Prediction of Customer Churn in Telecommunication using Machine Learning Algorithms

---

Perlate Diala

Supervisor:
Dr. Hairong Wang

A research report submitted in partial fulfillment of the requirements for the degree of Master of Science in the field of e-Science

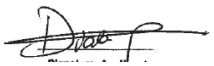
in the

School of Computer Science and Applied Mathematics
University of the Witwatersrand, Johannesburg

01 June 2020

Declaration

I, Perlate Diala, declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in the field of e-Science at the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination at any other university.



Perlate Diala

01 June 2020

Abstract

Loss of customers in telecommunication industries has become one of the major concerns in recent years. This is due to a very high of competition among industries and the customer acquisition costs, so it is of great value to keep existing customers. For that purpose, it is of great significant to prevent churn by implementing prediction models that are effective and accurate. However, the major problems with building models for telecommunication are large volumes of data, enormous feature space and Class Imbalance Problem (CIP). This study aims to compare the performance of various machine learning classifiers for the prediction of customer churn in telecommunication. In particular, we explore some pre-processing of the dataset such as dimensionality reduction and seven oversampling techniques to reduce CIP, and hence to improve the performance of the concerned machine learning models. To evaluate the performance of selected machine learning models, the Receiver Operating Characteristic and Area Under the Curve (ROC-AUC curve) was adopted. The experimental results showed that the Logistic Regression classifier coupled with Random Oversampling (ROS) and dimensionality reduction based on linear autoencoder performs better than all other classifiers.

Acknowledgements

I acknowledge with sincere appreciation and deep gratitude to my research supervisor, Dr. Hairong Wang, for her guidance, advice, support and and critical comments during the time of this study. The support of the DST-CSIR National e-Science Postgraduate Teaching and Training Platform (NEPTTP) with the funding towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NEPTTP.

Many thanks to Prof. Celik, Dr. Helen Robertson, and Ms. Casey Sparkes for their continuous and arranging monthly meetings to discuss research progress.

I would moreover like to expand my gratitude to all my friends for proofreading my work. Lastly, I wish to thank my parents for their support and encouragement throughout my studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Customer churn, simply meaning loss of customers, can be defined according to the type of business. For instance, in [50, 53] customer churn in telecommunication was defined as a condition in which existing customers or clients migrate from a company or cancel their service to move to its competitors [40], or in a case where a clients end their relationship with the organisation or close their accounts [49]. In the telecommunication businesses, customer churn or customer defection problem is one of the fundamental concentration in the Customer Relationship Management (CRM) division [50]. The main purpose of CRM in customer churn is to build strength and foster a good relationship with existing customers. According to [4] many telecommunication companies are suffering from a higher churn rate due to competition from their competitors. Moreover, [61] showed that customer turnover plays a very important role in a company as it negatively affects the business's revenues and growth. Due to a very high cost of acquiring new customers and competition among industries, numerous telecommunication companies are presently moving their concentration from customer procurement to customer maintenance [51, 66].

Given these potential negative effects, [22] argued that companies especially those working with big data such as telecommunication, marketing, and insurance must focus on developing effective and highly accurate predictive models that will be able to flag/identify customers that are at risk of defecting or leaving. Over time, different statistical techniques and machine learning models have been implemented and tested to predict customer turnover. Machine learning techniques such as ensemble learning-Random Forest (RF), Artificial Neural Network (ANN) [4, 60] yield

a higher accuracy as compared to statistical models such as Logistic Regression (LR) and Naive Bayes (NB) [60]. Moreover, the Support Vector Machines (SVM) with different optimal penalty parameters and kernel parameters on grid search and cross-validation can provide high accuracy [52].

When building churn prediction models for this specific industry, the problem that emerges is the issue of CIP [25, 28]. The CIP is widely known issue in the context of classification tasks. This problem can be defined as a situation whereby one of the classes we are trying to predict is rarely or highly represented than the other classes [11, 62]. For example, some of the applications known to have class imbalance issues include, but not limited to are medical diagnosis [67], customer attrition [68], and credit risk [19]. The main setback with learning from imbalance class is that machine learning classifiers turn to ignore class distribution by just concentrating only in the majority class and overlook the minority class which is our interest. Therefore, when building machine learning classifiers on imbalance classes, there is a great chance of getting misleading results. For example, consider a rare case of two classes with 98% majority and 2% minority class respectively. A classifier would give the accuracy of 98% and ignore the minority class which we are interested in. In order to overcome this problem, different techniques, i.e. data-level, algorithms, and ensemble approaches have been proposed. Data-level solutions attempts to balance the data by up-sampling the minority class or down-sampling the majority class. Some of the data-level solutions methods are; Random Over-sampling (ROS), Random Under-sampling (RUS) [25], and synthetic methods such as Synthetic Minority Over-sampling Technique (SMOTE) [11] and Adaptive Synthetic (ADASYN) sampling approach [27]. Algorithmic approaches try to improve the classification performance by focus lower represented class. This approach is divided in cost-sensitive and one-class learning. Cost-sensitive assign high and low costs to minority and majority classes respectively [58]. On the other hand, one-class learning approach learn the data of one class and ignore the rest [56]. Lastly, ensemble solutions uses the data-level solutions combined with ensemble learning such as RF and boosting to overcome CIP [18]. These techniques have the potential to improve customer churn prediction performance [25, 68].

Another essential step in building machine learning models is feature engineering. Feature engineering is defined as a task of building or generating new input variables using the existing variables from the raw dataset [21]. The purpose of applying feature engineering step is to obtain new features that best represent the underlying problem and in-turn improve the prediction performance. Hence, feature engineering play a very important part for any prediction models. One of the main issues that are essentially encountered in a telecommunication is the large amount of features space. These problem turns to affect or reduce the performance of the models as well as long-time to train. However, different feature selection and feature engineering techniques have been proposed [43, 63]. For the purpose of this study, feature engineering based on dimensionality reduction using autoencoders with various activation functions, i.e., under-complete autoencoder (based on linear, and non-linear sigmoid activation functions), sparse autoencoder (based on non-linear relu activation function), and deep autoencoder was used to address this issue.

1.1 Problem Statement

Most service-based businesses are now facing a high customer churn rate. Statistical techniques such as LR and NB have been widely used to solve this problem. Recently, different machine learning approaches have been implemented for customer defection prediction. These approaches have the potential to improve the prediction performance of statistical approaches. However, there is a problem with class distribution and high dimensional datasets in telecommunication which affect machine learning-based model predictions. Thus, the problem we address in this research is to investigate whether machine learning algorithms with data re-sampling (ROS, SMOTE, ADASYN, BorderlineSMOTE, SMOTE+Tomek, and SMOTE+ENN) techniques provide satisfactory performance when evaluated on the selected dimensionality reduction techniques (under-complete and sparse autoencoders) based on various activation functions, i.e., linear autoencoder, non-linear sigmoid-based autoencoder, non-linear relu autoencoder, and deep autoencoder.

1.2 Significance of the Study

Previous researches have shown that customer churn is the major priority concern for telecommunication industries. The current study intends to make the following contribution: the evaluation of different supervised machine learning classifiers when combined with some selected pre-processing methods and different over-sampling techniques for churn prediction using telecommunication data.

1.3 Research Aim and Objectives

1.3.1 Aim

The aim of this study is to develop a predictive model that can identify customers that are likely to churn using telecommunication dataset.

1.3.2 Objectives

The objectives of this study are as follows:

- To apply dimensionality reduction techniques to address the issue of high-dimensional space.
- To apply over-sampling techniques to address the CIP, and hence improve the prediction accuracy.
- To apply LR, SVM, and RF as the predictive models in our study to identify the customers that are likely to churn using publicly available telecommunication datasets.
- To evaluate and compare the performance of all the above mentioned predicting models using ROC-AUC measure.

1.4 Research Question(s)

- Which of the selected autoencoders methods, i.e., under-complete (based on linear and non-linear sigmoid-based activation functions), sparse autoencoder

(with non-linear relu), and deep autoencoder performs best in reducing feature dimensionality?

- Which of the selected over-sampling techniques, i.e., ROS, SMOTE, ADASYN, Borderline-SMOTE, SMOTE+Tomek, and SMOTE+ENN would be more suitable in dealing with CIP for telecommunication data and in improving the overall prediction accuracy?
- Which of the selected supervised machine learning algorithm (LR, SVM, and RF) perform best in customer churn prediction using telecommunication data?

1.5 Limitations

This section addresses the scope of the research by identifying and discussing its limitations. The empirical results reported herein should be considered in light with some limitations. Firstly, we have performed missing values imputation using median and mode for continuous and categorical features respectively. However, given the nature of the high number of missing values in the larger dataset, it could have been interesting to investigate the performance of other missing values mechanisms. In addition, the class imbalance problem was only limited to oversampling techniques study. Although these techniques show potential to improve prediction performance, it would be important to investigate the significance of some undersampling methods. Lastly, given the constraint on time and computational power for techniques such as SVM, this study did not include parameter tuning for all models, which might somehow limit the generalisation performance of the models.

1.6 Overview

The remainder of this research report is divided into 4 chapters. Chapter 2 focuses on background knowledge about dimensionality reduction using autoencoders with various activation functions and supervised machine learning techniques, data-level solutions, models performance measure, and lastly, provides a review of related work on customer churn prediction. Chapter 3 presents a detailed research methodology. Section 3.2 breaks down the experiment setup into six

phases and presents the datasets and algorithms used. Chapter 4 analyses the experimental results and discussion. Chapter 5 provides a conclusion and highlights the direction for future work.

Chapter 2

Background and Related Work

2.1 Introduction

In this chapter, we provide a review dimensionality reduction using autoencoders with various activation functions as well as a review of supervised machine learning techniques (LR, SVM, and RF). In Section 2.4, we discuss the oversampling techniques to deal with the CIP. The evaluation metric for supervised classification is given in section 2.6. A review of related work on customer churn prediction is provided in section 2.7.

2.2 Dimensionality Reduction

Dimensionality reduction is a process of transforming data from high-dimensional feature space $\boldsymbol{X} \in \mathbb{R}^n$, to a low-dimensional feature space $\boldsymbol{X} \in \mathbb{R}^p$, such that $p < n$, without losing much information. One commonly used reduction method is Principal Component Analysis (PCA) which takes the original features $X \in \mathbb{R}^n$ and extracts new features that are linearly uncorrelated called principal components. To overcome the limitation of linear transformation used by PCA, in this study we will apply the Autoencoders which is able to incorporate the non-linearity between features [16].

2.2.1 Autoencoders

An autoencoder is a special type of neural network that is trained to generate output values that are similar to the input values. It is often used for dimensionality

reduction or unsupervised feature learning. For an autoencoder network, the number of the nodes in the input layer must be equal to the number of nodes in the output layer. There are other various types of autoencoders such as sparse, denoising, and variational autoencoders. Some of the interesting use of autoencoders are data denoising and dimensionality reduction. Typically, Autoencoder is a feed-forward NN that connects many neurons as shown in Figure 2.1. Autoencoders usually consists of 3 phases; encoder layer, code or hidden layer, and decoder layer [16].

1. **Encoder:** The encoder maps the original input data from the input layer to the hidden layers (lower dimensional space for dimensionality reduction) using some functions.

2. **Code** or **Hidden layer:** This part of the network is the desired lower representation features space that describe and represent the original inputs.

3. **Decoder:** The decoder takes the features from the hidden layer and attempts to recreate the original inputs data using some functions. In other words, the decoder function attempts to approximate the input data.

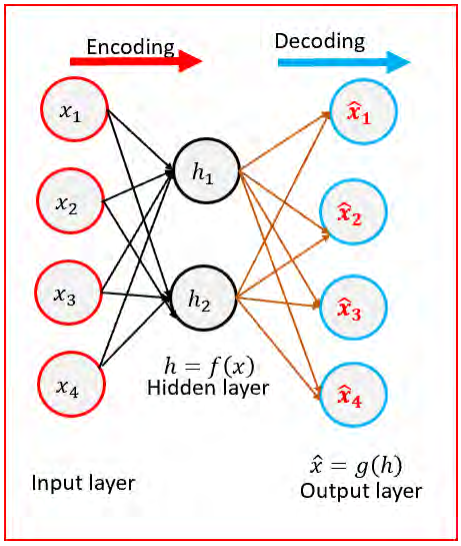


FIGURE 2.1: A simple representation of an autoencoder with one hidden layer [44].

From Figure 2.1, the red neurons with $\{x_1, \cdots, x_4\}$ represent the original features and black neurons $\{h_1, h_2\}$ are the compressed features (reduced feature space). The reconstructed features $\{\hat{x}_1, \cdots, \hat{x}_4\}$ of the original inputs features are shown in blue neurons.

To transform the weighted sum of inputs that goes into the neurons we apply the activation functions. These activation functions are used to compute the output of an NN and it can be a linear or non-linear function. However, the non-linear functions are mostly used in complex situations since they are confined to a restricted range. They are various popular activation functions that are adopted including ReLu, sigmoid, and hyperbolic tangent (tanh) functions. These activation functions are depicted in Figure 2.2.

- Sigmoid function: converts the encoding or decoding outputs into a range of $[0, 1]$ using the following bounding function

$$\tau(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

  where $\tau(x) \in [0, 1]$.

- Hyperbolic tangent (tanh) is a version of sigmoid which is bounded between -1 and 1,

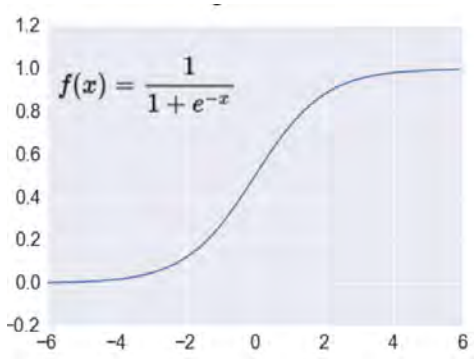
$$\tau(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

  for $\tau(x) \in [-1, 1]$.

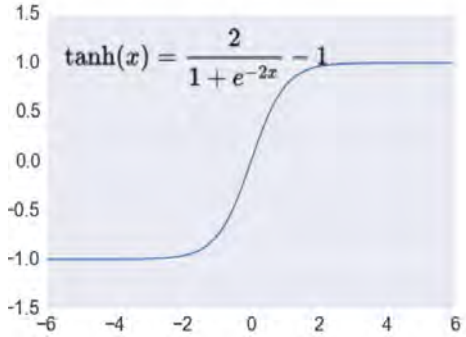
- Another activation function that is typically used is Rectified Linear Unit (ReLU) which is bounded between 0 and positive infinity given by

$$\tau(x) = max(0, x) \qquad x \geq 0 \tag{2.3}$$

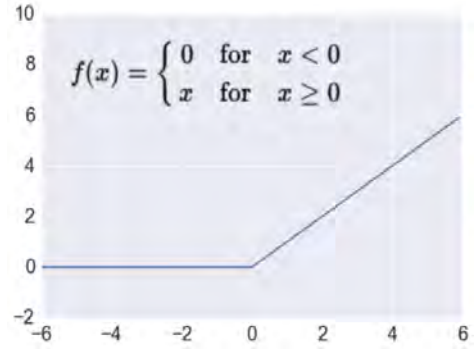
  where $\tau(x) \in [0, \infty]$.



(A) Sigmoid



(B) Tanh



(C) ReLU

FIGURE 2.2: Activation Functions [59]

2.2.1.1 Undercomplete Autoencoder

Suppose an input layer $\boldsymbol{x}$ of an m-dimensional vector is given, then the encoder part starts by transforming the input $\boldsymbol{x}$ to a hidden layer $\boldsymbol{h}$ of d-dimensional vector where $d < m$, with weights matrix $\boldsymbol{\alpha}_1$ of a $d \times m$ and a bias vector term $\boldsymbol{b}_1$ of a d-dimensional vector. Then an encoder transform the inputs $\boldsymbol{x}$ to a hidden layer using an activation function $\tau(\cdot) : \mathbb{R} \to \mathbb{R}$ for linear or $\tau(\cdot) : \mathbb{R} \to [0, 1]$ for non-linear sigmoid function.

$$\boldsymbol{h} = \tau_1(\boldsymbol{\alpha}_1 \boldsymbol{x} + \boldsymbol{b}_1) \tag{2.4}$$

Then the decoder attempts to compute the outputs from the hidden layer that approximates the given inputs. Now we are given that the hidden layer ϕ is a d-dimensional vector then the output layer $\boldsymbol{\phi}$ will be comprised of m-dimensional vector and a weight matrix of a $m \times d$ matrix and a bias vector term $\boldsymbol{b}_2$ of an m-dimensional vector. Then the dencoder transform the inputs $\boldsymbol{x}$ from a hidden

layer to the output layer using an activation function $\tau(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$ for linear or $\tau(\cdot) : \mathbb{R} \rightarrow [0, 1]$ for non-linear sigmoid function.

$$\boldsymbol{\phi} = \tau_2(\boldsymbol{\alpha}_2 \boldsymbol{h} + \boldsymbol{b}_2) \tag{2.5}$$

Then to train the autoencoder, we have to compute the parameters $\boldsymbol{W} = \{\boldsymbol{\alpha_1}, \boldsymbol{b}_1, \boldsymbol{\alpha_2}, \boldsymbol{b}_2\}$ using the gradient descent algorithm and the main objective is to minimise the reconstruction error ℓ between the original and recreated features [35, 45]. For linear reconstruction, that is τ_1 and τ_2 are linear, then the optimal solution will be the PCA and can be computed by minimising the following reconstructive error:

$$\ell(\boldsymbol{W}) = \min_{\boldsymbol{W}} \frac{1}{2N} \sum_{m=1}^{N} \|\boldsymbol{x}^{(m)} - \boldsymbol{\phi}^{(m)}\|^2 \tag{2.6}$$

For non-linear reconstruction, that is if τ_1 and τ_2 are non-linear, the reconstruction error can be given by:

$$\begin{aligned} \ell(\boldsymbol{W}) &= \|\boldsymbol{x} - \boldsymbol{\phi}\|^2 \\ &= \|\boldsymbol{x} - \tau_2(\alpha_2(\tau_1(\alpha_1 \boldsymbol{x} + \boldsymbol{b}_1)) + \boldsymbol{b}_2\|^2 \end{aligned} \tag{2.7}$$

2.2.1.2 Sparse Autoencoder

For sparse autoencoders, a regularisation penalty term, is included in the hidden layer $\boldsymbol{h}$. The main objective of sparse autoencoders is to penalise the neurons whose values are close to zero and retains the ones with output close to one. The penalty term in sparse autoencoder is based on the idea of using the Kullback-Leibler (KL) divergence which essentially explains how the two distributions are different [31]. Then the loss function for sparse autoencoder is given by:

$$\ell_{sparse}(\boldsymbol{W}) = \ell(W) + \beta \sum_{i=1}^{h_2} KL(\theta||\hat{\theta}) \tag{2.8}$$

Where, h_2 is the number of units in the hidden layer, KL divergence is given by

$$KL(\theta||\hat{\theta}) = \theta \log(\frac{\theta}{\hat{\theta}_i}) + (1 - \theta) \log(\frac{1 - \theta}{1 - \hat{\theta}_i})$$

The penalty term is given by

$$\beta \sum_{i=1}^{h_2} KL(\theta||\hat{\theta})$$

Where the weight parameter β is the adjusted parameter. The sparse autoencoder pushes $\hat{\theta}$ to be equal to the sparsity parameter θ so that some activation can close to zero and $\hat{\theta}$ is the average activation of the hidden layer [16].

2.2.1.3 Deep Autoencoder

An autoencoder network with more than one hidden layer is referred to deep autoencoder [5]. Since the deep autoeconder is more like a multi-layer neural network, so it has the same procedure for training a network. In deep autoencoder, we simply compute the composition of functions for the encoder and the composite functions for the decoder.

2.3 Supervised Machine Learning

[1] This section provides details of each supervised machine learning technique.

2.3.1 Logistic Regression

Logistic Regression is a supervised machine learning classification technique that is used to model the binary response. It can take an input variables that are either categorical, numeric or heterogeneous data types. For our case the target variable is given by

$$y_i = \begin{cases} 1 & \text{if } customer \quad churned \\ 0 & \text{if } customer \quad retained \end{cases} \tag{2.9}$$

[1] The discussion in this section is informed by a discussion that appears in (Perlate Diala), (Prediction of depression among university students using machine learning algorithms), University of the Witwatersrand, 2018. All material from external sources has been referenced.

When modelling the output variable, we use the linear combination of variables (features) and some weights $\boldsymbol{\eta}$:

$$g_\eta(\boldsymbol{x}) = \eta_0 + \sum_{k=1}^{m} x_k \eta_k \tag{2.10}$$

Where x_k represents the given variables and m is the total number of variables. LR is a linear method, however, in order to make the probability predictions, we need to transform predictions of $g_\eta(\boldsymbol{x})$ into the binary using the logistic function given in Eq.2.1. In addition, we need to build a classifier $\rho_{\boldsymbol{\eta}}(\boldsymbol{x})$ which may be used to assign new observations to their respective classes [14, 55].

Now substituting Eq.2.10 in Eq.2.1 we get the following LR function.

$$\begin{aligned} \rho_\eta(\boldsymbol{x}) &= \frac{1}{1 + e^{-(\eta_0 + \sum_{k=1}^{m} x_k \eta_k)}} \\ &= \frac{1}{1 + e^{-\boldsymbol{\eta}^T \boldsymbol{x}}} \end{aligned} \tag{2.11}$$

Where $\boldsymbol{\eta}^T$ represents the matrix transpose of $\boldsymbol{\eta}$ and

$\boldsymbol{x} = \left[x_0, x_1, \cdots, x_m\right]^T$
$\boldsymbol{\eta}^T = \left[\eta_0, \eta_1, \cdots, \eta_m\right]$
$\boldsymbol{\eta}^T \boldsymbol{x} = \sum_{k=0}^{m} \eta_k x_k = \eta_0 x_0 + \eta_1 x_1 + \cdots + \eta_m x_m$

The parameters $\boldsymbol{\eta}$ are estimated using the gradient descent algorithm [55].

2.3.2 Support Vector Machines

Another machine learning algorithm that one can use to classify classes is the SVM technique. SVM is a type of supervised machine learning algorithms that is used for classification and regression tasks. This technique was firstly introduced in 1992 by [8]. In the context of classification, this technique attempts to separate classes by using the concept of separating hyperplane (or decision boundary). To understand how SVM work, one may look in Figure 2.3. The SVM can also be used to solve both linear and non-linear problems. The decision boundary tries to separate the data in

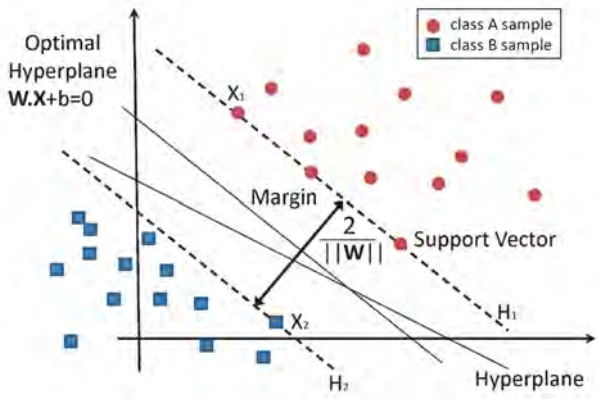


FIGURE 2.3: Binary Classification using SVM [20].

a feature space. Suppose we are given some input dataset $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \cdots, \boldsymbol{x_n})$, $\boldsymbol{x_i} \in \mathbb{R}^m$ and the target labels $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$, where $y_i \in \{-1, 1\}$. If the given dataset can be linearly separable, then the hyperplane can be expressed in the form

$$g(\boldsymbol{x}) = \boldsymbol{\beta}^T \boldsymbol{x} + b \tag{2.12}$$

Where $\boldsymbol{\beta} \in \mathbb{R}^m$ is the weight vector and $\beta \in \mathbb{R}$ is a bias term. Then all points are correctly classified by the separate hyperplane if

$$y_i(\boldsymbol{\beta}^T \boldsymbol{x} + b); \qquad \forall i \tag{2.13}$$

SVM attempts to maximise the distance between the closest training data points from the hyperplane. However, there are various possible hyperplanes that can be created for a dataset. Thus, in order to obtain the optimal hyperplane (the hyperplane that best separate the two classes), the Eq.2.12 should be rewritten in primal form using the Quadratic Programming (QP) [10, 13].

$$\underset{\boldsymbol{\beta}}{\text{maximise}} \quad \frac{2}{\|\boldsymbol{\beta}\|} \tag{2.14}$$

This maximisation problem is equivalent to minimising

$$\frac{1}{2}\boldsymbol{\beta}^T \boldsymbol{\beta}; \quad \text{subject to} \quad y_i(\boldsymbol{\beta}^T \boldsymbol{x} + b) \geq 1, \ \forall i \tag{2.15}$$

The above QP minimisation problem formulated can be computed using the Lagrangian dual maximisation formulation

$$\sum_{i=1}^{n} \omega_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \omega_j y_i y_j {\boldsymbol{x}_i}^T \boldsymbol{x}_j \tag{2.16}$$

subject to

$$\omega_i \geq 0, \qquad \sum_{i=1}^{n} \omega_i y_i = 0 \tag{2.17}$$

Where ω_i are Lagrange multipliers. Thus, solving the above problem then $g(\boldsymbol{x})$ becomes:

$$g(\boldsymbol{x}) = \sum_{i} \omega_i y_i ({\boldsymbol{x}_i}^T \boldsymbol{x}) + b \tag{2.18}$$

Where $\boldsymbol{x}_i$ are said to be the support vector points if ω_i are non-zeros.

In a case where the classes are not separable linearly, the hyperplane can be defined by minimising the following function

$$\frac{1}{2} \boldsymbol{\beta}^T \boldsymbol{\beta} + C \sum_{i=1}^{n} \xi_i \tag{2.19}$$

subject to:

$$y_i (\boldsymbol{\beta}^T \boldsymbol{x} + b) \geq 1 - \xi_i, \ \forall i, \xi_i \geq 0 \tag{2.20}$$

Where ξ_i for $i = 1, \cdots, n$ are the slack variables that measures the misclassification error and C is regularisation parameter. The larger value of C may lead to the issue of overfitting [10]. This problem for non-linear separable can also be solved with Lagrangian dual maximisation formulation

$$\sum_{i=1}^{n} \omega_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \omega_j y_i y_j {\boldsymbol{x}_i}^T \boldsymbol{x}_j \tag{2.21}$$

subject to

$$0 \leq \omega_i \leq C, \qquad \sum_{i=1}^{n} \omega_i y_i = 0 \tag{2.22}$$

Non-linear separable problem can also be solved using the kernel trick. Kernels K transforms input data that is non-linearly separable to a new high- dimensional space such that this new mapping is linearly separable [13, 37]. Let $\phi(\cdot) : \chi \mapsto$ Y be a mapping function that map non-linearly separable input data χ to a high-dimensional space Y such that this new space is linearly separable.

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \phi(\boldsymbol{x_i}) \cdot \phi(\boldsymbol{x_j}) \tag{2.23}$$

Where $\cdot$ is the dot product between $\phi(\boldsymbol{x_i})$ and $\phi(\boldsymbol{x_j})$ and K compute dot product between features x_i and x_j mapped into Y. Substituting Eq.2.23 in Eq.2.21 the Lagrangian dual maximisation formulation becomes

$$\sum_{i=1}^{n} \omega_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \omega_j y_i y_j K(\boldsymbol{x_i}, \boldsymbol{x_j}) \tag{2.24}$$

subject to

$$0 \leq \omega_i \leq C, \qquad \sum_{i=1}^{n} \omega_i y_i = 0 \tag{2.25}$$

Thus solving the Lagrangian dual maximisation above then $g(\boldsymbol{x})$ becomes:

$$g(\boldsymbol{x}) = \sum_{i} \omega_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b \tag{2.26}$$

Some of the popular used kernels are, linear, radial basis function, and polynomial [65].

2.3.2.1 Kernel Functions of the SVM

- Linear kernel Function

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i} \cdot \boldsymbol{x_j} \tag{2.27}$$

- Radial Basis Kernel Function (RBF)

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\frac{||\boldsymbol{x_i} - \boldsymbol{x_j}||^2}{2\sigma^2}}, \quad \sigma^2 \in \mathbb{R}^+ \tag{2.28}$$

  where σ is the Gaussian kernel width [7].

- Polynomial Kernel Function (POLY)

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i} \cdot \boldsymbol{x_j} + 1)^d \tag{2.29}$$

where d is the polynomial degree.

2.3.3 Random Forest

Random Forest is an ensemble learning method meaning that it involves a combination of multiple models [9]. RF operates by constructing a number of decision trees $\boldsymbol{D} = (D_1, D_2, \cdots, D_B)$ at training time using Bootstrap Aggregating or bagging and random feature selection. Bagging tries to reduce the variance by averaging the outputs of many classifiers. The RF makes use of many trees and its prediction is based on the average predictions of each component tree D_i [46]. Let $(x_1, x_2, \cdots, x_n)$ be a set of features, then using the bagging method we randomly select m number of features for which $m < n$. It is recommended for the classification problems to create $m = \sqrt{n}$ sub-features for any tree [46]. Then a decision tree is generated from these sets of m features. Below are the steps for constructing RF:

1. Initially take the original data $\{x^{(i)}, y^{(i)}\}_{i=1}^{n}$ of size n.
2. Randomly generate with replacement, B bootstrap samples s_*^k from the original data such that the bootstrap samples have the same size as the original data.
3. Train B number of decision tree models $D_1^*, D_2^*, \cdots, D_B^*$ using bootstrap data $s_*^{(1)}, s_*^{(2)}, \cdots, s_*^{(B)}$ respectively.
4. Output the final predictions based on the aggregation of the majority predictions (votes) from $D_{(i)}^*$ trees for classification.

2.4 Class Imbalance Problem

CIP is a situation where the number of samples in one class is more than that of other classes. As a result, it is advisable before building any machine learning classifier to check how classes are represented relative to one another. In predominant situations where one class is highly dominant compared to the other the issue of

CIP will emerge. Furthermore, some challenges related to class imbalance include that machine learning classifiers tend to be more biased to the majority classes in making predictions and give a misleading overall model accuracy [27]. That is, machine learning algorithms tend to give poor results because they were designed to reduce errors and not taking class distribution into consideration.

2.5 Class Imbalance Problem Solutions

Although CIP is still the main issue for many classification methods, various techniques such as re-sampling [2, 3], ensemble [69, 68], and cost-sensitive learning [33, 69] have been implemented to address the problem. In our study, we focus on re-sampling methods. To address the issue of CIP, different sampling methods have been implemented to alleviate this problem [11, 25]. Hence, this study was limited to only data-level solutions focusing on seven oversampling techniques namely, ROS, and six synthetic oversampling methods; SMOTE, ADASYN, Borderline-SMOTE1, Borderline-SMOTE, SMOTE + Tomek, and SMOTE + ENN. Some of the class-imbalance solutions are summarised in Figure 2.4.

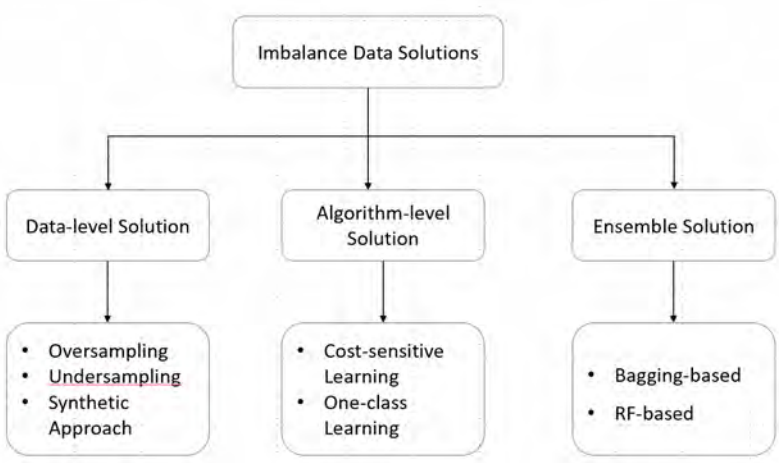


FIGURE 2.4: Class Imbalance Solution Approaches

2.5.1 Random Oversampling

ROS technique works by randomly adding some observations in the minority class by replicating some instances. The major drawback of ROS is that it may lead to

over-fitting due to replications of some information [6] and increase the computational time [25].

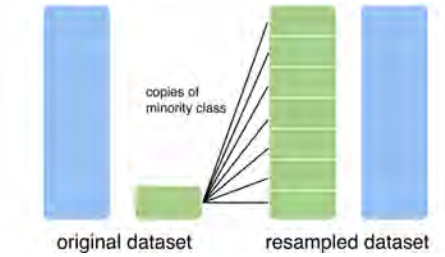


FIGURE 2.5: Data Re-sampling using Random Oversampling [34].

In Figure 2.5 the blue and green bars represent the majority and minority classes respectively. It is clear from the figure that the Random Oversampling algorithm takes samples from the minority class and make copies so that the classes are equally represented.

2.5.2 Synthetic Minority Oversampling Technique

Contrary to ROS, SMOTE is an oversampling technique that operates by creating artificial observations between neighbouring observations in the minority [11]. The advantage of SMOTE over ROS method is the fact that it reduces the issue of over-fitting by the machine learning classifiers [25]. SMOTE produces random synthetic points by finding a straight line between existing points. It then identify the feature vector and its nearest neighbour in a minority class and multiply those new data points by a random number between 0 and 1. Those synthetic data points are added to the original training data set which will be used for training the models. However, the major drawbacks of this technique is that (1) it turns to be ineffective in high dimensional data, (2) SMOTE does not look for neighbouring instances from other classes when it generates synthetic data which could lead to more noisy data due to class overlapping [11].

Let $y_i^{(ori)}$ represents each observation in the minority class, and $y^{(rand)}$ be its randomly selected neighbour. Then SMOTE creates new samples between the original

samples and the selected neighbour using the formula below:

$$\begin{aligned} y^{(new-SMOTE)} &= y_i^{(ori)} + (y^{(rand)} - y_i^{(ori)}) \times \kappa \\ &= y_i^{(ori)} + (y^{(rand)} - y_i^{(ori)}) \times rand(0,1) \end{aligned} \tag{2.30}$$

Where $\kappa = rand(0,1)$ is a random number.

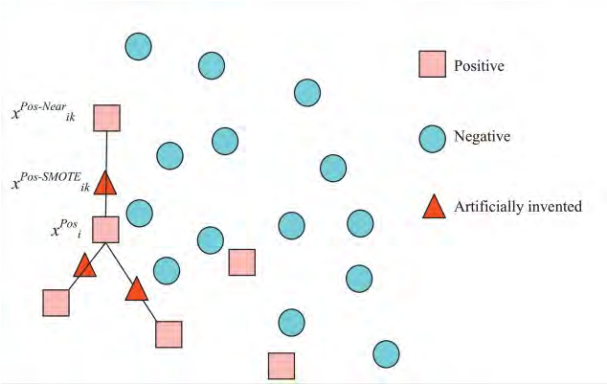


FIGURE 2.6: Synthetic Data Generation using SMOTE [57].

Figure 2.6 depicts the simple SMOTE algorithm with pink squares representing minority (positive) class, blue circles showing the majority class. The orange triangles between pink squares are the synthetic positive samples in the minority class generated by SMOTE. Together the original positive samples and the synthetic samples form the minority class. This will bring balance between the majority and the minority classes.

2.5.3 Adaptive Synthetic

Another synthetic data oversampling technique is called Adaptive Synthetic (ADASYN) sampling approach which was first introduced in [27] with the essence of bringing balance between data classes by adaptively generating data samples in the minority class based on their distribution with specific balance level. This algorithm has two objectives. Firstly, the algorithm is able to identify the required number of samples needed for each sample in the minority class. Second, ADASYN forces the machine learning algorithms to identify or learn the samples that are hard to learn.

2.5.4 Borderline-SMOTE

Borderline-SMOTE is one of the SMOTE variants [24]. However, instead of just creating artificial observations between neighbouring observations in the minority, it puts more focus on the samples closer to the decision boundary. Similar to regular SMOTE, Borderline-SMOTE oversamples the observations in the minority class and their nearest neighbours. There are two parameters variations in Borderline-SMOTE, namely, borderline-1 and borderline-2. In Borderline-SMOTE, samples in the minority class are firstly grouped into three sets, danger, noise, and safe based on the number of nearest neighbours each has from the majority class samples. Let a_i be a point from a minority class A_{min} and A_{maj} be the number of majority class samples that are the k nearest neighbours of point a_i. Then

- The point a_i is said to be noise if all of its k nearest neighbours are from the majority class samples (i.e. $k = A_{maj}$).
- If the half of point point a_i's k nearest neighbours are from the majority class, then the point a_i is said to be in danger (i.e. $\frac{k}{2} \leq A_{maj} < k$).
- If point a_i's all k nearest neighbours are from the minority class then the point a_i is said to be safe (i.e. $0 \leq A_{maj} < \frac{k}{2}$).

Using SMOTE, the points which are classified as the danger data points will be taken as the minority data points and considered for oversampling to create synthetic points. The main distinction between Borderline-SMOTE1 and Borderline-SMOTE2 is that the former creates the data points along the line between the danger points in the minority class and their k nearest neighbours while the latter go a step further by considering both classes when generates the data points.

2.5.5 SMOTE + Tomek Links

One of the major drawbacks of SMOTE is that it does not look for neighbouring instances from other classes when it generates synthetic data which could lead to more noisy data due to class overlapping. For SMOTE + Tomek, the minority class is firstly over-sampled using the SMOTE procedure to balance the classes and then Tomek Links is applied to the over-sampled data to clean the data space [6]. Suppose two data points $p_i \in$ minority class and $p_j \in$ majority class are given and

the distance between p_i and p_j is $\phi(p_i, p_j)$. Then a pair of instances (p_i, p_j) from different classes is said to have Tomek Link if there is no data point p_k such that $\phi(p_i, p_k) < \phi(p_i, p_j)$ or $\phi(p_j, p_k) < \phi(p_i, p_j)$. This method can be applied as either the data cleaning or under-sampling technique. For data cleaning, the data points that form Tomek Links from both classes are eliminated from. As a data cleaning method, the data points from the majority class that form Tomek Links are removed. Figure 2.7 below illustrates the procedure of Tomek Links.

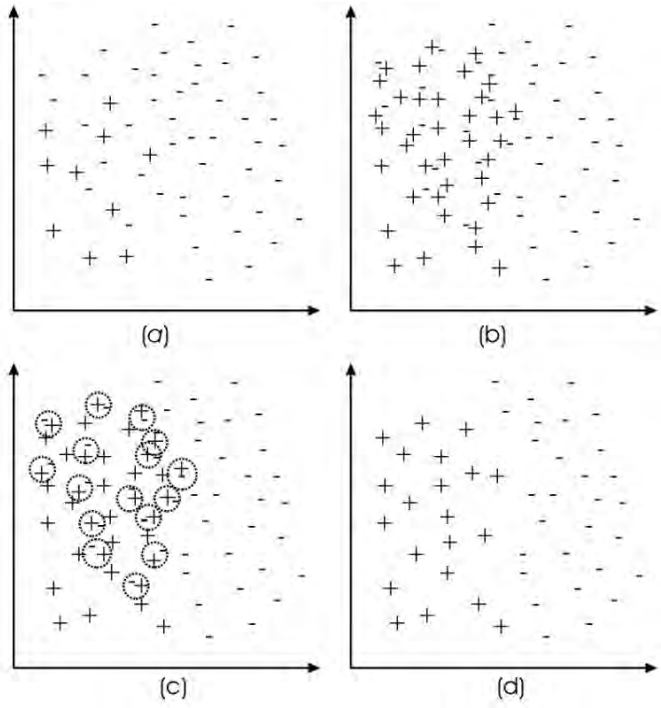


FIGURE 2.7: SMOTE + Tomek Links [6].

From 2.7, the original data is represented by (a) with negative (-) and positive (+) depicts the majority and minority classes respectively. In (b) the original data (a), is over-sampled using SMOTE procedure. The nearest neighbours (circled - and + data points) are selected by using Tomek Links as shown in (c). Lastly, the selected Tomek Links points are then removed from the data depicted in (d)

2.5.6 SMOTE + ENN

By using the same idea as SMOTE + Tomek, the SMOTE + ENN attempts to provides more deeper cleaning space by applying Wilson's Edited Nearest Neighbour rule (ENN). Firstly the data (minority class) is oversampled through SMOTE then the ENN followed [64]. Essentially, ENN eliminates all the data points in both minority and majority classes that are misclassified by at least 2 out 3 of its nearest neighbours. [6] highlighted that the ENN tends to remove more data points as compared to Tomek Links and hence provide more cleaning data.

2.6 Stratified K-Fold Cross-Validation

When we evaluate a machine learning performance, we usually divide the data into train and one test data to evaluate model performance. The problem with this split is that the model can be unreliable because it can perform differently on different test datasets. To overcome this, we can use the Stratified K-Fold Cross-Validation (stratified K-fold CV) [22]. It preserves the equal percentage of samples in each fold with respect to minority class and can evaluate the robustness of the model since it partitions the training data into k-groups, giving each piece of data chance to be used in both test and training. In a K-fold CV method, (1) original data set is randomly divided into roughly equal folds (groups); (2) randomly shuffle the train data; divide the training dataset into k number of groups; (3) take each group as a train data and build a model and use the testing dataset to validate the model (4) repeat the process until all the k groups have been used and average the accuracy of each model as one overall accuracy. For each round, each fold is used as either a train or validation data exactly once. More formally, suppose that we are given a dataset $\boldsymbol{y}$, then the main objective is to split a dataset into k groups, such that the test error CV for each group/fold is given by:

$$CV = \sum_{i=1}^{m} e_i = \sum_{i=1}^{m} (y_i \neq \hat{y}_i), \qquad \forall i = 1, \cdots, m \tag{2.31}$$

which implies that the average test error for all groups is given by:

$$CV_{average} = \frac{1}{k}\sum_{j=1}^{k} CV_{(j)}, \qquad \forall j = 1, \cdots, k \tag{2.32}$$

Where e_i is the misclassification error between the actual y_i and predicted $\hat{y}_i$ classes and m is the total number of samples in each fold. In addition, k is the total number of folds. Then we train a machine learning classifier on each group $j = 1, \cdots k$ and evaluate on the remaining fold .

2.6.1 Stratified K-Fold Cross-Validation for Imbalance Classes

[54] addressed the importance of correctly performing the K-fold cross-validation when the data is imbalanced classes. For dataset with equally distributed classes or for traditional cross-validation, the original train data is usually divided randomly into equal folds to evaluate the performance of the model. In cases where the imbalance exists in the given data, the re-sampling has to be implemented within the cross-validation to ensure that only the training samples are over-sampled. This procedure will avoid the problem of model overfitting and overoptimism. Overoptimism occurs when re-sampling is performed before cross-validation resulting in having similar samples in both training and validation data sets. Figure 2.8 shows how the K-fold cross-validation should be implemented if the class imbalance exists in a dataset.

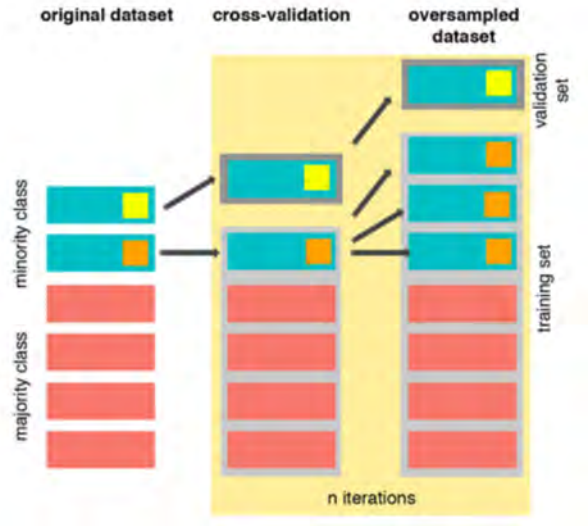


FIGURE 2.8: Proper K-fold Cross-Validation [41].

Figure 2.8 shows that the cross-validation is implemented such that at each iteration some samples in the minority class (blue bar with small yellow bar inside) are held-out for validation (not over-sampled).

2.7 Evaluation Metrics Review

After developing a predictive model, different measures are required to assess model performance. Some of the most commonly used are accuracy, F-score, and Geometric Mean. The ROC-AUC curve will be used to evaluate the performance of the classifiers.

2.7.1 Receiver Operating Characteristic and Area Under the Curve

ROC is a graph which plots the trade-off between the sensitivity/True Positive rate (TP_{rate}) against False Positive rate (FP_{rate}) based on diverse thresholds [25, 61]. It plots TP_{rate} on the y-axis and FP_{rate} on the x-axis. The AUC scores are bounded between a range of 0 and 1. An AUC value equal to 1 indicates a perfect classifier and a value closer to one indicate a good classifier. On the other hand, if an AUC score is equal to 0.5 then a classifier is equal to a random classier. The AUC score

can be evaluated using the following formula:

$$AUC = \frac{1}{2}(1 - TP_{rate} - FP_{rate}) \tag{2.33}$$

Where,

$$TP_{rate} = \frac{TP}{TP + FN} \tag{2.34}$$

and

$$FP_{rate} = \frac{FP}{FP + FN} \tag{2.35}$$

Where True Positive (TP) are the positive samples that are correctly classified as positive and True Negative (TN) are negative samples that are correctly classified as negative. Furthermore, False Positive (FP) refers to the samples that are actual negative but are classified as positive and False Negative (FN) refers to the samples that are actual positive but are classified as negative [25, 61]. An example ROC-AUC curve is depicted in Figure 2.9.

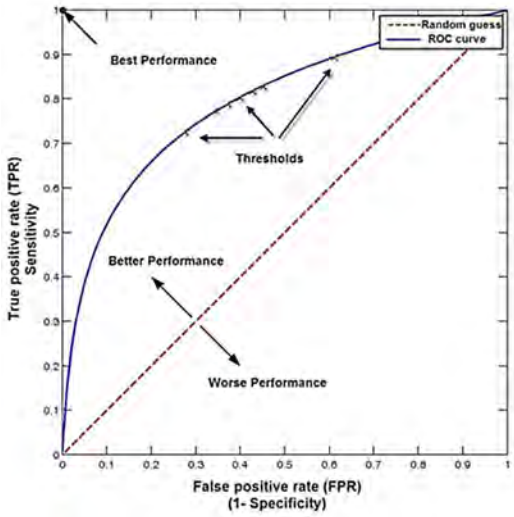


FIGURE 2.9: A simple representation of a ROC curve [26].

From the plot, it is clear that as the curve goes close to 1 (top left corner) then the best prediction model. The red line represents the roc curve of a random guess classifier.

2.8 Related Works On Churn Prediction

[2]This section provides a review of the literature associated with the use of different supervised machine learning algorithms in the prediction of customer churn. Machine learning techniques can be very useful in detecting customer satisfaction in telecommunications due to the vast amount of data generated daily in the telecommunications industries. In order to address and predict customer turnover or churn, various strategies including data mining methods have been implemented to extract knowledge.

Recent studies have shown that machine learning techniques can deliver better results for the prediction of churn [4, 25, 66]. In order to build a machine learning binary classifier, [17] used comprehensive SVM to predict customer churn in the American bank dataset. Four SVM models were implemented and compared, namely; SVM, SVM with sensitivity only, SVM with Naive Bayes (NB) using sensitivity alone, SVM with Naive Bayes of reduced features, NBTree. or feature selection, and SVM with RFE. Dataset about bank credit card customers was obtained from Latin American bank in 2004. The dataset comprised of a total sample of 13 812 and 22 features. The data has a class distribution proportion of about 7% churner customers compared to loyal customers, which means the dataset was highly imbalanced. Four class balancing techniques namely; SMOTE, RUS, ROS, and combined under-sampling and over-sampling were employed to balance the classes. The data was split into the ratio of 80:20 and a 10-fold cross-validation method was employed on the train data set and test data was used to validate the overall model accuracy. Using all the features on imbalance data the accuracy and sensitivity were high for NBTree and (SVM + NBTree) respectively. The results were lower with reduced features. The best model was achieved by (SVM + NBTree) with SMOTE with accuracy and sensitivity of more than 85% respectively.

To address the problem of data imbalance classes, [23] compares the performance

[2]The discussion in this section is informed by a discussion that appears in (Perlate Diala), (Prediction of depression among university students using machine learning algorithms), University of the Witwatersrand, 2018. All material from external sources has been referenced.

of various re-sampling techniques using the mobile telecommunication customer churn data. The study used three different feature selection methods; Standardised Regression Coefficients (SRC), Relative Weight (RW) and Random Forest (RF). All these selection methods were evaluated and compared on three sampling techniques namely; SMOTE, random under-sampling, and random over-sampling. Thus, 12 models were built. After data preprocessing step, 78 411 observations were used for the experiment. About 13% represents the not-loss class which implies that the class ratios were about 1:6.7. Hence the class distribution was highly imbalanced. The data set was split into the ratio of 80:20, 80% training and 20% testing for all the algorithms. To evaluate the model performances, five metrics were utilised; accuracy, recall, precision, F-score and cost which determine the value and real investment. The results showed that random forest with SRC over-sampling on the original dataset outperformed all other models.

Churn prediction using different machine learning techniques was studied and compared in [4]. A number of classifiers namely; LR, SVM, ANN, C4.5, DT, RF, AdaBoost, Gradient Boosting were compared with fuzzy models which are models that are able to deal with noise data. Therefore, the study compared various Fuzzy classifiers, namely; FuzzyNN, VaguelyQuantifiedQNN, FuzzyRoughNN, OWANN. The dataset used was from a telecom company operating in South Asia and data was large with 600 000 observations and 722 features. The data was highly imbalanced with only 9% represents the churners group. The data was divided into a ratio of 80:20. Since the data was noisy, the instances with noisy data were removed. Most relevant features were obtained by just using domain knowledge. All the features were numeric and min-max data normalisation technique was used to normalise the data. The models performance were evaluated using recall, precision, AUC, and lift curve with RF outperformed logistic regression and other models.

In addition, the study by [32] applied deep 1D Convolution Neural Network (CNN) to address the issue of customer churn. The features were manually reduced from and ultimately using the Least Absolute Shrinkage and Selection Operator (LASSO) feature selection method. The publicly available Telecom dataset from Orange company was used. Since the data consisted of many missing values, the features that

have more than 30% of missing values were removed from the data. The standard AUC metric with 10-fold cross-validation was used. Although the churn rate was very low, the re-sampling techniques were not utilised. The model consists of one convolutional layer and the ReLU activation function was used. The sigmoid activation was used in the output layer and the Adam optimiser was used to optimising the weights as well as compiling the model. The cross-entropy was used as the objective function. The CNN manages to achieve the accuracy of 98.85%. The CNN was compared to some classic machine learning algorithms and it outperformed all of them.

Lastly, in [22] it was argued that the SVM technique together with ROC-AUC metric can provide a good model generalisation performance. This means that a good choice of parameters could be helpful when building predictive models in terms of improving the overall performance of models. The SVM-RBF kernel function showed good performance compared to other kernel functions (for example, linear kernel). In addition, the results in different studies showed that the machine learning models with re-sampling techniques outperform models on imbalanced data. However, according to [4] very few studies where machine learning techniques were utilised to solve customer churn prediction did not put more focus on TP rate and ROC-AUC. Moreover, the study has also emphasised the importance of handling the CIP before building machine learning models for customer churn prediction.

The existing studies have shown that the performance of machine learning classifiers can be limited by various issues including small datasets, preprocessing, feature selection, handling class imbalance and choice of evaluation metric. For example, in [28] the problem of imbalanced classes was not handled. Dimensionality reduction, is one of the important step in building machine learning models. However, existing studies have been limited to linear methods such as PCA [38]. There is limited literature for unsupervised dimensionality reduction related to telecommunication. To avoid this potential issue of linearity, we perform dimensionality reduction based on linear and non-linear autoencoders. Although the issue of class imbalance has been addressed in many studies, very few studies in

telecom have used more advanced sampling techniques such as SMOTE+Tomek and SMOTE+ENN.

We did not find any study related to telecommunication sector that investigated the dimensionality reduction based on non-linear methods. With that said, in this research we investigate non-linear dimensionality reduction based on autoencoders using various functions, i.e, linear, sigmoid, and relu. In addition to that, we address the issue of class imbalance which is common with telecommunication data by investigating the performance of various oversampling, and oversampling followed by data cleaning techniques. The performance of these techniques is evaluated on machine learning models, i.e., LR, SVM, and RF.

2.9 Conclusion

Supervised machine learning techniques become popular in many areas including health care, business, and telecommunication domains. In this chapter, we discussed the various dimensionality reduction methods as well as the supervised machine learning techniques. The issue of CIP problem was also addressed and various over-sampling techniques were discussed. Different evaluation metrics have been used to evaluate the performance of the prediction models. Lastly, we reviewed the literature linked to how different supervised machine learning algorithms have been used in customer churn prediction.

Chapter 3

Research Methodology

3.1 Introduction

This chapter presents the methodology that was used to carry out this research. The research design was divided into six phases from the data collection, data pre-processing, features engineering, oversampling, to model training and evaluations. The classification framework that we have used is given in Figure 3.1. All analyses and implementations were performed using the open-source software Python 3.6 (Jupyter Notebook).

3.2 Research design

This section provides a framework that we used to implement all the models; LR, SVM, and RF. The pre-processing of data consisted of two phases, i.e., missing values handling and data standardisation. Moreover, To avoid data leakage issues, we perform data splitting before feature engineering and oversampling data. The stratified train-test split provided by sklearn library in python was used to split the data into the training data and testing data. After splitting the data, feature engineering based on autoencoders using various activation functions namely, linear, sigmoid, and ReLU were trained to address the problem of high dimensional data. These autoencoders were performed only on the training dataset and then evaluation on the test data was done in order to have the same dimension. We have also handled the issue of class imbalance through utilizing seven oversampling techniques; ROS, SMOTE, ADASYN, Borderline-SMOTE1, Borderline-SMOTE2, SMOTE + Tomek,

and SMOTE + ENN. The 5-fold cross-validation combined with oversampling techniques was used to train models on the training data and the test data was used to evaluate the final performance of the model. To evaluate the performance of the models, we have used ROC-AUC. In addition, we have built all the models under the following settings.

- Category 1: Baseline models were built on the original datasets with missing values filled with median and mode for continuous and categorical features respectively.
- Category 2: Build models based on features extracted by undercomplete autoencoder using linear activation functions.
- Category 3: Build models based on features extracted by undercomplete autoencoder using sigmoid activation functions.
- Category 4: Build models based on features extracted by sparse autoencoder using non-linear relu-based regularised activation function.
- Category 5: Build models based on features extracted by deep autoencoder.

The framework we have used to build machine learning classifiers is displayed in Figure 3.1 and the details of each component of the experimental setup are provided below.

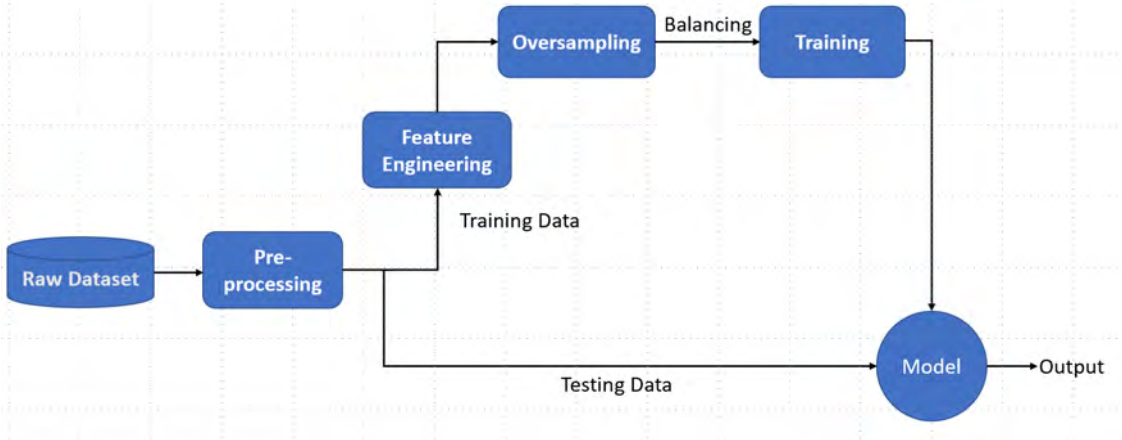


FIGURE 3.1: Machine Learning Classification Framework

3.3 Data

3.3.1 Data Description

In this research, we have used two publicly available telecommunication customer churn datasets, which we refer to as dataset 1 and dataset 2 hereafter. Dataset 1 was obtained from IBM sample datasets [30]. The dataset consists of 7043 instances and 21 features. Moreover, the data heterogeneous is meaning it contains mixed-data types; categorical and numeric features. In total it consists of 3 numeric features and 18 categorical features. The data contains customer information including demographic, service signed for such as the contract term of the customer, and tech support and account information such as total amount charged to the customer. The ratio between the churn and non-churn class is 1:2.8 with only 26.5% representing churn class. Dataset 2 was obtained from Orange, a telecommunication corporation [1]. The dataset consists of 40000 instances and 230 features, and the dataset is highly imbalanced with 7.4% representing churn class as compared to dataset 1. The names of the features in dataset 2 are anonymized. In total it consists of 190 numeric and 40 categorical features.

Table 3.1 displays the data description summary where IR refers to the imbalance ratio between churn class and non-churn class and labels represent the class of each sample. The class imbalance can be computed using formula below [15]:

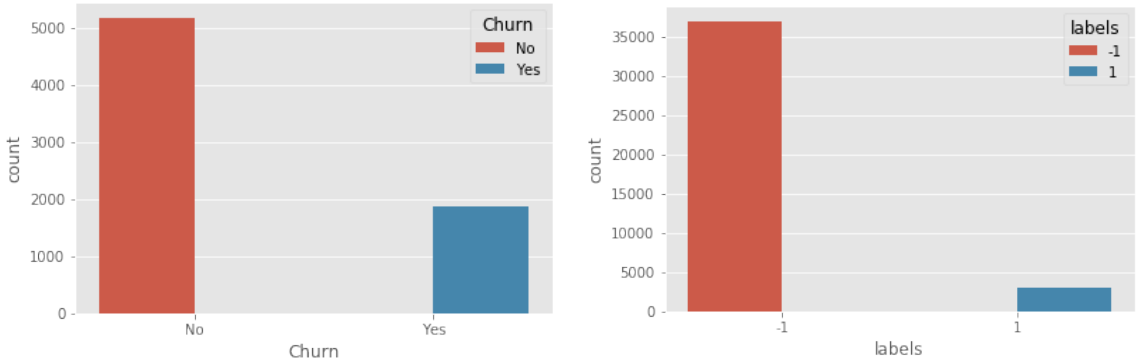
$$IR = \frac{n_B}{n_A} \tag{3.1}$$

Where n_A and n_B are the total number of minority and majority class samples respectively.

TABLE 3.1: Summary Description of the Datasets.

Data Source	Number of Samples	Number of Features	Labels	IR
Dataset 1 [30]	7043	21	{Yes, No}	2.77
Dataset 2 [1]	40000	230	{-1, 1}	12.51

The distribution of churn and non-churn classes for each dataset is shown in Figure 3.2. The red bar is the non-churn class and blue is churn class. The graphs below show that the two classes are not fairly distributed.



(A) Class Distribution for Dataset 1

(B) Class Distribution for Dataset 2

FIGURE 3.2: Class distribution of the original datasets.

3.3.2 Data Preprocessing

3.3.2.1 Missing Values Handling

Missing values or missing data refer to the missing records in the dataset. Data may have missing values due to various reasons such as, incorrect measurement and manually entering data. In the telecommunication industry, datasets tend to have a lot of noisy values such as missing values, inappropriate values like NULL and special characters. Dataset 1 has no missing values, therefore, no missing values handling technique was employed. The customerID was the only feature that was removed from the data since it has no impact on churn retention. In dataset 2 there were lots of features with missing values. The features with at least 70% of missing values were removed. The categorical features with more than 100 categories were removed from the data. Furthermore, after removing those features we remain with 75 features. With the remaining features, we have filled missing values for continuous features with the maximum value for each feature. For categorical features, however, due to higher cardinality for some features, instead of using OneHotEncoding, we opted for encoding categorical values with their frequencies to avoid a large number of features and it can be too computationally expensive.

In addition, we fill the missing values for categorical features with zero. The distribution of missing values for each feature of the dataset 2 where each vertical line represents a feature (e.g. first vertical line corresponds to Var1) is depicted in Figure 3.3. The x-axis represents features and y-axis shows the percentage of missing values. The values of 1.0 means that the feature has no missing values and 0.0 means that the feature column is empty as is shown in Figure 3.3 below.

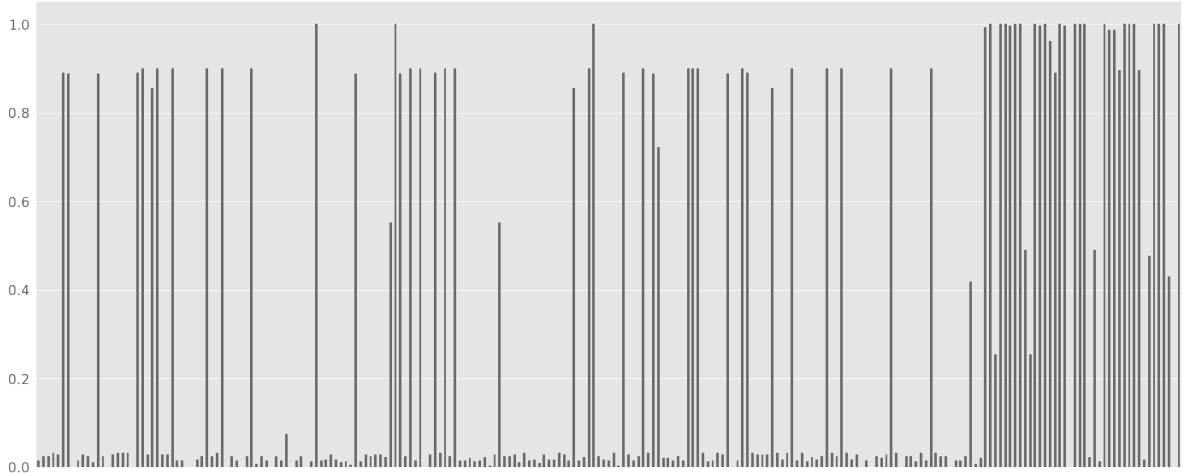


FIGURE 3.3: The Distribution of Missing Values for each Feature.

3.3.2.2 Standardisation of the Data

In order to avoid different scales of feature values, we need to normalise all the continuous feature values to the same scale. The Min-Max Scalar standardisation technique is used to normalise all feature values to the range of $[0, 1]$. The Min-Max Scalar equation is given by:

$$u_i^{'} = \frac{u_i - min(u)}{max(u) - min(u)} \tag{3.2}$$

Where $min(u)$ and $max(u)$ are the minimum and maximum values of the ith feature and $u_i^{'}$ is the normalised value [4].

3.4 Methods

3.4.1 Data Split

Both datasets were split into 80% training data and 20% testing data using a stratified train-test split mechanism provided in Python Sklearn Model Selection library. The main objective of a stratified train-test split is to preserve the same proportions of class samples for train and test datasets as the original class. The training data was further divided into 5 folds using 5-fold cross-validation to develop and evaluate the model performances on the train data. The test data was used in a final step to assess the overall performance of the models.

3.4.2 Feature Engineering

Feature engineering is a process of creating new features using the existing ones by transforming these features into a new feature space. The objective of performing feature engineering is to construct new features that could improve machine learning model performance [43, 63]. Feature engineering helps in the sense that it eliminates model over-fitting, reduces model complexity, and frees up memory. Although it sometimes vanishes model interpretability, it allows the algorithm to learn faster. Lastly, it eliminates unnecessary covariance between the features [39]. In this study, feature engineering was based on dimensionality reduction using autoencoders with various activation functions i.e., linear autoencoder, non-linear sigmoid-based autoencoder, non-linear relu-based regularised autoencoder, and deep autoencoder.

3.4.2.1 Dimensionality Reduction

One of the major problems with the telecommunication datasets is the large feature space. To address this problem, we have implemented the autoencoders technique for dimensionality reduction under three settings; autoencoders based on linear activation, non-linear sigmoid, and non-linear ReLU. For dataset 1, the autoencoder network was trained using two parameters; batch_size of 32 and 50 epochs. The shape of the input data was the size of the input features ($size\_input = 23$) and the hidden_layer size of 16. For dataset 2, the network was trained with the following

parameters; $batch\_size = 32$ and $number\_of\_epochs = 50$. Moreover, the autoencoder architecture consisted of $input\_size = number\_of\_inputfeatures = 76$ and one hidden layer of size 32. The size of the hidden layer (code) was selected by evaluating the performance of the autoencoders on different sizes of feature sets. The model was compiled using two arguments; Adam optimiser for learning and mean squared error (MSE) for evaluating the performance. All the autoencoders in dataset 1 have the same architecture except activation functions. The details of each layer for this autoencoder are given below.

The linear autoencoder architecture implemented for dataset 1 consists of three layers; input, code, output layers. The first layer is the input layer which contains the input features. Then the encoder maps the input features using linear activation function to a lower dimension which is the second layer. The second layer consists of a dimension of 16 number of features (23->[16]->23). The final layer, which is the third layer reconstructed from the second layer using decoder with linear activation function, also has 23 features, i.e., the same as the first layer. For non-linear sigmoid autoencoder, however, similar architecture was used. The main difference is that the latter used sigmoid functions in both layers. Furthermore, for non-linear sparse autoencoder with ReLU, we have used ReLU activation function in the second layer and sigmoid function in the output layer.

For dataset 2, the autoencoder with three layers was also used. However, for this case, the architecture has 76 inputs in its first layer. For linear autoencoder, the linear activation function was used for both the first and second layers. The data was compressed from 76 (input layer) to 32 features (second layer) using encoder with linear function and then was reconstructed to the 76 features (third layer) using the decoder with the linear activation function (76->[32]->76). In addition, we have used similar architecture for non-linear autoencoder with sigmoid functions used for both second and third layers. Lastly, with respect to a non-linear autoencoder with ReLU, the sigmoid function was used in the output layer.

With respect to deep autoencoder for dataset 1, the network consisted of five layers; input layer, hidden layer, code, hidden layer, output layer was implemented (23->[19->16->19]->23). The layers were connected using relu activation function except for the final layer which used sigmoid function. For dataset 2, we have used more layers. The architecture contained seven layers; input layer, hidden layer, hidden layer, code, hidden layer, hidden layer, output layer (76->[62->46->32->46->62]->76). Also relu activation function was used to connect layers. For the final layer, sigmoid function was used.

3.4.3 Class Imbalance

It is common that most of the datasets in the telecommunication domain have imbalanced classes due to the fact that the rate of customers that are leaving is much less than those who retained. The problem arising with imbalance class data is that the machine learning models only pay attention to the highly represented class overlooking the minority class which will lead to poor prediction accuracy. The two datasets we worked on had a class imbalance problem since the customer churn rate was about 26.5% and 7.4% on dataset 1 and dataset 2 respectively. To address this issue, we have implemented seven oversampling techniques namely; ROS, SMOTE, ADASYN, Borderline-SMOTE1, Borderline-SMOTE2, SMOTE + Tomek, and SMOTE + ENN. To avoid the issue of overoptimistic and overfitting, these oversampling methods were only applied to the training dataset within 5-fold cross-validation. This implies that oversampling will only be applied to the first 4 folds and then evaluated on the fifth fold. The following table provides the details of the oversampling methods used. The parameters used by oversampling techniques are the number of nearest neighbours k needed to generate artificial samples and the number of nearest neighbours m needed to decide whether the point is in danger for Borderline-SMOTE.

TABLE 3.2: Oversampling techniques implemented to address the class imbalance in this study.

Oversampling technique	Description	Parameters
ROS	Random Oversampling	-
SMOTE	Synthetic Minority Oversampling Technique	$k = 5$
ADASYN	Adaptive Synthetic Approach	$k = 5$
Borderline-SMOTE		
	borderline-1	$m = 10$
	borderline-2	$m = 10$
SMOTE + ENN	SMOTE followed by Edited Nearest Neighbours	-
SMOTE+Tomek	SMOTE followed by Tomek Links	-

3.4.4 Churn Prediction Models

After the data pre-processing phase was done, we performed the dimensionality reduction on the training data using various autoencoders; linear autoencoder, non-linear based on sigmoid, sparse autoencoder using ReLU, and deep autoencoder. To address the issue of class imbalance distribution, we investigated seven oversampling techniques to balance the classes. For the process of implementing models, every model was built on the combination of every dimensionality reduction coupled with the oversampling method. Furthermore, 5-fold cross-validation was used to train and evaluate the performance of the models for the train datasets. The test data was left out during the model training process and was only used to evaluate the overall performance at the last stage (See Figure 3.1). For model building, each classifier (LR, SVM, and RF) was implemented with the following parameter settings.

- Logistic Regression: We have fitted LR with the $L2$ regularisation parameter to prevent overfitting and the default inverse regularization parameter set to $C = 1.0$.
- Support Vector Machine: The SVM model was fitted with radial basis function (kernel) and the regularisation penalty term of 1 (i.e. $C = 1.0$).
- Random Forest: For the RF classifier, we have fitted the model with the following parameters; the number of trees was set to 100 ($n\_estimators = 100$).

3.4.5 Implementation

All the analyses and implementations of machine learning classifiers were performed using Dell Core i7 with Intel(R) Core(TM) i7-7Y75 CPU at 1.30GHz processor and 16GB Ram. We have utilised various open-source libraries provided in Python. For data importing and manipulations we have used Numpy [47] and Pandas [42]. Moreover, Matplotlib [29] (with Seaborn, for nice plots) was used for data visualisation. In addition, Keras library [12] was used to implement the autoencoders dimensionality reduction techniques. For data-level solution, imbalanced-learn [36] was used to implement all the oversampling techniques. Lastly, sklearn [48] library was used to build machine learning models (i.e. LR, SVM, and RF).

3.5 Analysis

This section identifies the metric used to assess the model performances, the procedures used to describe the baseline models as well as the comparison of machine learning models.

3.5.1 Baselines

In order to evaluate the impact of dimensionality reduction based on autoencoder and oversampling techniques for customer churn prediction, we need a baseline to make comparisons. For this purpose, as a baseline, we train all the selected classifiers; LR, SVM, and RF on the original datasets without performing dimensionality

reduction and oversampling. The baseline models were also trained using stratified 5-fold cross-validation. In addition, the overall performance of these baseline models was evaluated on test data using AUC-ROC.

3.5.2 Evaluation Metric

To evaluate how well our models are performing, we need to use metric that is suitable for the classification problem. Once all the predictive models are implemented and tested, we need to evaluate their performances. To do that, we have used the ROC-AUC curve which plots the trade-off between TPR and FPR. Furthermore, a good classifier should have an AUC score closer to 1, whereas a random classifier would have an AUC score of 0.5.

3.6 Conclusion

In this chapter, we have discussed the methodology that was adopted to carry out this research. We have started by providing the research design and framework to be used. The oversampling and dimensionality reduction techniques to address the class imbalance problem and the curse of high dimensional features space were implemented. In addition, machine learning classifiers used for customer churn prediction were defined.

Chapter 4

Results and Discussion

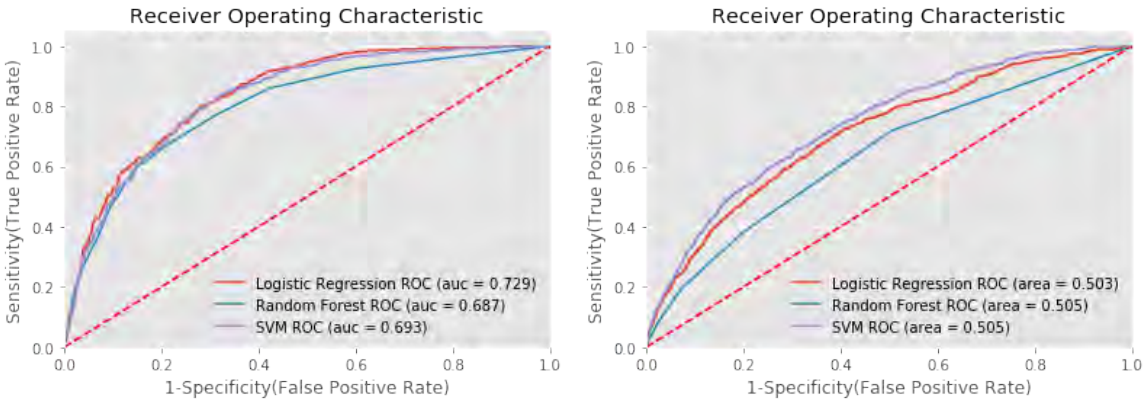
4.1 Introduction

In this chapter we present the results of various techniques using two publicly available telecommunication datasets. This chapter is divided into four sections. We start by providing the performance of the baseline model in section 1. In section 2, we compare the performance of various oversampling techniques, dimensionality reductions and the machine learning classifiers. Section 3 gives the overall performances of the classifiers. Lastly, section 4 presents a brief summary.

4.2 Baseline Results

This section provides the results of the baseline models (without dimensionality reduction and oversampling techniques). The results of these prediction models are based on 5-fold cross-validation as discussed in the previous sections. Figure 4.1 illustrates the performance of the prediction models (LR, SVM, and RF) based on AUC measure. The results also reveal that LR outperformed other classifiers for smaller dataset (i.e. dataset 1) and larger dataset (i.e. dataset 2). The red dotted line represents the performance results of a random classifier with AUC=0.50. The prediction models performed significantly better on dataset 1 and worst on dataset 2. From Figure 4.1a, it can be observed that all classifiers outperformed the random classifier (with AUC=0.50). In comparison, the LR achieved the highest score (AUC=0.729) outperformed both SVM and RF. Furthermore, the SVM performed better than RF with the former achieved AUC=0.693 and the latter obtained AUC=0.687.

On the other hand, the prediction models on dataset 2 performed slightly higher than the random classifier. However, all these baseline models obtained AUC score of less than 0.51. Contrary to dataset 1, the LR with AUC=0.5 was outperformed by both SVM and RF classifiers for dataset 2 as displayed in Figure 4.1b. Interestingly, both classifiers scored AUC=0.505.



(A) Performance of Baseline Models on Dataset 1

(B) Performance of Baseline Models on Dataset 2

FIGURE 4.1: ROC-AUC Curve for each Baseline Classification Model

4.3 Comparison of Dimensionality Reduction, Oversampling and Classification Techniques

This section presents the results of the dimensionality reduction and oversampling techniques. To evaluate or study the impact of the dimensionality reduction and oversampling techniques on churn prediction, we investigated two solutions; dimensionality reduction and oversampling techniques. Moreover, for dimensionality reduction, we have compared autoencoders using various activation functions, i.e. linear, sigmoid, non-linear based on ReLU, and deep autoecoder. To address the class imbalance problem, we have compared the performance of various oversampling techniques namely; ROS, SMOTE, ADASYN, Borderline-SMOTE1, Borderline-SMOTE2, SMOTE + Tomek, and SMOT + ENN. The performance of these techniques was evaluated using three classifiers (i.e. LR, SVM, and RF). Thus, we need to compare the results of these solutions to the baselines.

4.3.1 Comparison of the Performance of Classifiers and Oversampling Methods Evaluated on Features Reduced by Linear Autoencoder

The performance of various dimensionality reduction and oversampling techniques are shown in Table 4.1. The best performing strategy or a combination of techniques are shown in bold. As shown in Table 4.1, when applying linear autoencoder with one hidden layer on dataset 1, some of the selected classifiers performed well as compared to the baselines as shown in Figure 4.1a. Although these oversampling techniques showed effective improved results, when applied to the dataset from linear autoencoder dimensionality reduction, a simple naive ROS outperformed all the complex synthetic oversampling methods when applied with LR. ROS outperformed synthetic methods across all the classifiers with the LR obtaining the highest AUC score. This indicates that more advanced synthetic oversampling methods could have created more overlapping classes and generated noisy data points. Furthermore, both LR and SVM obtained the AUC of more than 0.73 across all resampling techniques for dataset 1. Although RF classifier was outperformed by others, its results have significantly improved when dataset is over-sampled first by SMOTE and then cleaning procedure followed (i.e. SMOTE + ENN). The model achieves the AUC score of 0.724 compared to ROS with AUC of 0.643. Additionally, two advanced versions of SMOTE; SMOTE + Tomek and SMOTE + ENN, which try to clean class overlapping created by SMOTE fail to outperform SMOTE when trained with SVM. From Table 4.1 it can be observed that ROS performed the best with AUC=0.747 followed by AUC=0.746 and SMOTE+ENN performed worst with AUC=0.737 when trained with the SVM.

For dataset 2, the best preforming classifier was LR (AUC=0.649). As reported in Table 4.1, the results indicated that ROS outperformed all the synthetic oversampling techniques. Similar to dataset 1, RF showed best results when coupled by SMOTE+ENN and SVM get good results when trained on dataset 2 with ROS. In addition, both Borderline-SMOTE1 and Borderline-SMOTE2 performed better than SMOTE+Tomek and SMOTE+ENN. Thus, when we build an LR classifier using a linear autoencoder, a naive ROS is recommended.

TABLE 4.1: The performance evaluation of machine learning classifiers, oversampling techniques and linear autoencoder based on AUC for dataset 1 and 2.

Linear AutoEncoder	Oversampling Technique	LR	SVM	RF
Dataset 1	ROS	**0.756**	0.747	0.643
	SMOTE	0.750	0.746	0.659
	ADASYN	0.753	0.743	0.665
	Borderline-SMOTE1	0.752	0.742	0.662
	Borderline-SMOTE2	0.753	0.741	0.679
	SMOTE + Tomek	0.749	0.745	0.675
	SMOTE + ENN	0.732	0.737	0.724
Dataset 2	ROS	**0.649**	0.634	0.505
	SMOTE	0.644	0.610	0.534
	ADASYN	0.645	0.604	0.525
	Borderline-SMOTE1	0.645	0.611	0.525
	Borderline-SMOTE2	0.640	0.604	0.536
	SMOTE + Tomek	0.645	0.604	0.524
	SMOTE + ENN	0.616	0.608	0.554

Figure 4.2 below illustrates the aggregated AUC scores between each classifier against the baseline results. Moreover, a negative bar plot indicates that the results of the baseline were better than of the classifiers trained on the dataset resulted from linear autoencoder and oversampling techniques. It is clear from Figure 4.2 that LR and SVM benefited more from dimensionality reduction and oversampling with each classifier shows an increased AUC score. Although RF does not show much improvement, it benefited more when combined with SMOTE+ENN followed by Borderline-SMOTE2.

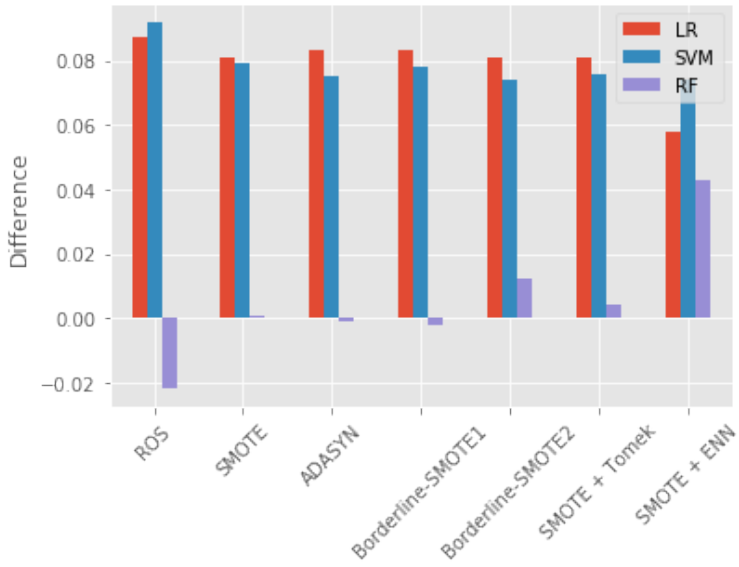


FIGURE 4.2: The difference in performance evaluation of various classifiers on datasets from dimensionality reduction using linear autoencoder and oversampling methods against the baseline results.

4.3.2 Comparison of the Performance of Classifiers and Oversampling Methods Evaluated on Features Reduced by Autoencoder using Sigmoid Functions

With respect to the autoencoder dimensionality reduction based on non-linear sigmoid, all the oversampling methods showed potentials to improve the prediction performance of LR and SVM classifiers. On contrary to the linear autoencoder, LR showed the best results when trained with the dataset 1 over-sampled with Borderline-SMOTE2. Again the classifier achieved the lower AUC score when SMOTE and SMOTE+Tomek were used and interestingly, these two methods were outperformed by ROS. Although LR showed worst results when using SMOTE followed by the cleaning method, however, it manages to outperform RF when SMOTE was followed by Tomek Links (SMOTE+Tomek). Interestingly, the RF classifier again showed good results when data was over-sampled by (SMOTE+ENN) and it performs better more similar to SVM and LR. Compared with the linear autoencoder, the results obtained by the LR trained on the features from an autoencoder using non-linear sigmoid functions and over-sampled by Borderline-SMOTE2 was the

highest as compared to all classifiers prediction performance on the linear autoencoder as reported in Table 4.2. That is, the LR classifier achieved an overall AUC score of 0.750 outperforming all other classifiers across the various combinations of techniques.

With respect to dataset 2, the LR classifier with ROS showed the best performance (AUC=0.649) as compared to other classifiers. The ROS did not show significant performance for RF classifier (AUC=0.503) outperformed by the baseline model. On the other hand, SVM and LR performed well when coupled with ADASYN outperforming both data cleaning methods SMOTE+Tomek and SMOTE+ENN respectively. With respect to RF, the best results were achieved when the dataset was over-sampled by ADASYN and SMOTE+Tomek. The best prediction performance was obtained by LR with ROS (AUC=0.655) as reported in Table 4.2.

TABLE 4.2: The performance evaluation of machine learning classifiers, oversampling techniques and non-linear sigmoid autoencoder based on AUC for dataset 1 and 2.

Non-Linear sigmoid	Oversampling Technique	LR	SVM	RF
Dataset 1	ROS	0.740	0.749	0.658
	SMOTE	0.738	0.747	0.664
	ADASYN	0.742	0.741	0.681
	Borderline-SMOTE1	0.748	0.745	0.651
	Borderline-SMOTE2	**0.750**	0.746	0.651
	SMOTE + Tomek	0.735	0.748	0.672
	SMOTE + ENN	0.747	0.741	0.732
Dataset 2	ROS	**0.655**	0.628	0.503
	SMOTE	0.654	0.613	0.529
	ADASYN	0.651	0.610	0.535
	Borderline-SMOTE1	0.643	0.614	0.527
	Borderline-SMOTE2	0.645	0.605	0.522
	SMOTE + Tomek	0.645	0.604	0.524
	SMOTE + ENN	0.616	0.608	0.552

Figure 4.3 shows the aggregated AUC score differences between the baseline models and classifiers implemented on the dataset reduced by non-linear sigmoid and over-sampled using various techniques. It is clear from Figure 4.3 that RF trained with ROS, Borderline-SMOTE1, and Borderline-SMOTE2 did not perform better than the baseline model. In addition, it shows the best improvements when SMOTE followed by ENN was used to balance the data. On the other hand, both LR and SVM showed much improvement with ROS reported the highest score (more than 0.08 AUC score improvement) respectively.

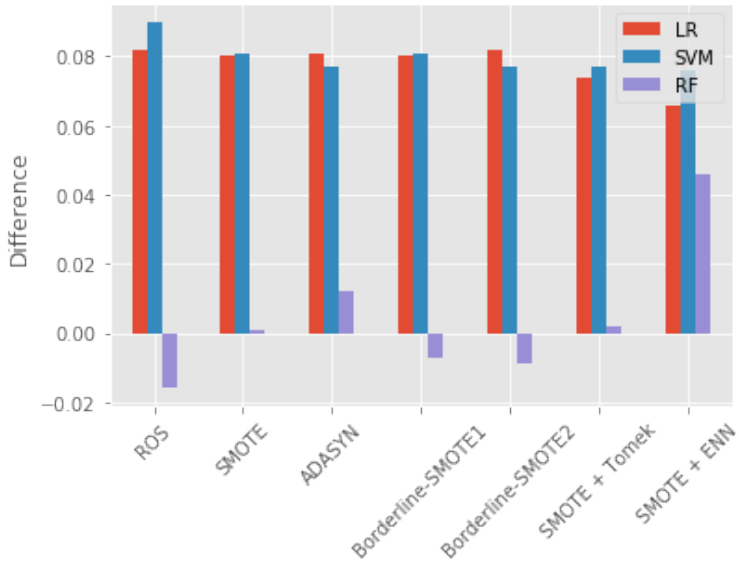


FIGURE 4.3: The difference in performance evaluation of various classifiers on datasets from dimensionality reduction using non-linear sigmoid autoencoder and oversampling methods against the baseline results.

4.3.3 Comparison of the Performance of Classifiers and Oversampling Methods Evaluated on Features Reduced by Autoencoder using ReLU Function

Another dimensionality reduction utilised was non-linear autoencoder using relu. From Table 4.3, it is clear that the prediction models continued improved results as compared to the baseline classifiers. In particular, LR performed better when dataset 1 was over-sampled with Borderline-SMOTE1, SMOTE, and SMOTE+Tomek.

The overall best score was obtained when LR is coupled with Borderline-SMOTE1 (AUC=0.753) compared to SVM and RF. Moreover, for SMOTE+ENN oversampling technique all the classifiers have obtained similar performance with LR with AUC=0.737 outperforming SVM with AUC=0.738 followed by RF with AUC=0.731. Interestingly, for dataset 2 SMOTE followed by Tomek's Links combined with LR achieved the highest prediction performance. Although the RF classifier showed best results when coupled with SMOTE+ENN for linear and non-linear sigmoid autoencoders, then with respect to non-linear relu both were outperformed by ADASYN (AUC=0.571).

TABLE 4.3: The performance evaluation of machine learning classifiers, oversampling techniques and non-linear relu autoencoder based on AUC for dataset 1 and 2.

Non-Linear ReLU	**Oversampling Technique**	**LR**	**SVM**	**RF**
Dataset 1	ROS	0.746	0.743	0.630
	SMOTE	0.748	0.748	0.665
	ADASYN	0.747	0.750	0.665
	Borderline-SMOTE1	**0.753**	0.750	0.655
	Borderline-SMOTE2	0.746	0.729	0.653
	SMOTE + Tomek	0.748	0.748	0.688
	SMOTE + ENN	0.737	0.738	0.731
Dataset 2	ROS	0.643	0.638	0.503
	SMOTE	0.643	0.598	0.528
	ADASYN	0.617	0.610	0.571
	Borderline-SMOTE1	0.644	0.617	0.515
	Borderline-SMOTE2	0.640	0.604	0.536
	SMOTE + Tomek	**0.645**	0.604	0.524
	SMOTE + ENN	0.616	0.608	0.549

Moreover, as illustrated in Figure 4.4 it can be seen that RF did not benefit from non-linear relu autoencoder with ROS, Borderline-SMOTE1 techniques. Compared to linear and sigmoid autoencoders, the RF obtained an improved AUC score of 0.02. For the SVM classifier, the best improvement was found when data was oversampled by ROS followed by Borderline-SMOTE1 and ADASYN. On the other

hand, the LR showed significant results improvement when data was balanced using Borderline-SMOTE1 and SMOTE+Tomek.

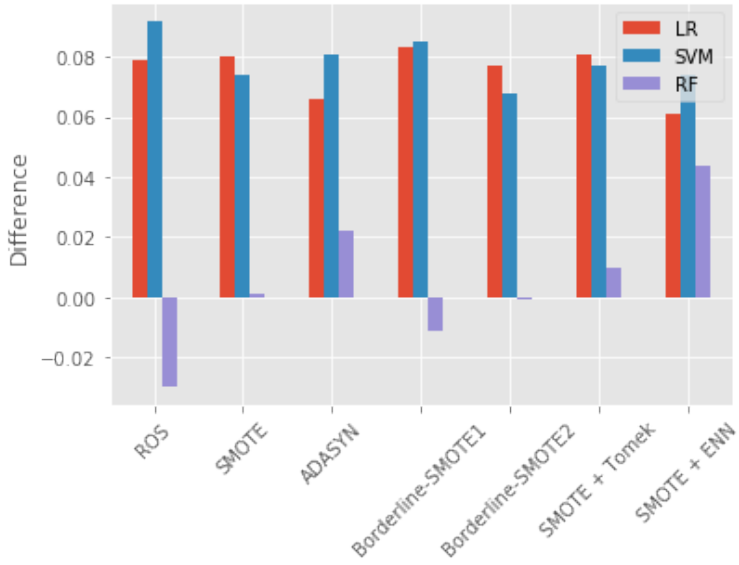


FIGURE 4.4: The difference in performance evaluation of various classifiers on datasets from dimensionality reduction using non-linear relu autoencoder and oversampling methods against the baseline results.

4.3.4 Comparison of the Performance of Classifiers and Oversampling Methods Evaluated on Features Reduced by Deep Autoencoder

Lastly, we have also studied deep autoencoder which consisted of 3 hidden layers for dataset 1. The results for this dimensionality reduction method decreased significantly for SVM across all the oversampling methods as compared to undercomplete autoencoders (i.e. linear, non-linear sigmoid, non-linear relu). As illustrated in Table 4.4, it is clear that the LR showed consistent and good results across all dimensionality reduction with all oversampling techniques. In addition, LR with SMOTE achieved the best results with an AUC score of 0.742 whereas Borderline-SMOTE2 got the lowest score (AUC=0.729). The performance of both SVM and RF, however, were significantly higher trained on the data over-sampled with SMOTE+ENN with both classifiers achieved an AUC score greater than 0.70.

On the other hand, LR gets better results for deep autoencoder when combined with SMOTE. Overall, RF outperformed LR and SVM when deep autoencoder dimensionality reduction was followed by SMOTE+ENN. Furthermore, the RF also obtained the highest score (AUC=0.752) followed by LR (AUC=0.740). This means for SMOTE+ENN the RF is preferred. Overall for dataset 1, the LR showed effective results for linear autoencoder with ROS, non-linear sigmoid autoencoder combined with Borderline-SMOTE2 and non-linear relu autoencoder combined with SMOTE+Tomek. On the other hand, for deep autoencoders the RF achieved the highest AUC score when combined with SMOTE+ENN. To evaluate the significance of autoencoder, we have used 5 hidden layers for dataset 2. The results are significantly lower for SVM as compared to the other three autoencoders dimensionality reduction techniques. Both LR and SVM obtained better results when ADASYN was applied as an oversampling technique. In addition, LR outperformed both SVM and RF across all oversampling techniques. Lastly, SMOTE followed by ENN proved to be the best oversampling technique for RF.

TABLE 4.4: The performance evaluation of machine learning classifiers, oversampling techniques and deep autoencoder based on AUC for dataset 1 and 2.

Deep AutoEncoder	**Oversampling Technique**	**LR**	**SVM**	**RF**
Dataset 1	ROS	0.740	0.672	0.634
	SMOTE	0.742	0.668	0.666
	ADASYN	0.730	0.663	0.672
	Borderline-SMOTE1	0.734	0.650	0.654
	Borderline-SMOTE2	0.729	0.660	0.645
	SMOTE + Tomek	0.740	0.667	0.688
	SMOTE + ENN	0.740	0.715	**0.752**
Dataset 2	ROS	0.639	0.606	0.518
	SMOTE	0.638	0.607	0.523
	ADASYN	**0.643**	0.614	0.519
	Borderline-SMOTE1	0.637	0.608	0.524
	Borderline-SMOTE2	0.633	0.598	0.534
	SMOTE + Tomek	0.639	0.606	0.518
	SMOTE + ENN	0.614	0.604	0.558

Compared to linear, non-linear sigmid, and relu autoencoders it can clearly be observed in Figure 4.5 that the average performance of both SVM and RF significantly decreased. Except for SVM+ENN, the results of SVM coupled with oversampling significantly decreased to scores of about 0.04 from much improvement of 0.08. In line with ROS oversampling, the results still not improved for RF similar to other autoencoders which shows performance decrease (0.02) compared to baseline results.

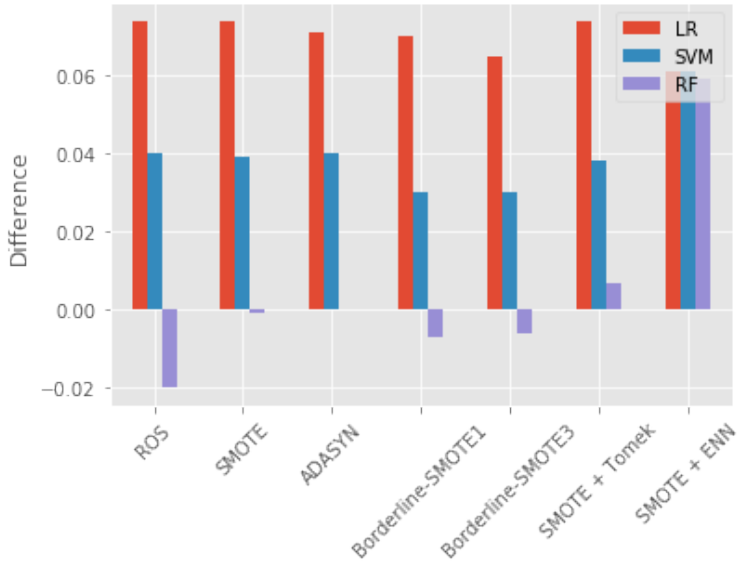


FIGURE 4.5: The difference in performance evaluation of various classifiers on datasets from dimensionality reduction using deep autoencoder and oversampling methods against the baseline results.

4.4 Overall Comparison Performance

Figure 4.6 depicts the aggregated overall results of all classification models. In comparison with aggregated AUC scores of the baseline results, 4.6 clearly shows that for LR and SVM the results are pretty across all oversampling and dimensionality reduction techniques. In line with RF, pre-processed results have not improved when RF was coupled by ROS mostly, and all the dimensionality reduction techniques. Overall, results were lower for RF and SVM when deep autoencoder was used whereas LR showed better consistent results.

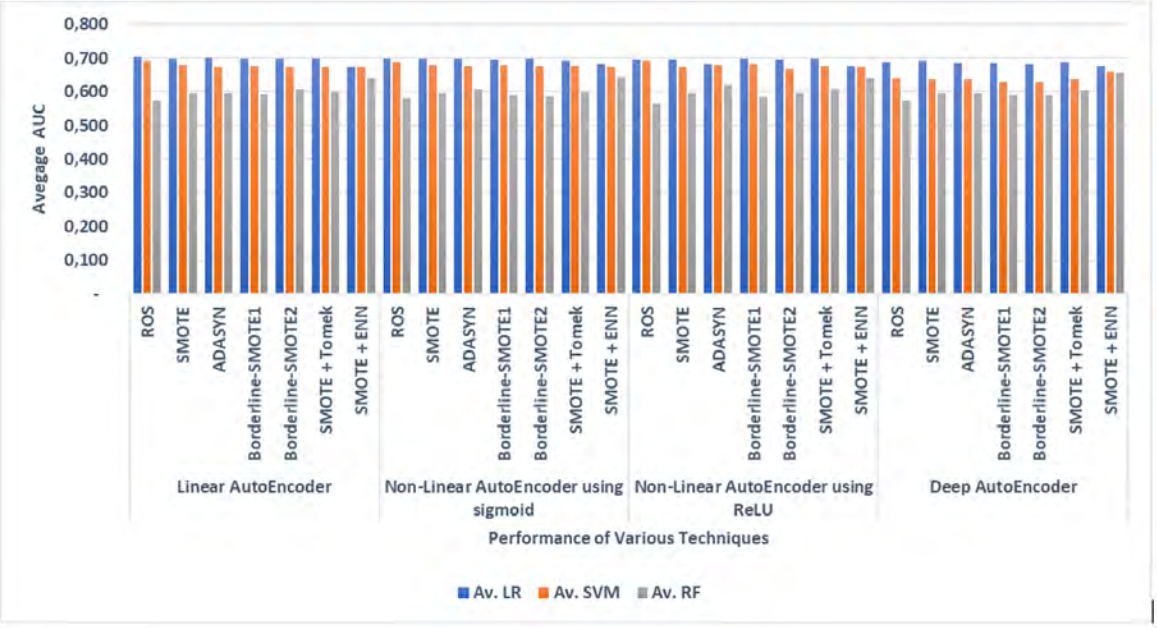


FIGURE 4.6: Experimental Performance of Oversampling, Dimensionality Reduction, and Machine Learning Techniques

4.5 Summary

In this chapter, we presented on how machine learning classification techniques combined with various oversampling methods and dimensionality reduction using autoencoder variations can be used to prevent and address the problem of customer churn. With the issue of class imbalance, we have demonstrated how oversampling techniques could improve machine learning models prediction performance. Moreover, 5-fold cross-validation was used as a remedy to build the classification models. Additionally, variations of autoencoders dimensionality reduction techniques showed a significant impact in improving model performances and hence reduce computational run-time. Thus, through our experimental setup, we were able to improve models performance as compared to the baseline results.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

In this study, we examined and compared the performance of three supervised machine learning classifiers i.e, Logistic Regression, Support Vector Machine, and Random Forest. Moreover, we also addressed the issue of class imbalance problem (CIP) and high dimensionality. To overcome the CIP, the performance of seven oversampling techniques namely, ROS, SMOTE, ADASYN, Borderline-SMOTE1, Borderline-SMOTE2, SMOTE + Tomek, and SMOTE + ENN were compared. On the other hand, dimensionality reduction using autoencoders with various functions; linear autoencoders, non-linear sigmoid, ReLU, and deep autoencoders were used. The experimental results show that the performance of machine learning classifiers can significantly vary. In addition, it was revealed that more complex techniques such as advance synthetic oversampling methods and deep autoencoder dimensionality reduction do not necessarily guarantee the best results. The findings from this study demonstrate that there is no much of difference in terms of models performance among these oversampling techniques. From the experimental analysis, it was found that the LR combined with a simple ROS and linear autoencoder yielded the overall best prediction performance. Moreover, LR which is a linear model have yielded a higher performance compared to the RF and non-linear SVM. Since LR is able to separate churn and non-churn classes, then this means that the dataset is linearly separable.

5.2 Future Work

There are some interesting areas that could be extended from this current study. It would be interesting for future work to address the significant impact of the outliers and high cardinality for categorical features. Although the oversampling techniques showed the potential to improve machine learning models performance, it will be important to investigate some of the under-sampling and cost-sensitive methods. In addition, deep learning techniques could also be implemented for the improvement of prediction performance.

Bibliography

[1] Kdd cup 2009. *Customer relationship prediction Data*. 2009. URL: https://www.kdd.org/kdd-cup/view/kdd-cup-2009/Data/ (visited on 06/12/2019).

[2] Abdelrahim Kasem Ahmad, Assef Jafar, and Kadan Aljoumaa. "Customer churn prediction in telecom using machine learning in big data platform". In: *Journal of Big Data* 6.1 (2019), p. 28.

[3] Adnan Amin et al. "Comparing oversampling techniques to handle the class imbalance problem: A customer churn prediction case study". In: *IEEE Access* 4 (2016), pp. 7940–7957.

[4] Muhammad Azeem, Muhammad Usman, and ACM Fong. "A churn prediction model for prepaid customers in telecom using fuzzy classifiers". In: *Telecommunication Systems* 66.4 (2017), pp. 603–614.

[5] Pierre Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012, pp. 37–49.

[6] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. "A study of the behavior of several methods for balancing machine learning training data". In: *ACM SIGKDD explorations newsletter* 6.1 (2004), pp. 20–29.

[7] Rezaul Begg and Joarder Kamruzzaman. "A machine learning approach for automated recognition of movement patterns using basic, kinetic and kinematic gait data". In: *Journal of biomechanics* 38.3 (2005), pp. 401–408.

[8] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, pp. 144–152.

[9] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[10] Christopher JC Burges. "A tutorial on support vector machines for pattern recognition". In: *Data mining and knowledge discovery* 2.2 (1998), pp. 121–167.

[11] Nitesh V Chawla et al. “SMOTE: synthetic minority over-sampling technique”. In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[12] François Chollet et al. *Keras*. https://keras.io. 2015.

[13] Nello Cristianini, John Shawe-Taylor, et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[14] Dursun Delen, Glenn Walker, and Amit Kadam. “Predicting breast cancer survivability: a comparison of three data mining methods”. In: *Artificial intelligence in medicine* 34.2 (2005), pp. 113–127.

[15] InĹes Domingues et al. “Evaluation of oversampling data balancing techniques in the context of ordinal classification”. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–8.

[16] Ganggang Dong et al. “A Review of the Autoencoder and Its Variants: A Comparative Perspective from Target Recognition in Synthetic-Aperture Radar Images”. In: *IEEE Geoscience and Remote Sensing Magazine* 6.3 (2018), pp. 44–68.

[17] Mohammed Abdul Haque Farquad, Vadlamani Ravi, and S Bapi Raju. “Churn prediction using comprehensible support vector machine: An analytical CRM application”. In: *Applied Soft Computing* 19 (2014), pp. 31–40.

[18] Mikel Galar et al. “A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4 (2011), pp. 463–484.

[19] Vicente García, Ana Isabel Marqués, and Jose Salvador Sánchez. “Improving risk predictions by preprocessing imbalanced credit data”. In: *International conference on neural information processing*. Springer. 2012, pp. 68–75.

[20] Esperanza García-Gonzalo et al. “Hard-rock stability analysis for span design in entry-type excavations with learning classifiers”. In: *Materials* 9.7 (2016), p. 531.

[21] Vijay N Garla and Cynthia Brandt. “Ontology-guided feature engineering for clinical text classification”. In: *Journal of biomedical informatics* 45.5 (2012), pp. 992–998.

[22] Niccolò Gordini and Valerio Veglio. "Customers churn prediction and marketing retention strategies. An application of support vector machines based on the AUC parameter-selection technique in B2B e-commerce industry". In: *Industrial Marketing Management* 62 (2017), pp. 100–107.

[23] Chun Gui. "Analysis of imbalanced data set problem: The case of churn prediction for telecommunication." In: *Artif. Intell. Research* 6.2 (2017), p. 93.

[24] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. "Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning". In: *International conference on intelligent computing*. Springer. 2005, pp. 878–887.

[25] Eka Pura Hartati, Moch Arif Bijaksana, et al. "Handling imbalance data in churn prediction using combined SMOTE and RUS with bagging method". In: *Journal of Physics: Conference Series*. Vol. 971. 1. IOP Publishing. 2018, p. 012007.

[26] Mohammed Hassouna et al. "Customer churn in mobile markets a comparison of techniques". In: *arXiv preprint arXiv:1607.07792* (2016).

[27] Haibo He et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 1322–1328.

[28] Yiqing Huang et al. "Telco churn prediction with big data". In: *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. ACM. 2015, pp. 607–618.

[29] John D Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.3 (2007), p. 90.

[30] IBM. *Guide to Sample Data Sets Watson Analytics Community*. 2016. URL: https://www.ibm.com/communities/analytics/watson-analytics-blog/predictive-insights-in-the-telco-customer-churn-data-set/ (visited on 12/02/2019).

[31] Lei Jin et al. "Training large scale deep neural networks on the intel xeon phi many-core coprocessor". In: *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*. IEEE. 2014, pp. 1622–1630.

[32] Mirjana Karanovic et al. "Telecommunication Services Churn Prediction-Deep Learning Approach". In: *2018 26th Telecommunications Forum (TELFOR)*. IEEE. 2018, pp. 420–425.

[33] Clement Kirui, Li Hong, and Edgar Kirui. "Handling Class Imbalance in Mobile Telecoms Customer Churn Prediction". In: *International Journal of Computer Applications* 72.23 (2013), pp. 7–13.

[34] German Lahera. *Unbalanced Datasets What To Do About Them Medium*. 2018. URL: https://medium.com/strands-tech-corner/unbalanced-datasets-what-to-do-144e0552d9cd (visited on 07/10/2019).

[35] Quoc V Le et al. "A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks". In: *Google Brain* (2015), pp. 1–20.

[36] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 559–563.

[37] JP Lewis. "Tutorial on SVM". In: *CGIT Lab, USC* (2004).

[38] Wei-Chao Lin, Chih-Fong Tsai, and Shih-Wen Ke. "Dimensionality and data reduction in telecom churn prediction". In: *Kybernetes* 43.5 (2014), pp. 737–749.

[39] Huan Liu et al. "Evolving feature selection". In: *IEEE Intelligent systems* 20.6 (2005), pp. 64–76.

[40] Ning Lu et al. "A customer churn prediction model in telecom industry using boosting". In: *IEEE Transactions on Industrial Informatics* 10.2 (2014), pp. 1659–1665.

[41] marcoaltini. *DEALING WITH IMBALANCED DATA: UNDERSAMPLING, OVERSAMPLING AND PROPER CROSS-VALIDATION marcoaltini*. 2015. URL: https://www.marcoaltini.com/blog/dealing-with-imbalanced-data-undersampling-oversampling-and-proper-cross-validation (visited on 07/11/2019).

[42] Wes McKinney et al. "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56.

[43] Fatemeh Nargesian et al. "Learning Feature Engineering for Classification." In: *IJCAI*. 2017, pp. 2529–2535.

[44] Nathan-Hubens. *Deep inside: Autoencoders Towards Data Science*. 2018. URL: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f (visited on 05/06/2019).

[45] Andrew Ng and Sparse Autoencoder. "CS294A Lecture notes". In: *Dosegljivo: https://web. stanford. edu/class/cs294a/sparseAutoencoder_2011new. pdf.[Dostopano 20. 7. 2016]* (2011).

[46] SV Patel and Veena N Jokhakar. "A random forest based machine learning approach for mild steel defect diagnosis". In: *Computational Intelligence and Computing Research (ICCIC), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1–8.

[47] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[48] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12.Oct (2011), pp. 2825–2830.

[49] U Devi Prasad and S Madhavi. "Prediction of churn behavior of bank customers using data mining tools". In: *Business Intelligence Journal* 5.1 (2012), pp. 96–101.

[50] Saad Ahmed Qureshi et al. "Telecommunication subscribers' churn prediction model using machine learning". In: *Eighth International Conference on Digital Information Management (ICDIM 2013)*. IEEE. 2013, pp. 131–136.

[51] Yossi Richter, Elad Yom-Tov, and Noam Slonim. "Predicting customer churn in mobile networks through analysis of social groups". In: *Proceedings of the 2010 SIAM international conference on data mining*. SIAM. 2010, pp. 732–741.

[52] Ali Rodan et al. "A support vector machine approach for churn prediction in telecom industry". In: *International journal on information* 17.8 (2014), pp. 3961–3970.

[53] Julian Runge et al. "Churn prediction for high-value players in casual social games". In: *2014 IEEE conference on Computational Intelligence and Games*. IEEE. 2014, pp. 1–8.

[54] Miriam Seoane Santos et al. "Cross-validation for imbalanced datasets: Avoiding overoptimistic and overfitting approaches [research frontier]". In: *ieee ComputatioNal iNtelligeNCe magaziNe* 13.4 (2018), pp. 59–76.

[55] Ashish Shah. "Prediction of Malignant and Benign Tumor using Machine Learning". In: *International Journal of Computer Applications* 135.5 (2016), pp. 19–23.

[56] Hyun Joon Shin, Dong-Hwan Eom, and Sung-Shick Kim. "One-class support vector machines—an application in machine fault detection and classification". In: *Computers & Industrial Engineering* 48.2 (2005), pp. 395–408.

[57] Jie Sun et al. "Imbalanced enterprise credit evaluation with DTE-SBD: decision tree ensemble based on SMOTE and bagging with differentiated sampling rates". In: *Information Sciences* 425 (2018), pp. 76–91.

[58] Kai Ming Ting. "An instance-weighting method to induce cost-sensitive trees". In: *IEEE Transactions on Knowledge and Data Engineering* 14.3 (2002), pp. 659–665.

[59] V Umayaparvathi and K Iyakutti. "Automated feature selection and churn prediction using deep learning models". In: *International Research Journal of Engineering and Technology (IRJET)* 4.3 (2017), pp. 1846–1854.

[60] Thanasis Vafeiadis et al. "A comparison of machine learning techniques for customer churn prediction". In: *Simulation Modelling Practice and Theory* 55 (2015), pp. 1–9.

[61] Wouter Verbeke et al. "New insights into churn prediction in the telecommunication sector: A profit driven data mining approach". In: *European Journal of Operational Research* 218.1 (2012), pp. 211–229.

[62] Shuo Wang and Xin Yao. "Multiclass imbalance problems: Analysis and potential solutions". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42.4 (2012), pp. 1119–1130.

[63] Carolina YV Watanabe et al. "SACMiner: a new classification method based on statistical association rules to mine medical images". In: *International Conference on Enterprise Information Systems*. Springer. 2010, pp. 249–263.

[64] Dennis L Wilson. "Asymptotic properties of nearest neighbor rules using edited data". In: *IEEE Transactions on Systems, Man, and Cybernetics* 3 (1972), pp. 408–421.

[65] Jianning Wu, Jue Wang, and Li Liu. “Feature extraction via KPCA for classification of gait patterns”. In: *Human movement science* 26.3 (2007), pp. 393–411.

[66] Guo-en Xia and Wei-dong Jin. “Model of customer churn prediction on support vector machine”. In: *Systems Engineering-Theory & Practice* 28.1 (2008), pp. 71–77.

[67] Min Zeng et al. “Effective prediction of three common diseases by combining SMOTE with Tomek links technique for imbalanced medical data”. In: *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*. IEEE. 2016, pp. 225–228.

[68] Bing Zhu, Bart Baesens, and Seppe KLM vanden Broucke. “An empirical comparison of techniques for the class imbalance problem in churn prediction”. In: *Information sciences* 408 (2017), pp. 84–99.

[69] Bing Zhu et al. “Improving Resampling-based Ensemble in Churn Prediction”. In: *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*. 2017, pp. 79–91.