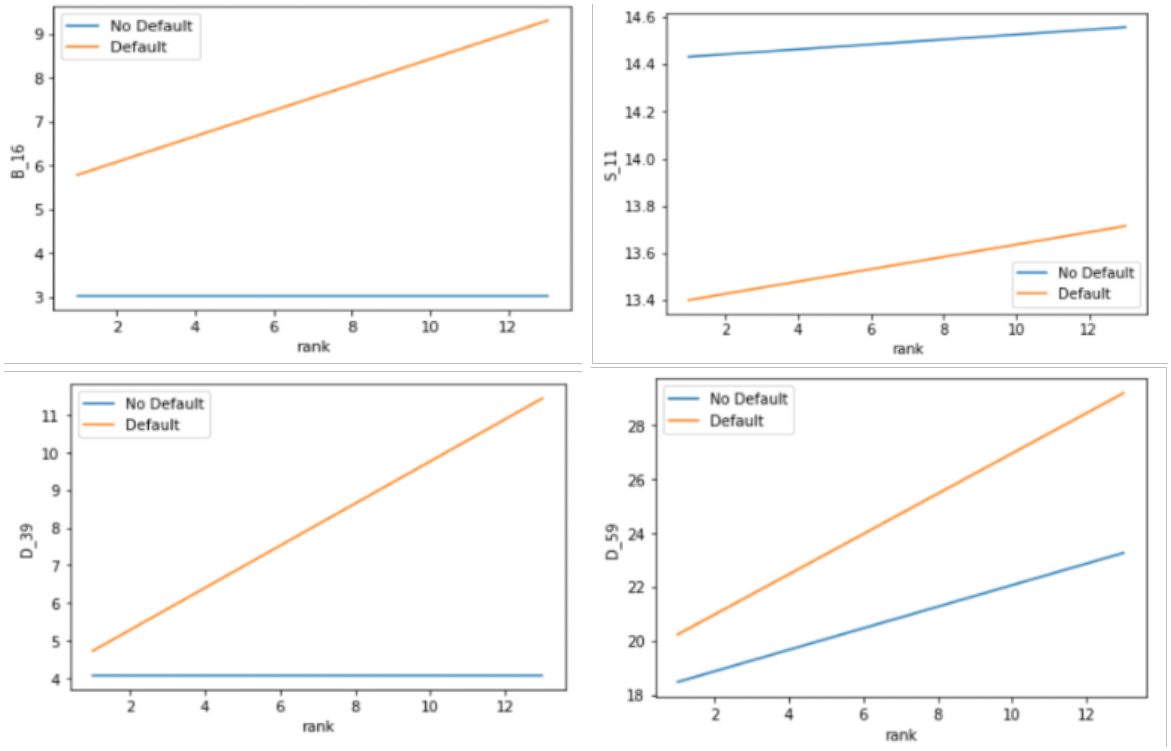

Figure 4.6: Liner Regression Models for 4 of the 188 columns

Figure 4.6 show some of the columns with the most prominent variation in trends for the different types of customers. The visualization thus shows that there are different patterns for customers who default and customers who don't. Thus we need a model which can identify these monthly trends for each customers and then output whether they have defaulted or not.

Using a RNN model for predictions:

Since we plan to use the temporal relationships in data to predict if the customer will default or not, we can use a RNN model. For each customer, the RNN model will take a sequence of the 13 records and then output a value of 0 or 1. However in the data, there are some customers with less than 13 records. In such cases, the data is padded with zeroes. Example: If a particular customer has only 10 records, then all columns belonging to that customer is padded with 3 zeroes to make the total record count as 13.

To address the issue of data imbalance, a cost sensitive classification or balanced weights classification is used.

4.4.2 Exponentially Weighted Average Ensemble Model

This is an attempt to combine the predictions obtained from training a model on the aggregated data and the predictions based on temporal trends in the data. This idea was first proposed by Tsai et al. (2018). The output obtained from each of the models are assigned weights. The predictions are multiplied by their weights and summed up. The weights are assigned in such a way that the model that gives the higher accuracy is given a higher weight.

Tsai et al. (2018) proposes that a exponentially decreasing weight works best when the accuracy of all models are quite close to each other. The weight assigned to model n is given as:

$$weight_{model(n)} = \frac{(1 - accuracy_{model(n)})^{-x}}{\sum_n (1 - accuracy_{model(n)})^{-x}}$$

4.5 Methodology Summary

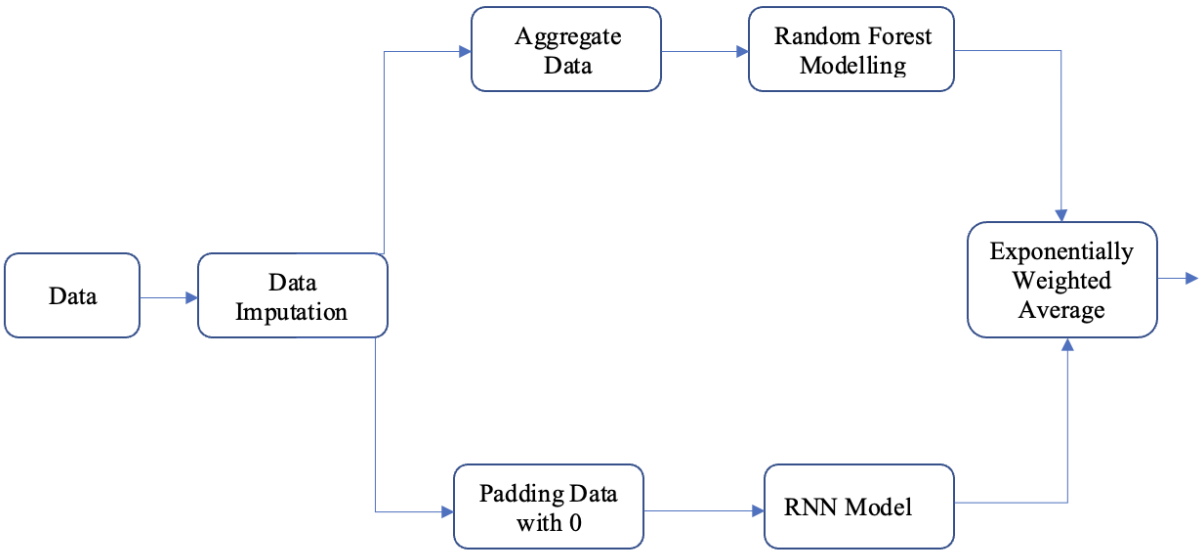

Figure 4.7: Summary of methodology

Figure 4.7 shows how the two models are individually modelled and then an exponential weighted average is obtained to get the final output. The Random forest model aims at getting predictions from aggregated information of a customer while the second model aims to get predictions from temporal trends present in the historical data of a customer.

Chapter 5

Evaluation

5.1 Metrics used for Evaluation

Both the models have been evaluated using a number of metrics. Before understanding the evaluation metrics used, it is important to understand the concept of confusion matrix.

Confusion Matrix

		Actual Values	
		1	0
Predicted Values	1	True positive	False Positive
	0	False Negative	True Negative

Figure 5.1: Confusion Matrix

Figure 5.1 shows the confusion matrix. The horizontal axis represents the actual values in the data and the vertical axis shows the predicted values by the model. Positive refers to the value 1 and negative refers to the value 0.

- **True Positive (TF)**: The top left quadrant shows the number of positive records which were predicted as positive.

- **False Positive (FP)**: The top right quadrant shows the number of negative records which were predicted as positive.
- **False Negative (FN)**: The bottom left quadrant shows the number of positive records which were predicted as negative.
- **True Negative (TN)**: The bottom right quadrant shows the number of negative records which were predicted as negative.

Using the above information about a confusion matrix, we can now define some evaluation metrics used for classification.

Accuracy

Accuracy is calculated as the ratio of the correctly predicted records to the total records in the data.

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

Although this is the mostly widely used metric, these can be misleading when the data is not properly balanced. Example: If we consider a data considering 99 positive records and 1 negative record, then even if the model classifies everything as positive, we will get an accuracy of 0.99. Hence accuracy is only used when the data is properly balanced.

Precision

Precision is generally used when we need to measure the models efficiency in predicting positive values. In cases when evaluating false positives is more important than false negatives, this could be a good metric.

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

Recall

Recall is used when we need to measure the models efficiency in predicting negative values. In cases when evaluating false negatives is more important than false positives, this could be a good metric.

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

For our case, this is an important evaluation factor. If a customer is predicted to default and if he does not, there is not much loss to the banks. However, if a customer is predicted to not default, but defaults in real, then it can lead to a loss for the bank. Hence we need to minimize the number of wrong predictions for the negative values.

F1 score

When evaluating both false positives and false negatives are important, we use F1 score. F1 score is calculated as the harmonic mean of precision and recall.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

5.2 Evaluation of Random Forest

Model	Accuracy	Precision	Recall	F1 score
Basic	0.8909	0.8114	0.7546	0.7819
Balanced Weights	0.8952	0.805	0.786	0.795
Near Miss under-sampling	0.7598	0.5186	0.9559	0.672
SMOTE over-sampling	0.759	0.517	0.955	0.671

Table 5.1: Evaluation results of random forest

Table 5.1 shows the evaluation result of various techniques used in random forest. The basic random forest model has a high accuracy of 0.8909. The balanced weight model performs slightly better than the basic random forest model in terms of accuracy. The sampling methods like Near miss and SMOTE have show reduced accuracy than the other methods. However it is worthwhile to note that the sampling methods have a higher recall value. This shows that the sampling methods predicted a higher percentage of the defaulters correctly than the other methods. However the precision of the sampling models is poor comparatively. This means that among the customers who were predicted as defaulters, only a small percentage of them actually defaulted.

Although the sampling methods have a higher recall, the drop in precision is too much and hence these models cannot be said to be the most optimum methods to balance data. Considering all the criteria, random forest with balanced weights seems to the best model. It not only has a high accuracy, but also has a better recall value compared to the basic random forest model. The same conclusion can be reached by observing the F1 scores of the different models. Random forest with Balanced weights has the highest F1 score among all the models

5.3 Evaluation of RNN models

Model	Accuracy	Precision	Recall	F1 score
Basic RNN	0.915	0.785	0.7782	0.7816
Balanced Weights	0.8931	0.662	0.925	0.772

Table 5.2: Evaluating results of Recurrent Neural Network

Table 5.2 shows the evaluation metrics of the different RNN models. Right away, we can see that this model performs much better than the random forest model. The accuracy of the basic RNN model is better than the RNN model using balanced weighting. However the RNN model with balanced weighting has a better recall value as compared to the basic RNN model. Just like the random forest model, this gain in recall is obtained at the expense of precision.

When we compare the metrics of the RNN models and the random forest models, we can see that the RNN model gives a good value of recall without much drop in the final accuracy. Thus the RNN model which looks into trends in the historical data for its predictions performs much better than the random forest model which looks into the aggregated information for its predictions.

5.4 Evaluation of ensemble model

Model	Accuracy	Precision	Recall	F1 score
Exponentially Weighted Ensemble Model	0.923	0.911	0.686	0.782

Table 5.3: Evaluating results of Exponentially Weighted Ensemble Model

As can be seen from table 5.3, by combining the predictions of the random forest model and the RNN model, we get a higher accuracy in predictions. This model also has the highest precision among all the given models.

As can be seen in figure 5.2, the exponentially weighted ensemble model and the RNN model have the highest accuracy. The lowest accuracy is provided by the random forests using sampling methods.

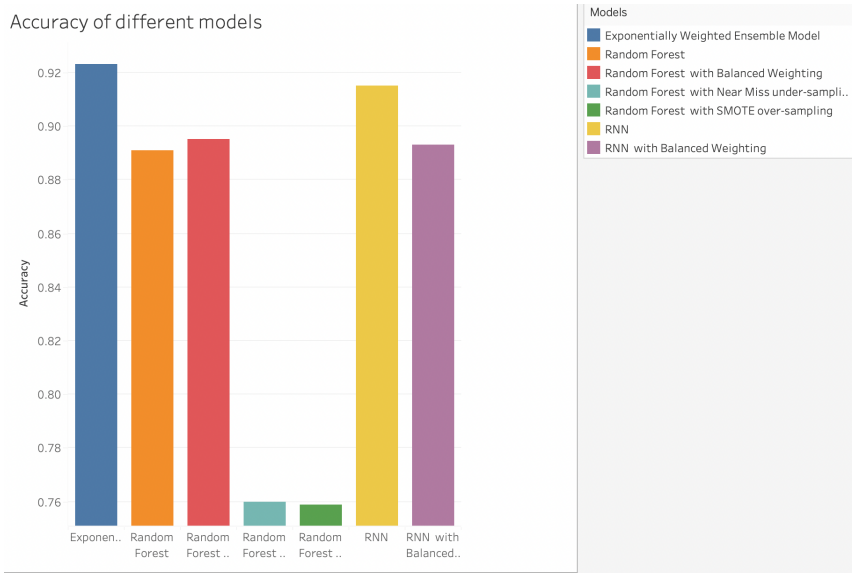

Figure 5.2: Bar plot of accuracy for the different models

Figure 5.3 shows that the exponentially weighted ensemble model and random forest have the highest precision, while the random forest models with sampling have poor precision.

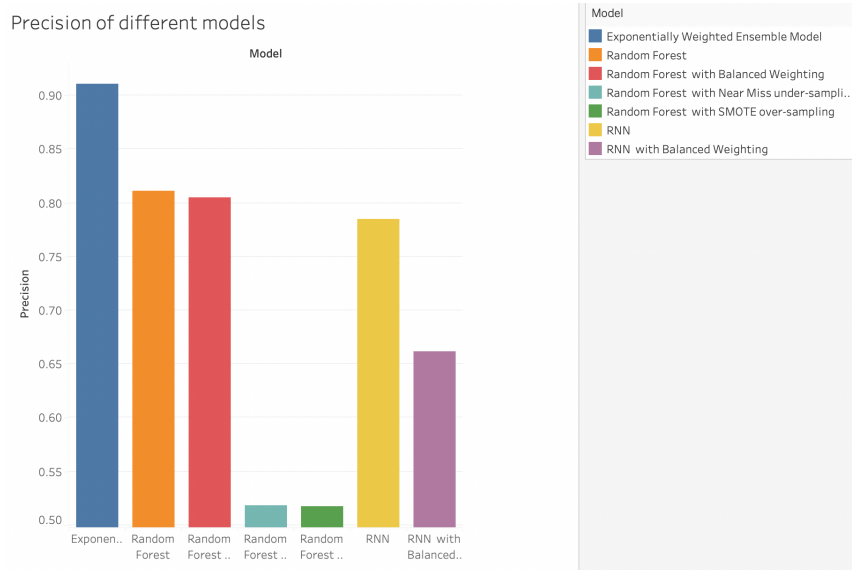

Figure 5.3: Bar plot of precision for the different models

Although the random forest models using sampling have a lower accuracy and precision among all models, they have the highest recall values as shown in 5.4. This shows that the sampling methods indeed achieved its target of balancing data appropriately.

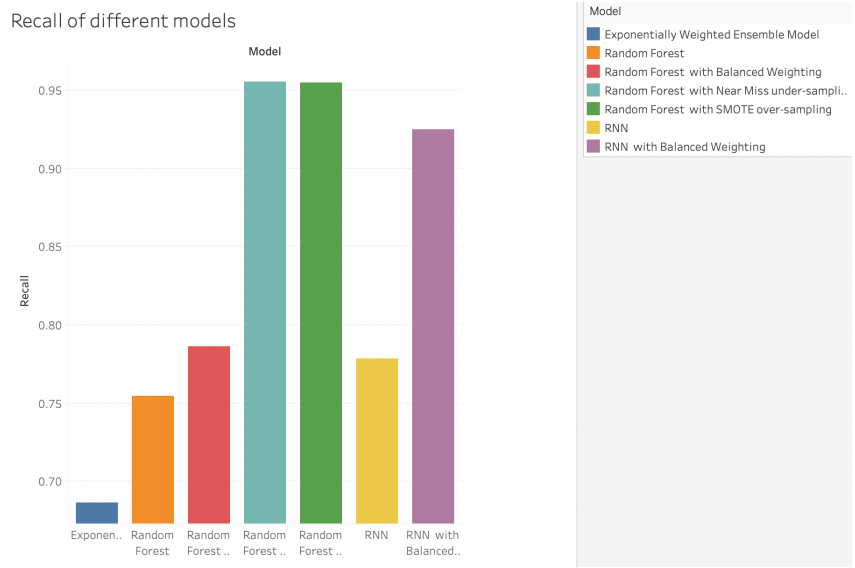

Figure 5.4: Bar plot of recall for the different models

Going by the F1 scores, the random forest model with balanced weighting applied to the classes works the best.

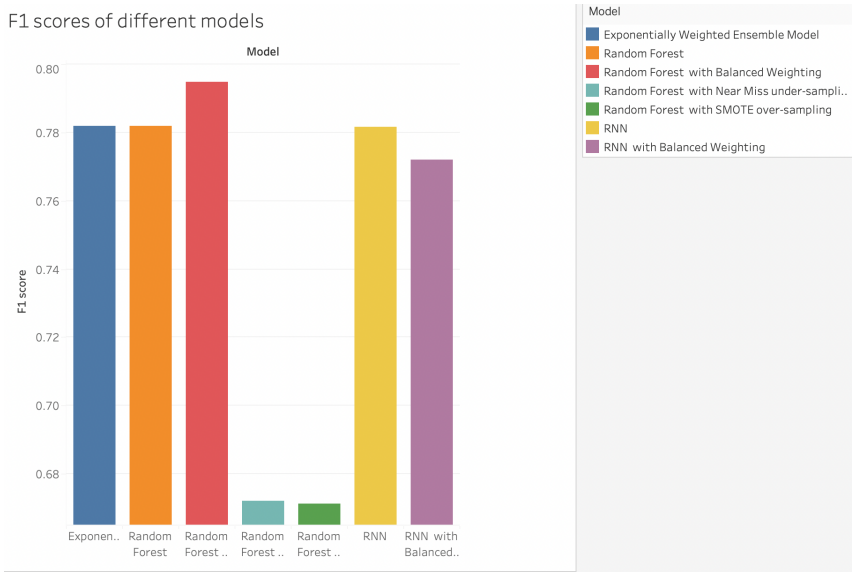

Figure 5.5: Bar plot of f1 score for the different models

Chapter 6

Conclusion

The motivation of this research was to find ways to model customer data to predict efficiently if a customer will default or not. The research provided two different ways of extracting insights from a high dimensional customer data. The first approach used aggregated customer data to fit a random forest model. The second approach treated the historical data of a customer as a sequence of data and found different trends in the data for a customer who defaults and a customer who does not. Both of these approaches gave very promising results. Later the insights obtained from both these approaches were combined by using an exponentially weighted average approach. The research showed that combining the insights of both these models gave the highest accuracy.

The research also looked into addressing the data imbalance issue which is commonly present in data dealing with defaults. Different approaches like under-sampling, over-sampling and cost based classifiers were used to fix the problem. It was observed that sampling methods provided the highest recall values, but at the expense of the models accuracy. The cost based classifiers seemed to be much robust than the sampling methods. It improved the recall of the model with very little effect on accuracy.

The research also showcased using analytics, machine learning models and visualization on huge datasets with large number of dimensions. For visualization, the results were brocken down to its principal components and the 3 most relevant principal components were used to analyse the effects of the various feature variables on the target column.

6.1 Limitations

Some of the limitations of this research originated because of the fact that the data size was huge. The following were the limitations:

- The two approaches followed for data imputation in this research were removing

feature with higher percentage of null values and imputing the records with the median/mode of the column. However, each individual columns could have been analyzed better and specific imputation techniques could have been used based on the reason for the column values being null. This was not done because the data size was huge and analyzing each and every column would have need a higher computational power.

- The hyper-parameters of the recurrent neural network were not tuned. Each run for the recurrent neural network model required around 20 minutes. Hyperparameter tuning would have required the recurrent neural network to run almost 40 to 50 times. This process would have taken a really long time.

6.2 Future scope

As already discussed in the limitations section above with the use of a system with higher computational power, better data imputation strategies like using nearest neighbours or multi-variate regression models can be implemented. This could increase the accuracy of the models even more.

The sampling methods used to address the data imbalance issue gave high recall values for the random forest model. But the accuracy of the model dropped drastically. More research could be done on developing sampling methods which do not affect the accuracy of the models much.

Bibliography

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural computation*, 9(7):1545–1588.

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

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

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Do, H. W. and Jeong, Y.-S. (2016). Temporal relation classification with deep neural network. In *2016 International Conference on Big Data and Smart Computing (BigComp)*, pages 454–457. IEEE.

Gerardi, K., Herkenhoff, K. F., Ohanian, L. E., and Willen, P. S. (2018). Can't pay or won't pay? unemployment, negative equity, and strategic default. *The Review of Financial Studies*, 31(3):1098–1131.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417.

Lawrence, S., Giles, C. L., and Fong, S. (2000). Natural language grammatical inference with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 12(1):126–140.

Lawton, W. H. and Sylvestre, E. A. (1971). Self modeling curve resolution. *Technometrics*, 13(3):617–633.

Mani, I. and Zhang, I. (2003). knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126, pages 1–7. ICML.

Nik (2022). Decision tree classifier with sklearn in python • datagy.

Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., and Brunk, C. (1994). Reducing misclassification costs. In *Machine Learning Proceedings 1994*, pages 217–225. Elsevier.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

Perera, P. (2019). Decision tree approach for predicting the credit risk of leasing customers in sri lanka. In *Proceedings of the 3rd International Conference on Business and Information Management*, pages 65–68.

Seb, P. b., Author Seb, A., and Posts, R. (2022). Principal components analysis explained for dummies.

Tsai, K.-Y., Ding, J.-J., and Lee, Y.-C. (2018). Frontalization with adaptive exponentially-weighted average ensemble rule for deep learning based facial expression recognition. In *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pages 447–450. IEEE.

Venkatesh, A. and Jacob, S. G. (2016). Prediction of credit-card defaulters: a comparative study on performance of classifiers. *International Journal of Computer Applications*, 145(7).

Vikram, M., Pavan, R., Dineshbhai, N. D., and Mohan, B. (2019). Performance evaluation of dimensionality reduction techniques on high dimensional data. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1169–1174. IEEE.

White, B. T. (2010). Take this house and shove it: the emotional drivers of strategic default. *SMUL Rev.*, 63:1279.

Zhou, J., Li, W., Wang, J., Ding, S., and Xia, C. (2019). Default prediction in p2p lending from high-dimensional data based on machine learning. *Physica A: Statistical Mechanics and its Applications*, 534:122370.