Can Generative Adversarial Networks Help Us Fight Financial Fraud?

Sean McIver

Technological University Dublin

Can Generative Adversarial Networks Help Us Fight Financial Fraud?

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Sean McIver

A dissertation submitted in partial fulfilment of the requirements of
Technology University Dublin for the degree of
M.Sc. in Computer Science (Data Science)

Date: 2021-06-15

Declaration

I certify that this dissertation which I now submit for examination for the award of M.Sc. in Computer Science (Data Science), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Dublin Institute of Technology and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institutes's guidelines for ethics in research.

Signed: [illegible]

Date: 2021-06-15

Abstract

Transactional fraud datasets exhibit extreme class imbalance. Learners cannot make accurate generalizations without sufficient data. Researchers can account for imbalance at the data level, algorithmic level or both. This paper focuses on techniques at the data level. We evaluate the evidence of the optimal technique and potential enhancements. Global fraud losses totalled more than 80 % of the UK's GDP in 2019. The improvement of preprocessing is inherently valuable in fighting these losses. Synthetic minority oversampling technique (SMOTE) and extensions of SMOTE are currently the most common preprocessing strategies. SMOTE oversamples the minority classes by randomly generating a point between a minority instance and its nearest neighbour. Recent papers adopt generative adversarial networks (GAN) for data synthetic creation. Since 2014 there had been several GAN extensions, from improved training mechanisms to frameworks specifically for tabular data. The primary aim of the research is to understand the benefits of GANs built specifically for tabular data on supervised classifiers performance. We determine if this framework will outperform traditional methods and more common GAN frameworks. Secondly, we propose a framework that allows individuals to test the impact of imbalance ratios on classifier performance. Finally, we investigate the use of clustering and determine if this information can help GANs create better synthetic information. We explore this in the context of commonly used supervised classifiers and ensemble methods.

Keywords: Fraud detection, generative adversarial networks, SMOTE, class-imbalance, supervised learning, clustering

Acknowledgements

Special thanks to Dr Giancarlo Salton, my thesis supervisor for all his contributions and help along the way. Muito Obrigado.

Thanks to all the staff at TUD who helped me gain the skills I needed to complete this dissertation. Thank you to my family who have helped support me in the past 2 years. Thanks to Hayley for doing an excellent job proof reading. Finally, thanks to Cartrawler and the Customer Transactions Team who have been flexible and supportive during my studies.

Contents

List of Figures

List of Tables

List of Acronyms

GAN	Generative Adversarial Network
WGAN	Wasserstein Generative Adversarial Network
cGAN	Conditional Generative Adversarial Network
SMOTE	Synthetic Minority Upsampling
ADASYN	Adaptive Synthetic Sampling
OOB	Out-of-Bag
TSS	Total Within Sum of Squares

Chapter 1

Introduction

Financial fraud most commonly occurs when bad actors obtain stolen credit card information and use it for their gain. Global fraud losses totalled more than 80 per cent of the UK's GDP in 2019 (Gee & Button, 2019), highlighting the inherent need for financial institutions to invest in fraud detection solutions. Fraud detection using supervised statistical learners is hence a common and important area of machine learning research. The problem itself can be simplified into a simple binary classification problem, where statistical learners use certain features to learn distributions of fraudulent transactions. A common attribute faced by fraudulent dataset is the presence of extreme class imbalance. Having little volume of fraudulent data hinders a learners ability to correctly identify new fraud instances. To remedy this we explore the use of synthetic data creation techniques.

1.1 Background

Transactional fraud classification is a complex problem. Fraud instances tend to be non-static, resulting in distributional change over time. Secondly, fraud is scarce, creating difficulty for statistical learners to generalise the desired distribution. The research problem we are considering focuses on the latter. Given the rarity of fraud instances, classifiers will be biased toward any majority class. For example, if 99.9% of a dataset is not fraudulent then the classifier will be considerably better at predicting this class. Misclassification is hence more common if this problem is not addressed and by definition financial fraud has a high cost associated with it. Data-level techniques that change the imbalance ratio to reduce the classifiers bias toward the majority class. Upsampling at an algorithmic level using nearest neighbour (Chawla et al., 2002; Han et al., 2005)) and cluster approaches (He et al., 2008) have been widely researched. (Haixiang et al., 2017). Haixiang et al. (2017) showed out of a sample of 156 papers 29.6% used oversampling techniques.

Newer research has employed generative adversarial networks (GAN) (Goodfellow et al., 2014) for synthetic data generation. GANs originated and are commonly used in image processing literature to generate realistic images based on large training sets of images. GANs are less commonly used for the generation of tabular data, however, more recent papers have found GANs can be used to generate tabular structured data and have outperformed traditional oversampling methods in aiding supervised classifiers performance for extremely imbalanced datasets (Douzas & Bacao, 2018; Gangwar & Ravi, 2019). Research to date has adopted the same GAN framework that is used for image generation tasks. Xu et al. (2019) proposes a GAN specifically designed for tabular data. This framework has yet to be applied to aid fraud classifiers. Our research paper will focus on this gap. The introduction of conditional GANs (Mirza & Osindero, 2014) allowed for more realistic synthetic image generation. This occurs when the data generation is conditioned on a particular feature e.g. annotation and tagging. Credit card fraud occurs in fraud *rings*. The application of clustering can be applied to identify meaningful groups of common account holders. These clusters can have varying fraud risks attached with them (Kasa et al., 2019). We further explore if the use of clustering can help GANs create better synthetic data and hence better classification performance.

1.2 Research Problem

Our research will focus on GANs to understand if this method of synthetic data generation is superior to commonly used traditional methods in aiding supervised classifier's performance. The main contribution of the project is to consider a GAN framework that is built specifically for tabular data (Conditional Tabular GAN) (Xu et al., 2019). We define our research problem as:

"To what extent can generative adversarial networks improve the performance of supervised classifiers on fraud detection for financial transactions?"

Secondly, we conduct multiple quantitative experiments across different imbalance ratios. The imbalance ratio is defined as the ratio between the minority class and the majority class. This provides a sub research question:

Research Sub-Question A: *"Does an optimal imbalance ratio threshold exist when adding synthetic data to maximise the performance of a classifier for the detection of financial transactions?"*

Finally, we consider the use of k-means clustering assignment on our fraud data to understand if this can improve the GAN training process. This provides an additional sub research question:

Research Sub-Question B: *"To what extent can the use of additional cluster information reduce the accuracy of a classifier synthetic and non-synthetic data generated by GANs?"*

1.3 Research Objectives

There are several objectives of the research project. Firstly, we perform a literature review of data level oversampling techniques, GANs and GAN developments over time. The aim of this review determines any gaps or limitations that exist and research newer GAN techniques that have yet to be applied in this space. Secondly, we use empirical experiments to determine if GANs designed for modelling tabular data will improve supervised classifier performance. Through our experimental framework, we gather evidence to compare GAN methods against traditional oversampling methods. Our experimental framework involves 65 comparisons across differing imbalance ratios. This provides us with evidence towards the existence of an optimal imbalance ratio. Additionally, this should give us insight into the relationship between classifier performance and a datasets imbalance ratio. Finally, we explore the use of k-means clustering, to determine if the addition of this information can aid us in synthetic data generation for CTGAN.

1. Literature review of traditional oversampling techniques and GANs
2. Explanatory data analysis and cleansing
   (a) Feature Engineering - Identify features within the dataset that can be manipulated and transformed to create new features. New features will be used in the aims of improving classifier performance and GAN training.
   (b) Skewness and distribution analysis - Define asymmetry of distributions for each class. Use standard error to standardise skewness and compare against bounds of a normal univariate distribution. Visualise distributions using ggplot2 (Wickham, 2016).
   (c) Duplicates and common values - Use *'dplyr'* (Wickham et al., 2018) to identify any rows that are exact duplicates. Group fraud transaction by amount to determine any common amounts.
   (d) k-Means cluster analysis
      i. Elbow Plot - Initialise k in range 0-10. For each k, calculate the total within sum of squares (TSS) as a performance metric achieved using function *fviz nbclust* by Kassambara and Mundt (2017). Investigate values k which represent large reduction in TSS.
      ii. Cluster Visualisation - Using *fviz cluster* function (Kassambara & Mundt, 2017) compute principal component analysis (PCA) to determine principal components of chosen k and visually inspect cluster formations.
      iii. Concatenate fraud results and their cluster assignments for use in GAN training.
3. Hyper-parameter tuning for tree models (Decision Tree, Random Forest, XG-Boost)

(a) Initialise 5-fold cross validation framework using *scikit-learn* (Pedregosa et al., 2011). Identify desired parameters. Note both optimal average performance and computational requirements to gather evidence for parameters to use in our experimental framework.

(b) GAN training framework

   i. Define up parameters for WGAN with gradient penalty and CTGAN.

   ii. Define GAN evaluation framework. Define an XGBoost classifier to determine accuracy.

   iii. Train CTGAN 10 times per 100 epochs from 100-500. Store accuracy result for each iteration. Repeat using clustered information. Visualise results in box-plot using *matplotlib*(Hunter, 2007).

   iv. Compare significance across groups - Perform Sharpio-Wilk test and F-test to assess t-test assumptions. If assumptions hold, perform a t-test to assess evidence of inclusion of cluster information.

   v. Repeat steps for WGAN to determine epoch level to use in our experimental framework

   vi. Replicate framework used by Nash (2017) to benchmark GAN results.

(c) Baseline result calculations - Use defined parameters from hyper-parameter tuning results. Use 5-fold cross validation to identify baseline F1 score using original imbalance ratio.

(d) Experimental framework engineering

   i. Engineer and code a framework for use with each oversampling methodology. This framework should allow the user to repeat 5-fold cross validation across a list of imbalance ratios.

   ii. Visualise the average F1 scores achieved from the framework across oversampling methods using matplotlib (Hunter, 2007)

(e) Test significance of results - Store and rank results. Employ a Friedman test to determine if results are significantly different across groups. Report on the mean rank to give evidence for the research hypothesis.

(f) Synthetic Data Distributional Analysis - Using best ranked GAN and traditional method, investigate the difference in distributions. Identify the 2 most important features from decision tree training. Tag synthetic and non-synthetic samples. Use ggplot2 to produce a scatter plot of the two features as and histograms faceted by synthetic identifier.

1.4 Research Methodologies

The research employs a quantitative methodology. The experiments undertook are based on the use of statistical models for learning and classification. Traditional models include probabilistic models (Naive Bayes), linear models (Logistic Regression) and tree-based models (Decision Trees). We employ bagging by using Random Forest and boosting using extreme gradient boosting (XGBoost). k-Means clustering is used as an additional method to provide potentially useful information to GANs that uses a conditional framework. A classifier optimisation strategy is defined using grid search architecture for decision tree, random forest XGBoost models. For traditional minority oversampling we consider SMOTE and ADASYN. We consider WGAN and CTGAN as our generative network architectures. Empirical investigations employ multiple 5-fold cross validations across combinations of classifiers and upsampling algorithms. Further, a Friedman test determines significance in ranking methods across classifiers and imbalance ratios.

1.5 Scope and Limitations

1.5.1 Scope

The scope of the research relates to generative adversarial networks as a data level oversampling strategy to improve the detection of fraudulent financial transactions using machine learning.

1.5.2 Limitations

1. Financial information is highly sensitive, therefore the obtained dataset comprises of majority principal component analysis (PCA) transformed columns. These limits are the ability to properly interpret the dataset in the context of influencing variables. A suggestion for further study would be to redesign the experiment using domain-specific columns.

2. The dataset spans only 2 days. Fraud is an ever-evolving problem, therefore we do not have sufficient evidence to understand if the results of our experiments would hold for newer observations.

3. Our research is focused primarily on financial fraud with extreme imbalance. We cannot discern from our experiment if the results are transferable to other non-transactional datasets.

4. We focus on methods to learn the distribution of the fraudulent dataset. Given fraud distributions are subject to distributional changes, it would be advisable to also investigate methods that are conditional on the majority class and look for distributional differences i.e. learn what is genuine and score transactions that deviate from this.

1.6 Document Outline

The research document is structured into the following chapters

- Chapter 2: "Review of existing literature" - This chapter aims to give a comprehensive view of supervised classifiers used for fraud, GANs and empirical evidence of relevant papers. From this review we identify clear gaps which mould the focus of our research project.

- Chapter 3: "Design and methodology" - This chapter defines our main research hypothesis. We detail the experimental framework used to gather evidence for this hypothesis. Details of the dataset, exploration and preprocessing techniques used are further detailed. GAN training evaluation and k-means clustering set up is also considered.

- Chapter 4: “Results, evaluation and discussion” - This chapter details the results from our experimental research and any experiment undertaken which provide evidence toward answering research questions and sub-questions. We discuss our findings in detail.

- Chapter 5: “Conclusion” - The concluding chapter aims to link our empirical results against our research objectives detailed in section 1.3. We critique our results and offer recommendations for future work for any gaps not considered in our experimental framework.

Chapter 2

Review of existing literature

2.1 Traditional Upsampling Methodologies

For class imbalance, the literature suggests 3 main techniques to tackle this problem:

1. Algorithmic level - When applying an algorithm, create a cost-sensitive function that commits higher cost to the minority class and boost class importance (Zhou & Liu, 2010).
2. Data level - Solve for the imbalance through:
    - Upsampling the minority class.
    - Undersampling the majority class - Remove majority class data to settle the imbalance. A common algorithm used is Random-Under-Sampling (Tahir et al., 2009).
3. Hybrid Model - an ensemble method combining technique 1 and 2.

Our research will focus on data level techniques. Haixiang et al. (2017) review of 159 papers tacking fraud found 29.6% employed these techniques.

2.1.1 Synthetic Minority Oversampling (SMOTE)

SMOTE (Chawla et al., 2002) is a distance-based algorithm that generates new samples based on a random distance between points. Using the minority class, the technique introduces new observations within the line segments joining k nearest neighbours. k is randomly chosen depending on the level of oversampling needed. The distance between the sample and its nearest neighbour is multiplied by a random number between 0 and 1. This number is added to the feature space. There are over 85 SMOTE extensions since the original paper (Fernandez et al., 2018) which vary in techniques. An example is employing kernel functions to replace the nearest neighbour with a clustering framework. Our experiments will only consider *vanilla* SMOTE.

2.1.2 Adaptive Synthetic Upsampling Technique (ADASYN)

ADASYN (He et al., 2008) builds upon SMOTE logic with the addition of using weighted distribution across minority examples. The intuition behind this is to shift the classifiers decision boundary to focus more on examples that are difficult-to-learn. This results in more synthetic data around these difficult to learn areas compared to observations with more well-defined distributions. This further aims to reduce the bias that occurs due to class imbalance.

2.1.3 Problems with SMOTE algorithms

SMOTE algorithms are affected by the location of the minority class. SMOTE shows problems when classes overlap or there are disjuncts within the data. A disjunct relates to areas within a larger cluster where classes overlap (Prati et al., 2004). This means the algorithm may create more data in an area that is not easily separable and requires more complex classifiers. Cluster-based SMOTE extensions e.g. ADASYN accounts for this, however, it is constrained by assumptions. Fernandez et al. (2018) argues that these assumptions may not be applicable for generating these complex distributions.

2.2 Generate Adversarial Networks

Generative adversarial networks (GAN) (Goodfellow et al., 2014) are neural networks used for synthetic data creation first introduced within image processing literature. The process is dependent on two models:

1. A generative model: $G : Z \rightarrow X$

   Z represents a space of noise with random dimension dZ. This is dependent on given hyper-parameters and X represents the data space.

2. The discriminative model: $D : X \rightarrow [0, 1]$

   D considers data from the real dataset.D will assign a probability that the sample is genuine.

This creates a min-max game. The discriminator aims to maximise the average log probability of the real data and the inverse log for the synthetic data (Douzas & Bacao, 2018). G seeks to minimise the log inverse probability predicted by the generator to encourage the creation of data that is difficult to discern as synthetic. The value function is denoted as:

$$(min)_G \, (max)_D \, V \, (\, D, G \,) = E_D + E_G$$

where:

$$E_D = E_{x,y \sim p_{data}(x,y)} \, [\, logD \, (\, x, y \,) \,]$$

$$E_G = E_{z \sim p_z(\, z \,), y \sim p(\, y \,)} \, [\, log \, (\, 1 - D \, (\, G \, (\, z, y \,) \, , y \,) \,) \,]$$

x values are sampled from the real data while z values are sampled from the noise distribution. The aim is to optimise this process toward a probability of 0.5 i.e. the discriminator can't distinguish between actual and generated samples. The original paper from Goodfellow et al. (2014) uses Nash equilibrium as a point of optimisation. The training process of a min-max game will always be unstable. Vanilla GANs have a problem with convergence as the point at which to stop training can not be known.

Further, this attribute of vanilla GAN can create a mode collapse problem leading to a vanishing gradient (Goodfellow et al., 2014). There have been many additions to the literature to create a more stable GAN training process (Salimans et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017; Xu et al., 2019).

2.2.1 GAN Developments

Conditional Adversarial GANs

Conditional GAN (cGAN)(Mirza & Osindero, 2014) improves upon the vanilla GAN architecture by adding some conditional input c to both our generator (G) and discriminator (D) models. Given the conditional input's information is significant, the expected result is to create better structuring of latent space. This has been shown to have more favourable results within image processing. An example of helpful conditions is image tagging and annotations.

Applying this to our problem, we can identify clustering as a potential condition to aid in data generation. This unsupervised method will aim to identify similar types of fraudulent activities or behaviours of fraud rings e.g. same fraud actors making multiple fraud attempts. We will focus our experiments on using k-means to initialise potential fraud clusters.

Wasserstein GAN

Wasserstain GAN (Arjovsky et al., 2017) attempts to tackle the issue of bad gradients by creating a more stable optimisation process. This is achieved by replacing Jensen-Shannon (JS) divergence with a Wasserstain distance when comparing the synthesised samples against the generated samples (Liu et al., 2019). Wasserstein distance $W(q,p)$ can be described as the minimum cost of transporting mass to transform distribution q into distribution p. The value function of a WGAN uses Kantorovich-Rubinstein duality and is defined in equation 2.1.

$$L = \underset{G}{min}\underset{D\in D}{max} E_{x\sim P_r}\left[D\left(x\right)\right] - E_{\tilde{x}\sim P_g}\left[D\left(\tilde{x}\right)\right] \tag{2.1}$$

Here P_r relates to the distribution of the real data and P_g is the distribution of the synthetic data. This is generated using $\widetilde{x} = G(z)$. z represents a random noise vector initialised at the start of training. For training, the generator remains constant and the discriminator is trained by maximising the value function (2.1). After the maximisation is complete, the discriminant model stays constant to minimise the value function 2.1. This creates the Wasserstein distance between the two distributions (Gao et al., 2020). The aim here is to have the generated and real data as similar as possible. The min-max game will converge once the discriminator can no longer distinguish between real and synthetic data. Since the Wasserstein value function is continuous, the lower the Wasserstein distance the higher quality the synthetic data.

The discriminant model is comprised of a set of 1-Lipschitz functions. Equation 2.1 D is represented by K-Lipschitz functions. This is possible by clipping weight in each of the discriminator's layers. This stabilisation improvement has been shown empirically within the image processing domain (Zhu et al., 2019) and has shown as a successful oversampling technique (Wang et al., 2019).

WGAN with Gradient Penalty

An unwanted by-product of the WGAN framework is that in some instances the network can still fail to converge resulting in poor quality synthetic data. These instances are often caused by the use of weight clipping (Gao et al., 2020). Weight clipping is used to satisfy the Lipschitz constraint which defines WGAN's discriminant model. This clipping can restrict the weights of every layer to a restrictive range which may result in either vanishing or exploding gradients (Gao et al., 2020). Gulrajani et al. (2017) proposed adding a penalty factor to satisfy the Lipschitz constraint instead of WGAN's original clipping mechanism. This creates a new value function 2.2.

$$E_{x\sim P_r}\left[D\left(x\right)\right] - E_{\widetilde{x}\sim P_g}\left[D\left(\widetilde{x}\right)\right] - \lambda \underset{\hat{x}\sim P_{\hat{x}}}{E}\left[\,\left(\left(\left(\|\left(\nabla\right)_{\hat{x}} D\left(\,\hat{x}\,\right)\|\right)_2 - 1\,\right)\right)^2\,\right] \tag{2.2}$$

From equation 2.2, λ represents the penalty coefficient. This works by penalising any gradient norms that are far from 1, ensuring all gradient norms move toward 1. This is a property of a 1-Lipschitz function i.e., one whose gradient norm is a maximum of 1. This allows for optimised performance as this property allows for faster convergence. We will focus our non-tabular GAN training using this GAN formulation and address going forward as *WGAN*.

Conditional Tabular GANs (CTGAN)

CTGAN (Xu et al., 2019) is a GAN specifically targeted at generating data in a tabular format. Their extension accounts for features within the dataset that have more complicated distributions. The framework models continuous and discrete columns separately. The authors create a mode-specific normalisation to account for non-Gaussian and multi-modal distributions.

For every continuous column C_i that exists, a variational Gaussian mixture model (Reynolds, 2009) is used to detect multiple modes M_i. For each value $c_{i,j}$ in C_i a probability coming from each mode is computed. One mode is sampled from the given probability density, this is then used to normalise the value. The representation of a row in our dataset is detailed as the concatenation of continuous and discrete columns.

Vanilla GANs do not account for the imbalance of categorical columns. Xu et al. (2019) argues that if any rows fall into a minor category they will not be sufficiently represented during training. This is due to data being randomly sampled. They approach this problem by resampling in a way that categories from discrete attributes are sampled evenly while also recovering the real data distribution during testing. The generator in CTGAN is described as a conditional distribution of a particular column and row.

The output of the conditional generator is assessed by a critic network similar to WGAN. This calculates the distance between the learned conditional distribution and the conditional distribution of the real data (Xu et al., 2019). In this framework, the authors empirically show they learn distributions better compared to Bayesian networks. As CTGAN framework has never been tested as an oversampling method for fraudulent transactions, we will focus our research on this gap.

GANs as an Upsampling Strategy - Empirical Evidence

Dal Pozzolo et al. (2014) dataset containing fraudulent credit card attempts is often used within this domain (Sisodia et al., 2017; Tanaka & Aranha, 2019; Ba, 2019; Fiore et al., 2019). The dataset represents 284,807 financial transactions captured by a financial institution over two days in 2013. 492 of these are fraudulent (0.172%).

- Tanaka and Aranha (2019) uses a decision tree as a classifier comparing GAN, simple SMOTE and ADASYN. The best performing GAN model showed a recall of 0.82. The use of ADASYN showed better performance for the classifier (0.86).
- Douzas and Bacao (2018) compares SMOTE methods against GAN methods on 71 datasets with varying imbalance ratios using 3 evaluation metrics (AUC, F-score and G-Mean) across 5 classifiers. They find cGAN to significantly rank the highest on average in terms of performance.
- Gangwar and Ravi (2019) found upsampling using WGAN led to a significantly higher F1 score for the classifier compared to SMOTE and ADASYN.

2.3 Clustering Techniques

The objective of clustering is to identify similarities within a feature space and label them. For datasets dealing with financial fraud, this information can be useful in identifying fraud rings, or a single fraud actor attempting multiple times.

2.3.1 k-Means Clustering

The k-means algorithm splits our observations into a predetermined number of clusters (k). Initially, each observation is assigned to the nearest centroid. If an observation has the same distance between two centroids, one will be chosen at random. Given this initial centroid assignment the algorithm iterates using 2 steps (Friedman et al., 2001):

1. At each centroid we determine a subset of training points that is closer to it than any other centroid.

2. Calculate the mean of each feature for the points in each cluster. This mean vector becomes the new centre for that cluster.

This creates an optimisation problem 2.3, where $C(\dot{)}$ relates to the cluster assignment function and m_k represent cluster means .

$$\min_{C,\{m_k\}_{k=1}^{K}} \sum_{k=1}^{K} \sum_{C(i)=k} \|x_i - m_k\|^2 \tag{2.3}$$

For our experiment we will employ Euclidean distance $\|x-y\|=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$ which influences the shape of clusters.

To determine the optimal value of k we employ the use of the *elbow* method originally proposed by Thorndike (1953). At each k we compute the within sum of squares distance between each observation and its assigned centroid. k is plotted against the within sum of squares to visually determine a point of inflection where the reduction in the sum of squares is considerably less than the previous k. This creates an elbow-shaped line graph.

2.4 Traditional Supervised Classifiers for Financial Fraud

The below section explains the statistical models and techniques which we will employ during our research.

2.4.1 Logistic Regression

Logistic regression was originally proposed by Cox (1958). It is a probabilistic model designed for use on binary target variables and can be extended to multi-categorical targets (multiple logistic regression). The algorithm will learn from a training set of vector weights and a bias term, where each weight w_i is a number associated with a predictor feature. Weights distinguish a predictor's importance in the decision for classification. The bias term is added to the weighted inputs (Keselj, 2009). The weighted sum of this evidence can be described in 2.4 which is the dot product of two vectors. To ensure our probabilities lie between 0 and 1 we pass z through a logit function 2.5. This bounds outputs between 0 and 1 and has a well defined derivative.

$$z = w \cdot x + b \tag{2.4}$$

$$y = \sigma(z) = \frac{1}{1 + exp(z)} \tag{2.5}$$

To determine the best parameter values for our model we need to employ a loss function $L(\hat{y}, y)$ which calculates the magnitude in which the predicted value of our target $(\hat{y})$ is from our actual value y. This is known as conditional maximum likelihood estimation. We want to find the model parameters $(\beta_0,\ \beta_i)$ that maximise the log probability of y. We optimise our model using cross-entropy loss function 2.6. This heavily penalises misclassified instances.

$$L_{CE}(\hat{y}, y) = -y log \sigma(w \cdot x + b) + (1 - y) log(1 - \sigma(w \cdot x + b)) \tag{2.6}$$

To calculate the best function weights we use a limited Broyden–Fletcher Goldfarb–Shanno algorithm (LBFGS). The goal is to find a minimum or local minimum of a given objective function. Logistic regression has a convex loss function with a single minimum which simplifies the task at hand. LBFGS finds which direction we should descend in by preconditioning the objectives curvature information. The algorithm makes use of both the gradient of the objective function and its values. This is achieved by improving the Hessian matrix of the loss function using gradient evaluations (Dennis Jr & Schnabel, 1996). The limited version uses only the most recent m gradients which improves the computational performance of the operation.

2.4.2 Naive Bayes

Naive Bayes applies simplified learning for classification as it assumes all features are independent. Although this condition is unrealistic, there is empirical evidence that the method works well. Rish et al. (2001) shows that the accuracy of the classifier is not directly correlated with the degree of feature dependency, which may explain its empirically good performance.

The algorithm is based on Bayes Theorem. For a given feature $X = (x_1, x_2, ..., x_n)$ and target class C_k we define Bayes Theorem as 2.7. $P(X|C_k)$ is the posterior probability, $P(C_k|X)$ is the likelihood, $P(X)$ is the prior probability of the predictor and $P(C_k)$ is the prior probability of class (Fan & Fan, 2018).

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}, for k = 1, 2, ..., K \tag{2.7}$$

Naive Bayes adds the assumption of conditional independence 2.8.

$$P(x_i \mid x_{i+1}, ..., x_n \mid C_k) = P(x_i \mid C_k) \tag{2.8}$$

This assumption yields that $P(X|C_k)$ is the product of all given points posterior probabilities:

$$(X \mid C_k) = P(x_1, ..., x_n \mid C_k) = \prod_{i=1}^{n} P(x_i \mid C_k) \tag{2.9}$$

Therefore, the posterior probability can be defined as:

$$(C_k|X) = \frac{P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)}{P(X)} \tag{2.10}$$

The Naive Bayes model looks to find the maximum of $P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k)$ for each class k. This is defined as:

$$\hat{C} = \arg\max_{C_k} P(C_k) \prod_{i=1}^{n} P(x_i \mid C_k) \tag{2.11}$$

2.4.3 Decision Tree

Decision or classification tree modelling is commonly used in supervised classification. They are hierarchical models which identify an optimal strategy for classification. This is achieved by identifying variables that are important for classification. For example, in fraud detection, the amount of a transaction may be more important than the gender of the cardholders. A decision tree is made up of *"nodes"* that create a rooted tree. The initial node is defined as the *"root"*. Nodes in preceding layers that connect to other nodes are *"internal"*. Nodes that are only connected to a previous internal node are *"leaf"* or *"terminal"* nodes (Maimon & Rokach, 2014). Decision trees decide on which variables to split on by using impurity measures. This is calculated using Gini or Shannon's Entropy.

Entropy Based Approach

$$H_i = - \sum_{k=1,\, p_{i,k} \neq 0}^{n} p_{i,k} \log_2 (p_{i,k}) \tag{2.12}$$

Equation 2.12 defines Shannon Entropy. p_i is the frequentest probability of class i in our set of observations (fraud or genuine). Higher entropy values are associated with lower levels of purity. We can think of this as variables that do not discriminate our target variable much, for example, if 50% of fraudulent transactions were from female participants and 50% male we would expect the feature to have high entropy. Using only gender to determine our fraud classification would be the same as random guessing (Maimon & Rokach, 2014).

Next, we calculate information gain. Information gain of a new variable (Y) is calculated by subtracting the entropy of Y given X from the entropy of Y. Decision trees decide which variables to split on by maximising the calculated information gain.

Gini Based Approach

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2 \tag{2.13}$$

Gini impurity (2.13) is an alternative purity measure. The Gini itself is a probability measure where the higher the Gini coefficient the higher the purity between nodes (Maimon & Rokach, 2014). Gini impurity is computationally more efficient compared to an Entropy based approach.

2.4.4 Random Forest

Random Forest is an ensemble learning technique that can be applied to classification tree models using a method known as *bagging*. Decision trees suffer from high variance, meaning small changes to inputs or the dataset can result in different decisions and tree architectures (James et al., 2013). Bagging aims to reduce the variance of any statistical learning method. It achieves this by using many training sets, building separate predictive models for each set and averaging the resulting predictions. We often do not have access to many different training sets. We can use *bootstrapping* to account for this. Bootstrapping generates B separate training sets. We train our model on the b^{th} set and average all predictions.

A Random Forest is an improvement on bagging by implementing a mechanism to decorrelate the trees (James et al., 2013). This decorrelation method is achieved by choosing a random m predictors for splitting from our full set of predictors p. A new sample is taken at each split, commonly chosen as $m = \sqrt{p}$. The intuition is to eliminate a situation where we have multiple trees with one strong predictor being the top split. In this instance, the multiple bagged trees would have similar starting nodes and hence be correlated by this fact. Averaging many correlated trees leads to a higher variance compared to uncorrelated trees (James et al., 2013).

The model's performance over several trees is calculating using out-of-bag (OOB) error. On average each bagged tree will use over 70% of the dataset, the remaining unused observations are OOB. We can predict the i^{th} observation using the trees where that observation was OOB (James et al., 2013) which is akin to leave-one-out cross-validation i.e., train on all observations expect i, test on i (Mosteller & Tukey, 1968).

2.4.5 XGBoost

Boosting is similar to bagging as it involves combining a multitude of decision trees to influence a prediction. Different to bagging, boosting looks to train trees sequentially using the residuals from each subsequent tree. It is an ensemble of weak learners meaning trees are strongly correlated with each other. Boosting models focus on misclassification areas and improving these areas over time. A shrinkage parameter (λ) allows different shaped trees to improve the residuals (Mosteller & Tukey, 1968).

The gradient boosting algorithm identifies areas subject to higher misclassification by using gradients in the loss function. Extreme Gradient Boosting (Chen & Guestrin, 2016) uses a more regularised model formalisation to improve performance and reduce over-fitting. We define a trees output as:

$$f(x) = w_q(x_i) \tag{2.14}$$

x represents an input vector and w_q is the score of a corresponding leaf q. The output of the full tree ensemble is hence:

$$y_i = \sum_{k=1}^{K} f_k(x_i) \tag{2.15}$$

XGBoost minimises the below objective function (J) per step t:

$$J(t) = \sum_{i=1}^{n} L(y_i, \hat{y_i}^{t-1} + f_t(x_i)) + \sum_{i=1}^{t} \Omega(f_i) \tag{2.16}$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{2.17}$$

The initial term looks into the loss function between predicted and real class. The second term is the regularisation term to prevent the model from overfitting. Complexity (Ω) considers the number of leaves in the tree (T). γ is a pseudo-regularisation term. λ relates to a shrinkage parameter defined as the L2 norm for leaf weights (Dimitrakopoulos et al., 2018).

XGBoost uses the gradients of the second-order approximation of the given loss function to optimise the best weight values, creating an objective function below where g_i and h_i relate to gradient statistics for the loss function and I is the set of leaves (Dimitrakopoulos et al., 2018).

$$J(t) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda} + \gamma T \tag{2.18}$$

2.4.6 Empirical Results of Supervised Classification for Fraud Detection

Which classifier works best for classifying fraud? This is a key research question covered throughout anomaly detection literature. Shen et al. (2007) study used a similar credit card transactional dataset comparing decision tree. They found logistic regression to outperform decision tree methods. Ensemble methods are often cited as well-performing methods when applied to financial fraud datasets. Sohony et al. (2018) conducted an empirical approach proposing an ensemble model that combines Random Forest and Feed-Forward Networks, showing improved results compared to baseline methods. shimin2020xgboost showed that using SMOTE to fully balance the dataset combined with an XGBoost classifier resulted in a state-of-the-art performance and improvements in the classifiers recall score. A limitation of this paper is that they ran single experiments and did not cross-validate their results. On the same dataset, Randhawa et al. (2018) show that the use of boosting algorithm AdaBoost ("AdaBoost", 2009) outperforming standard models (Naive Bayes, Logistic Regression) even in the presence of noise. Xuan et al. (2018) used Random Forest and showed Gini impurity strategy to significantly improve the F1 score of the classifier. They employed undersampling, however they used data set with a much larger number of fraudulent transactions ($n > 81000$) compared to our research problem.

Chapter 3

Design and methodology

This chapter aims to outline the quantitative methodology used for the empirical study. We outline our primary null and alternative hypothesis. Description of the dataset and techniques to process this data is outlined. An experimental framework design for gathering evidence for our hypothesis is formalised. The optimisation strategy of tree-based classifiers using hyper-parameter tuning is detailed. GAN training evaluation and methods for determining the inclusion of cluster information is specified. Finally, we include a summary of methods used including strengths and limitations.

3.1 Hypothesis

H_1: Employing a Conditional Tabular Generative Adversarial Network as a data-level oversampling method will result in a significant increase in the F1 score of a classifier compared to traditional oversampling methods.

H_0: Employing a Conditional Tabular Generative Adversarial Network as a data-level oversampling method will result in no significant increase in the F1 score of a classifier compared to traditional oversampling methods

3.2 Experimental Framework

Figure 3.1, A.1 represents our experimental framework for testing our upsampling methods across classifiers and imbalance ratios. Let $\mathcal{D}$ represent our full dataset. We randomly split D into 5 identically sized folds. Let $X \subset \mathcal{D} = \sum_{j \neq i}^{5} x_j$ Where i represents the fold used for testing. X is of comprised of $X_{majority}$ and $X_{minority,real}$. $X_{majority}$ represents genuine transactions. $X_{minority,real}$ represents real tagged fraudulent transactions. We pass $X_{minority,real}$ to the given upsampling method for synthetic data generation. The number of new samples to be generated (N_{syn}) is a function of the sets original imbalance ratio IR_X and the desired imbalance ratio IR_Y. We denote N as the number of observations.

$$IR_X = \frac{N_{majority}}{N_{minority,real}} \tag{3.1}$$

$$\theta = IR_Y - IR_X \tag{3.2}$$

$$N_{syn} = \theta * N_{majority} \tag{3.3}$$

Adding N_{syn} synthetic samples creates new set $X_{minority,syn}$, which is joined to our original set. This is added to our original set X. We denote the resulting set as $Y = X_{majority} + X_{minority,real} + X_{minority,syn}$. We train our classifier Ci on training set Y and test on our remaining fold x_i using F1 score for performance evaluation (3.7). This is repeated 5 times and an average F1 score across folds is stored. We repeat this over a number of imbalance ratios and classifiers. A list of chosen imbalance ratios is detailed in section 5.3. Given the relatively small number of minority instances (492), within our dataset we choose to only use 5 folds, however the solution can be generalised for any k number of folds.

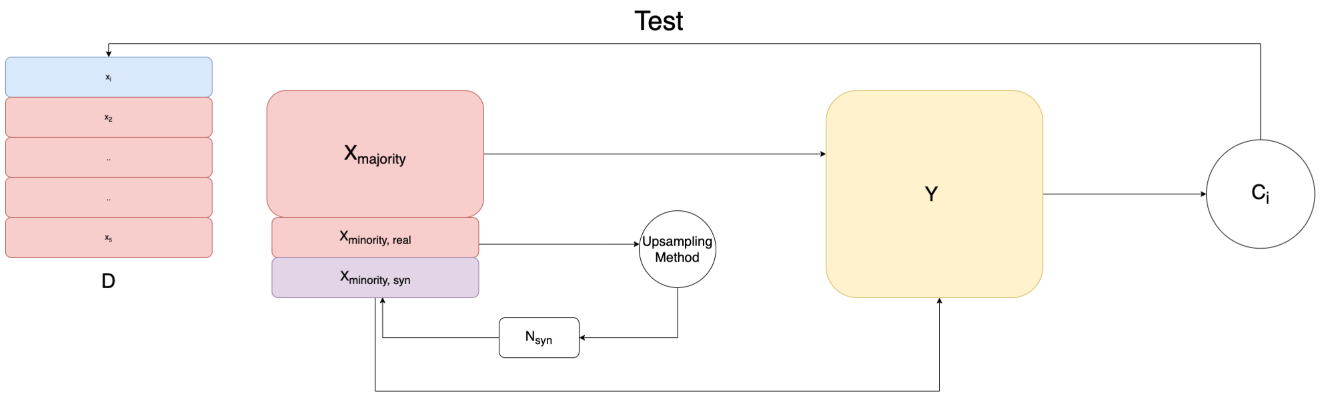


Figure 3.1: Upsampling Framework Using 5-Fold Cross Validation

3.3 Dataset Description

Dal Pozzolo et al. (2014) dataset containing fraudulent credit card attempts is often used within this domain (Sisodia et al., 2017; Tanaka & Aranha, 2019; Ba, 2019; Fiore et al., 2019). The dataset represents 284,807 financial transactions captured by a financial institution over two days in 2013. 492 of these are fraudulent (0.172%). Given the high sensitivity of the data, the majority of feature names are omitted to protect the identity of the persons. The features of the dataset are given as so:

- ‘Time’ - Numerical - Number of seconds elapsed between this transaction and the first transaction in the dataset.
- ‘Amount’ - Numerical - Represents the financial amount taken from the card-holders account.
- ‘Class’ - Logical - Identifies if a transaction is fraudulent. 1 represents fraud, 0 represents non-fraud.
- ‘V1’ - ‘V28’ - Numerical - principal components obtained by PCA.

3.4 Data Exploration and Pre-processing

3.4.1 Exploratory Data Analysis and Feature Engineering

Time

The time variable was transformed to identify the number of days the dataset encompasses using equation 3.4.1. The maximum value for variable *Days* was 1.99 i.e. the transactions span 2 days.

$$\text{Hours} = \frac{Time}{3600}$$
$$\text{Days} = \frac{Hours}{24}$$

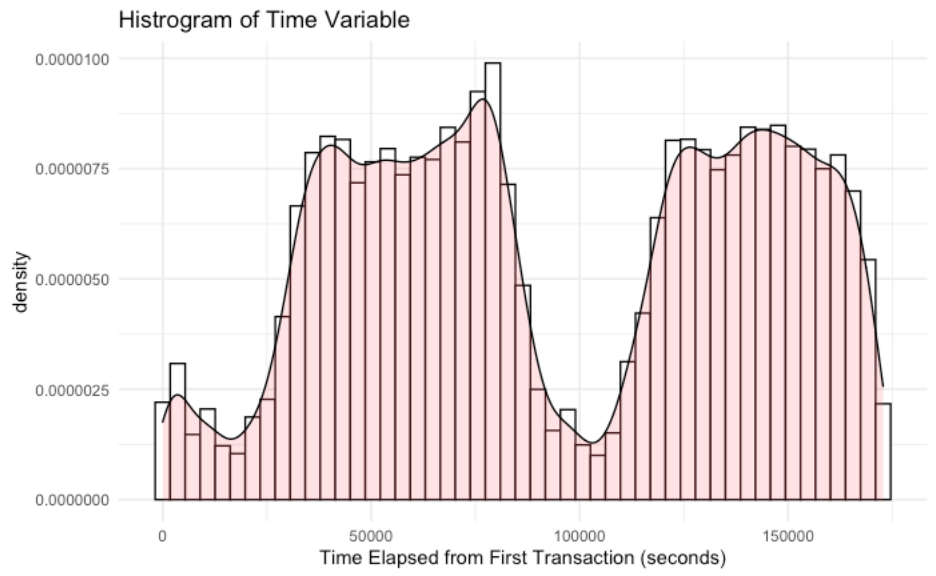


Figure 3.2: Histogram and Distribution of Time

Figure 3.2 shows a histogram of the variable. The low density areas represent off-peak times in terms of consumer behaviour. From this we create a new logical vector based on a transactions relative frequency within the time range. $Denisty > 0.000005$ is the positive class indicating *'on-peak'* transactions. Otherwise a transaction is *'off-peak'*

Duplicated rows

927 duplicated rows were identified. A duplicate row is defined as identical values across all records. 911 relate to normal transactions, 16 relate to fraudulent transactions. Given there is a possibility of multiple re-attempts for transactions and the small proportion we did not remove these instances.

Amount - Distribution and Skewness

Visual inspection of transaction amount by class using 3.3 shows very right skewed distributions. Groeneveld and Meeden (1984) defines skewness as the degree of asymmetry for a distribution. This is denoted in 3.4.1, where μ_i represents the i^{ith} central moment.

$$\gamma_1 = \frac{\mu_3}{\mu_2^{3/2}} = \frac{\mu_3}{\sigma^3}$$

We standardise this value by dividing by the variable's standard error 3.4.1.

$$\operatorname{var}(\bar{x}) = \frac{\sigma^2}{n}$$

Genuine transactions were determined as having a standardised skew of 36.25. This is outside the bound needed to indicate a normal univariate distribution $[0-2]$ (George, 2011). We see a similar distributional shape for Fraud class. We observe a higher frequency of high value transaction. 7.1% of fraud class is greater than 500$ while 3.3% of genuine transactions showed this amount.

Table 3.1 looks into our fraudulent transaction data grouped by amount. We notice a number of non-unique amounts. 113 fraudulent transactions only show as \$1 and 27% show as \$0. It is a common occurrence for fraudulent actors to test credit card information multiple times using smaller amounts, which could explain this phenomenon.

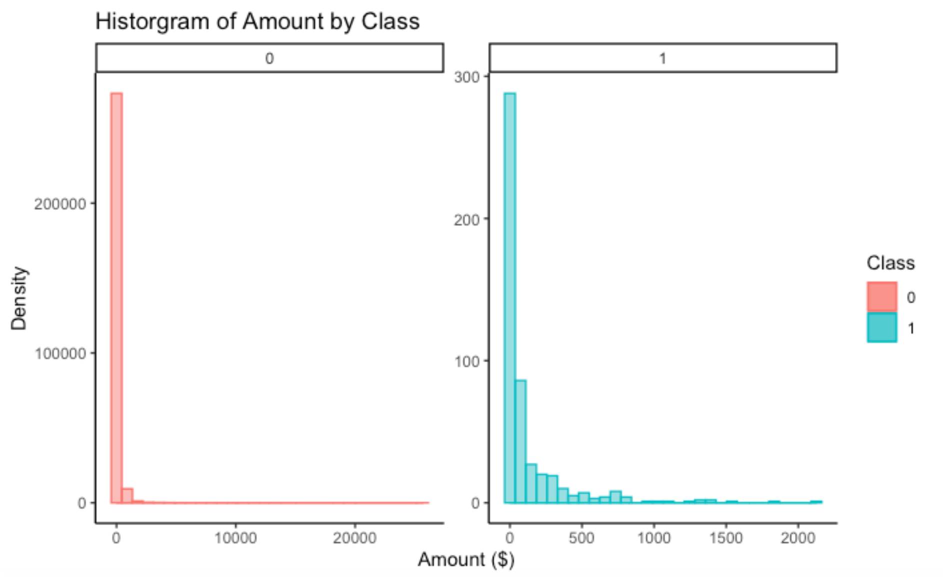


Figure 3.3: Histogram of Amount by Class

Table 3.1: Frequency Table: Amount

Amount ($)	n	% of Total Fraud
1	113	22.97%
0	27	5.49%
99	27	5.49%
0.76	17	3.46%
Other	308	62.5%

3.5 Hyper-parameter tuning

Our analysis considers multiple tree-based classifiers with varying parameters. Our upsampling framework will consider the best performing parameter combination for each. We determine this by employing a grid search with 5-fold cross validation. Each combination of our parameters is passed through a 5-fold cross validation framework to determine the best combination. The whole dataset is considered and randomly shuffled to create 80% for training each combination and 20% for testing. The combination which maximises the evaluation metric will be kept as our parameters for additional training. We evaluate based on F1 score. We employed tuning framework on our tree-based classifiers. Decision tree parameters are defined in table 3.2. Random forest classifiers are defined in table 3.3. XGBoost parameters are defined in table 3.4.

Table 3.2: Parameters for Decision Tree

Random Forest Parameters	**Parameter Values (Step)**
Criterion	entropy, gini
Tree Depth	4-12 (2)

Table 3.3: Parameters for Random Forest

Random Forest Parameters	**Parameter Values (Step)**
Max Depth	5,8,15,25
Min Samples	3-5 (1)
Min Samples Split	8-12 (2)
Number of estimators	100-500 (200)

Table 3.4: Parameters for XGBoost

XGBoost Parameters	**Parameter Values (Step)**
Estimators	50-100 (1)
Learning Rate (ETA)	.025 -.05 (.025)
Tree Depth	1-14 (1)
Minimum Child Weight	1-6 (1)
Subsample	.5-1 (.05)
Gamma	.5-1 (.05)
Colsample by Tree	0.5-1 (.05)

3.6 GAN Performance Analysis

GAN performance is measured using framework detailed in Figure 3.4, A.2. Post training, we create a synthetic dataset with the same number of observations as the real dataset ($n = 492$) our generative network G_z. The sets are both randomly split, setting half for training and half for testing. An XGBoost classifier is trained on training set and accuracy (3.4) is used as a performance metric. The central idea is to obtain a result close to 0.5, as this indicates the classifiers performance is akin to random guessing. This would result in synthetic data that is difficult to distinguish from real data.

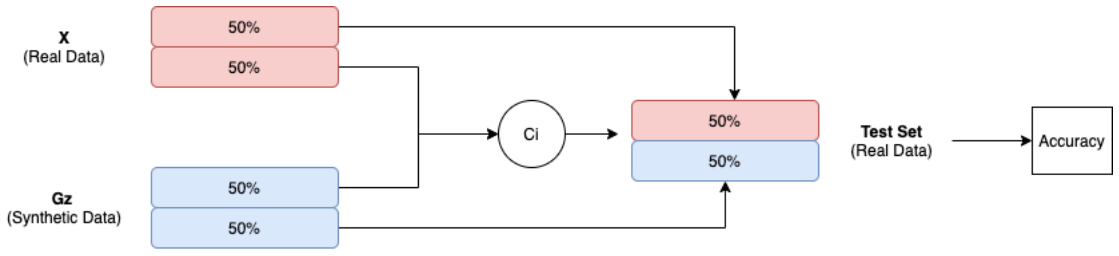


Figure 3.4: GAN Performance Framework

3.6.1 Cluster information for training

For GAN training, our generative model G_z is based a random initialization of a noise vector dZ. Hence, in practice we may get varying results when generating synthetic data. To account for this we repeated our framework 3.4 10 times for 5 different training epoch values ranging from 100 to 500. We use CTGAN for this experiment. This is repeated for training using cluster information and non-cluster information. The results in a distribution of scores to analyse. To determine if we should continue to use cluster information we combined all results across epochs for both groups and conducted a t-test to determine if training with cluster information is significantly different to training without.

3.7 Performance Evaluation

3.7.1 Evaluating Classifiers

Prediction outcome

		p	**n**	**total**
	p′	True Positive	False Negative	P′
actual value	**n′**	False Positive	True Negative	N′
	total	P	N	

The above represents a confusion matrix. This is a tabular representation of the possible outcomes of a binary classification problem. We define the positive class as a fraudulent transaction and the negative class as genuine.

- Accuracy: Calculates the ratio of correct predictions over all attempts.
- Precision: Calculates the true positive rate.
- Recall: Measures the proportion of actual positives that were identified correctly.
- F1 score: Calculates the harmonic mean between precision and recall.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.4}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

$$F1 = \frac{2 * (precision * recall)}{precision + recall} \tag{3.7}$$

Accuracy (3.4) is not useful for imbalanced classification. For example, if the classifier predicted everything as genuine and genuine transactions represent 99% of the dataset, the accuracy would be 99% even though it did not correctly classify any fraudulent observations.

For imbalanced datasets it is most common to use a performance metric that is based on the classifiers general ability to detect both genuine and fraudulent instances. For our research, we will focus on F1 score given it is the most commonly used across similar papers (Fiore et al., 2019; Douzas & Bacao, 2018; Gangwar & Ravi, 2019).

3.8 5-Fold Cross Validation

In our experimental framework and training baseline classifier we employ the use of 5-fold cross validation. This obtains an average score F1 score to reduce variance in our results. The dataset is randomly split into 5 equal sized folds. We train our classifier C_i on $\sum_{j}^{k-1} x_j$ where j $\neq i$. This results in 5 separate F1 scores. The average of these scores are taken as our final result.

3.9 Ranking Oversampling Methods

Our experimental framework involves us testing 5 classifiers across 13 different imbalance ratios and 4 upsampling methods. This results in 260 separate experiments undertaken. Across each classifier and imbalance ratio where synthetic data generation took place we rank the best performing method. This results in 65 rank data. We conduct a Friedman's test to determine significance in the difference across these methods. These results will aid us in gathering evidence toward our research question if CTGAN can outperform traditional methods.

3.9.1 Summary and Limitations

This chapter successfully details the methods needed to obtain evidence needed for our research hypothesis and sub-research questions. We use static parameters across our experimental framework. Hyperparameter tuning allows us to ensure we are using paramters that gives optimal performance combined with computational efficiency. A limitation of this framework is the use of 5-fold cross validation. Given we have a very limited number of fraud observations, we wanted to ensure each test set was sufficiently large enough for testing. To improve the framework one may increase to 10 fold to improve the validity of the average F1 score. GAN training framework ensures we have a scalable method of measuring the quality of our synthetic data. Statistically comparing groups with and without cluster information provides evidence if cluster information will improve our synthetic data. Considering GAN training involves random intialisation, repeated experiments over epochs acccounts for confounding factors associated with training i.e. instead of simply setting a reproducible seed we want to determine the distribution of results if we were to repeat random intialisation. This creates a limitation as this particular section of experiments are not exactly reproducible.

Chapter 4

Results, evaluation and discussion

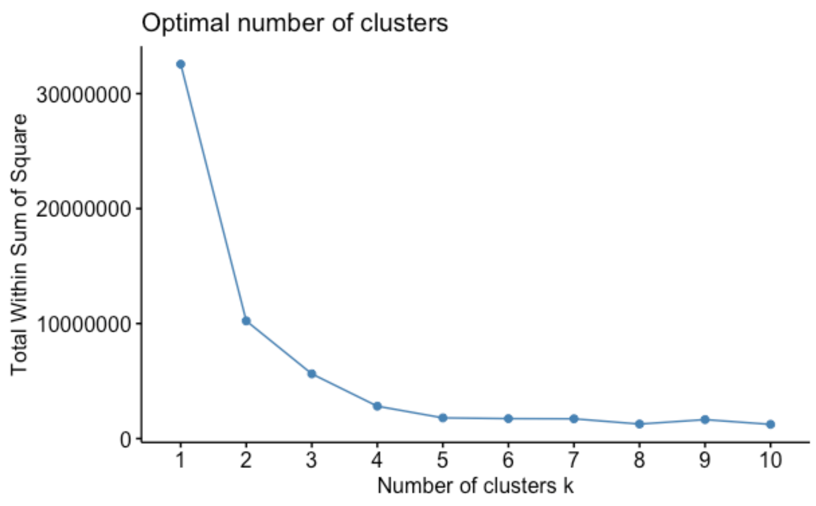
The existence of extreme class imbalance and lack of data is a significant problem in the performance of supervised classifiers for fraud detection. The ability to remedy this imbalance and improve classfier performance is a key area of research as it transalates to great financial benefits for financial institutions and businesses, given the high cost associated with missed fraud. This research focuses on the improvement of classifier performance by comparing the use of GAN technologies against traditional, nearest neighbour methods. Further, we explore the relationship between datasets imbalance ratio and performance to understand if there exists an optimal point of adding additional synthetic samples. Finally, we explore the use of clustering information on synthetic data generation using GANs and accounting for counfounding factors.

In this chapter we report on our experimental results and discuss them with respect to basineline methods. Limitations and critiques will be highlighted in the summary section.

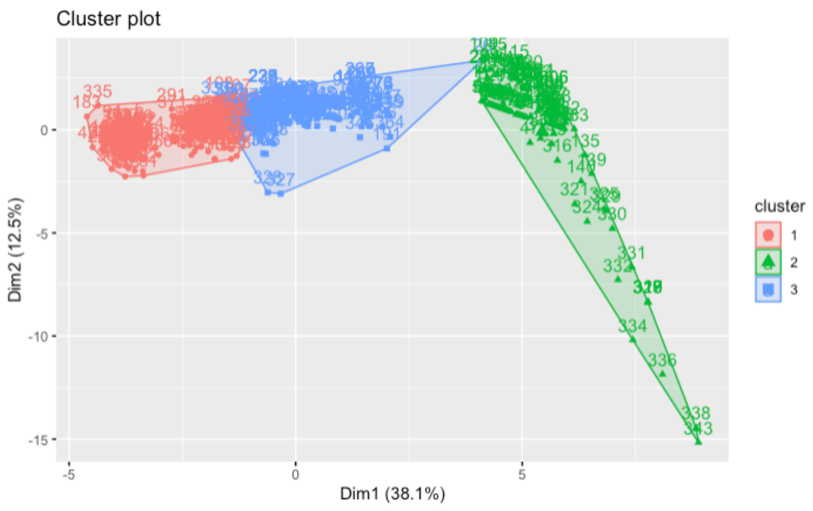
4.1 Results

4.1.1 Clustering Results

From Figure 4.1 we observe that the the greatest reduction of within sum of squares is achieved when moving from 1 to 2 clusters. We observe a smaller but significant decline from 2 to 3, hence we choose k=3. Visually inspecting this in plot (b) shows well defined clusters using k=3.



(a) Elbow Plot



(b) Visualisation of Clusters (k = 3)

Figure 4.1: Cluster Analysis Results

4.1.2 Hyperparameter Tuning Results

Iterating through parameters in Table 3.2 best F1 score for a simple decision tree was found using a Gini strategy combined with a maximum tree depth of 4.

For the Random Forest algorithm, we noted the best performance using a Gini approach and a maximum depth of 5 using 200 trees ($F = 0.77$). Random forest classier was computationally expensive. A varying number of estimators was tested to determine the impact on performance as described in Table 4.1. We observe a 0.009 difference from 50 to 200 trees. The computational cost of training is 25% using 50 trees, hence we decided to use this for training.

Table 4.1: Random Forests: Average F1 Performance Across Different Estimators

No. estimators (trees)	F1 score
25	0.7578
50	0.7694
200	0.7703
500	0.7676

XGBoost showed the best performance using a maximal tree depth of 69 trees. The optimal eta value was .125 with a gamma value of .7. The best randomly selected fraction of features used to train each tree was .5. The best randomly selected fraction of features used in every node to train each tree was 0.9.

4.1.3 GAN Training Results

CTGAN Cluster Information and Training Epochs

Figure 4.2 shows the accuracy of our results across different epochs. Across both groups training using 100 epochs showed the smallest interquartile range of results. Non-Cluster group showed lowest median accuracy at 200 epochs *(Mdn = .91)*. Non-clustered median accuracy was also lowest at 200 epochs *(Mdn = .92)*.

A Sharpiro-Wilk test (Shapiro & Wilk, 1965) was undertaken to determine the t-test assumption of normality. Both groups were found to be approximately normal: Clustered Group *(W = .978, p= .44)*, Non-Clustered Group *(W = .973, p= .32)*. An F test was applied and concluded no significant difference between each group's variance, *F(49,49) = 0.86, p = 0.61*. A t-test was conducted given we have evidence to support the tests assumptions between groups. Accuracy results training with cluster information *(M =.93, SD = .02)* compared to removing cluster information *(M =.915, SD = .02)* demonstrated significantly higher results, *t(97) =3.31, p .05*. Based on these results we decided to leave out cluster infromation when training our GANs.

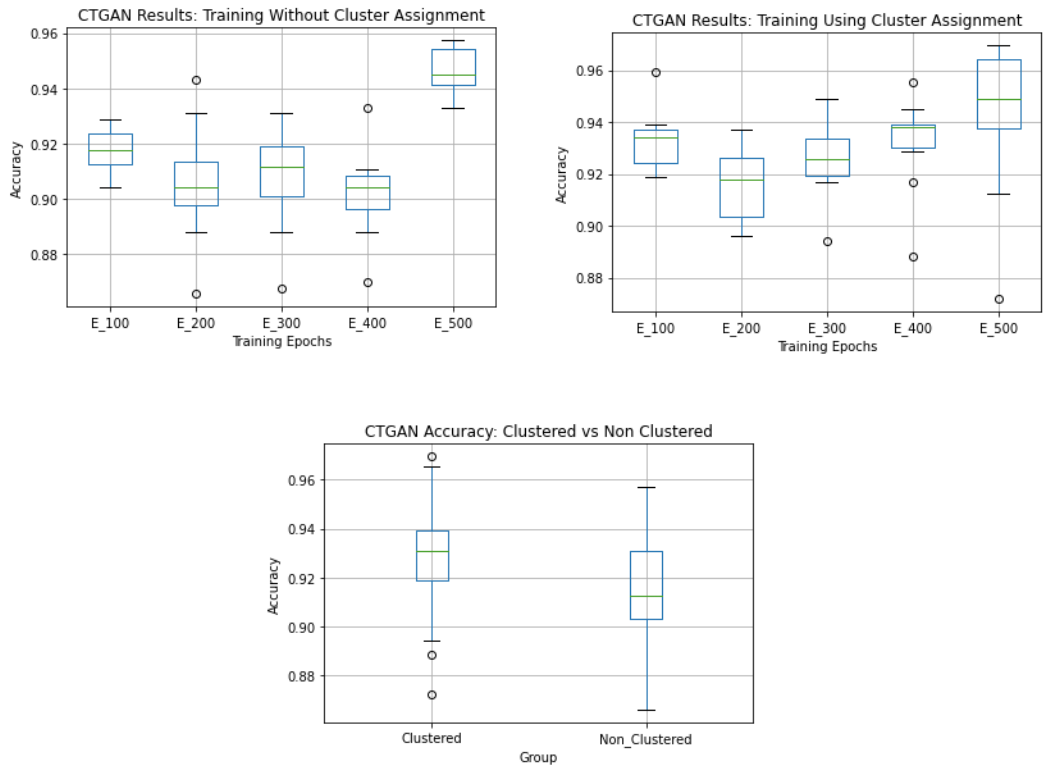


Figure 4.2: CTGAN: Boxplot of Accuracy Across Cluster Groups and Epochs

WGAN Training Epochs

Given results in section 4.1.3 we trained our WGAN without any cluster information. Figure 4.3 denotes WGAN accuracy across each 100 training epochs repeated 10 times. 400 epochs showed lowest median accuracy (Mdn = .982). Increasing to 500 epochs showed a smaller interquartile range. For our experiments 500 epochs was used as standard for use in our experimental framework.

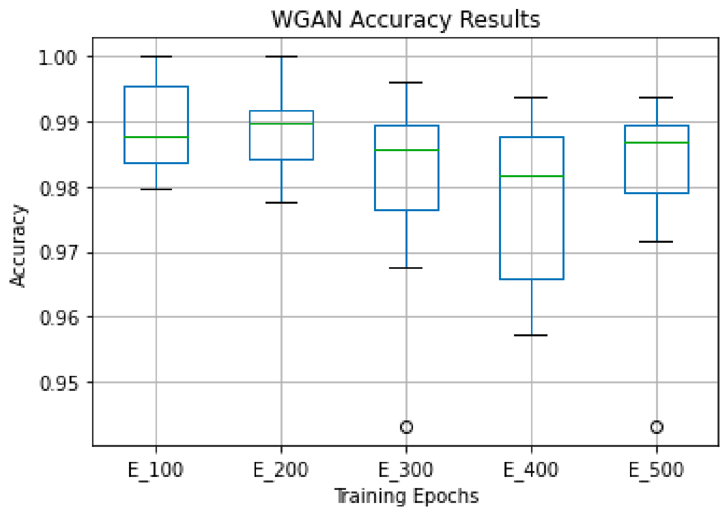
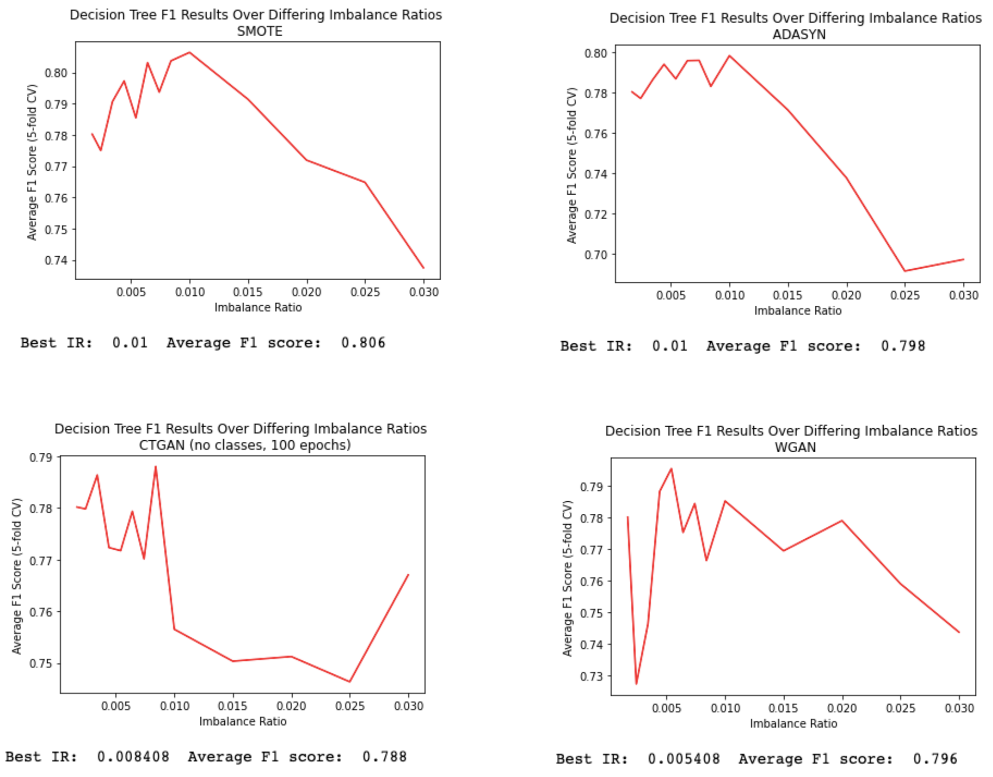


Figure 4.3: WGAN: Boxplot of Accuracy Across Epochs

4.1.4 Upsampling Framework Results

Decision Tree

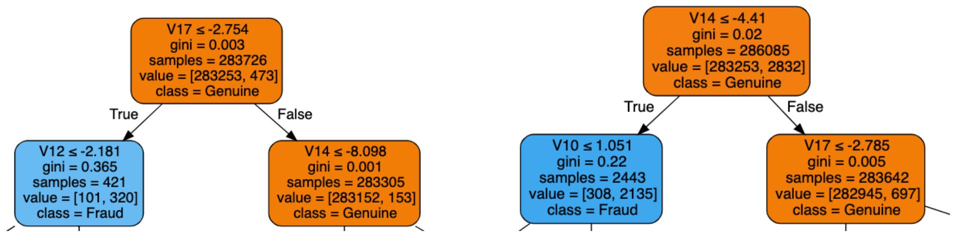
Applying both traditional and GAN oversampling methods had a positive effect on average F1 scores compared to baseline (IR = .017). Table 4.4 shows SMOTE having the highest average F1 score across all upsampling methods ($F1 = .8064$). Traditional methods (SMOTE and ADASYN) resulted in optimal performance using an imbalance ratio of .01. WGAN reach a maximum of .0054 and CTGAN at .7864. Figure 5.1 shows that SMOTE and ADASYN displays an almost concave relationship between average F1 score and imbalance ratio, both reaching a peak and subsequent sharp decline. GANs exhibited a more volatile relationship.



```
Best IR:  0.01  Average F1 score:  0.806
Best IR:  0.01  Average F1 score:  0.798
Best IR:  0.008408  Average F1 score:  0.788
Best IR:  0.005408  Average F1 score:  0.796
```

Figure 4.4: Decision Tree: Average F1 Scores over Imbalance Ratios

Feature V17 and V14 showed the highest importance for the original dataset with no oversampling. To further investigate the effect of our upsampling method on the tree structure we compare our original tree and refit a decision tree on our best over-sampling method and imbalance ratio (SMOTE, IR = .1). From Figure 4.5 we observe a change in the first split from V17 to V14.



(a) First Split: Orginal Dataset

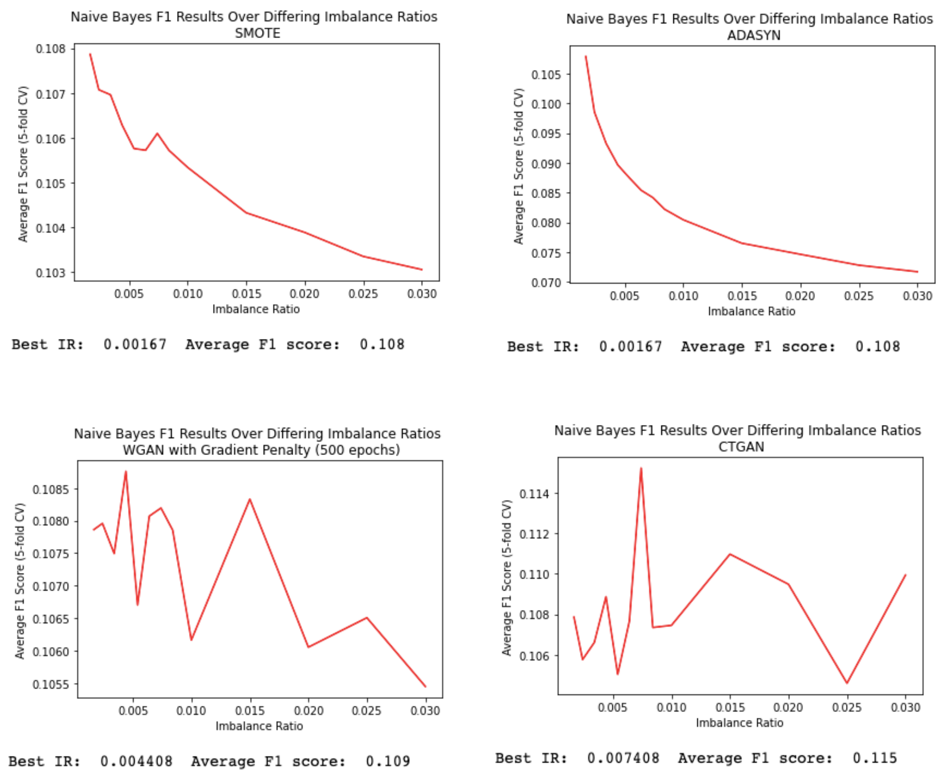
(b) First Split: Dataset with IR = .1 using SMOTE

Figure 4.5: Comparing First Decision Tree Split

Naive Bayes

Naive Bayes classifier showed overall poor performance with a maximum average F1 score reaching .116 using CTGAN. Oversampling using GAN methods achieved minor improvement from the original imbalance ratio. Traditional oversampling methods did not improve performance. Table 5.2 shows both SMOTE and ADASYN having optimal average F1 score using our original imbalance ratio (IR = .0017). WGAN achieved an optimal score using an imbalance ratio of .0044. CTGAN showed the best score at .0074.

Figure 4.6 shows SMOTE and ADASYN to have a negative correlation between average F1 performance and imbalance ratio. GAN methods exhibited more random results across imbalance ratios.



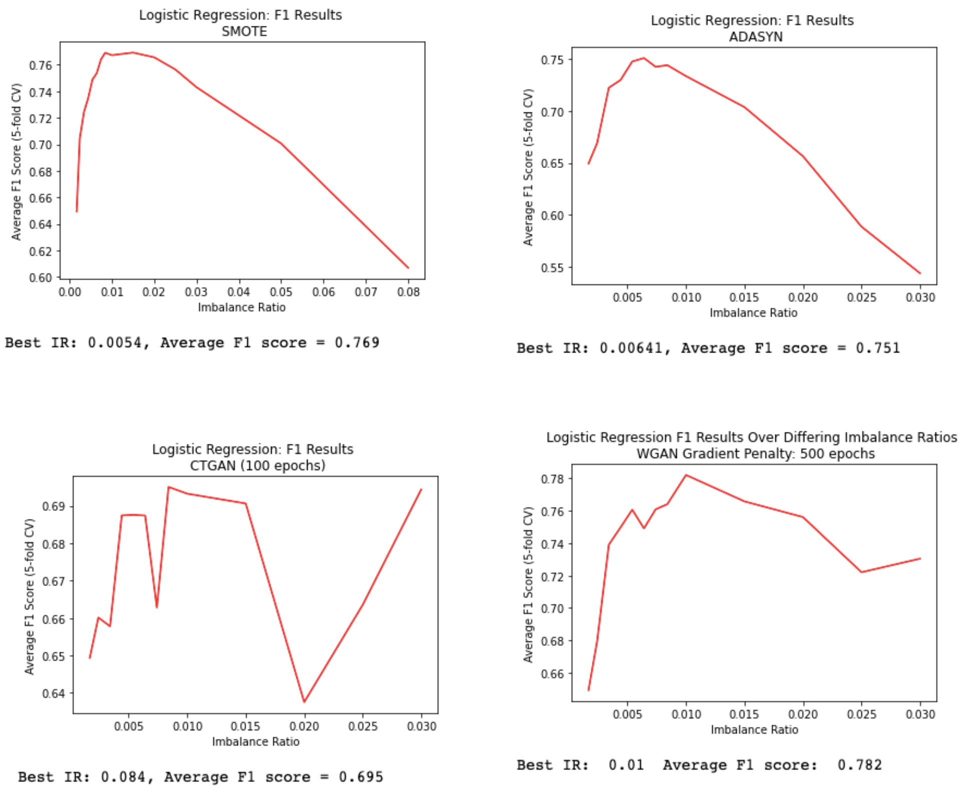
```
Best IR:  0.00167  Average F1 score:  0.108
Best IR:  0.00167  Average F1 score:  0.108
Best IR:  0.004408  Average F1 score:  0.109
Best IR:  0.007408  Average F1 score:  0.115
```

Figure 4.6: Naive Bayes: Average F1 Scores over Imbalance Ratios

Logistic Regression

Logistic Regression classifier achieved best results using WGAN by increaisng our imbalance ratio to .01 (F1 = .7821). Both traditional and GAN methods improved the classifiers average F1 score by increasing the imbalance ratio. Table **??** shows that SMOTE reached an optimal imbalance ratio at .0054 (F1 = .7489) and ADASYN at .0064 (F1 = .7478). Across imbalance ratios CTGAN showed poorest maximum performance, reaching a maximum at .0084 (F1 = .6951).

Figure 4.6 shows SMOTE and ADASYN having a concave relationship between imbalance ratios and average F1 scores. Both methods hit a distinct peak and eventual decline. WGAN shows a similar pattern, yet performance is more stable across imbalance ratios. CTGAN exhibits a more volatile relationship. The majority of average F1 scores are higher when oversampling compared to applying no oversampling.



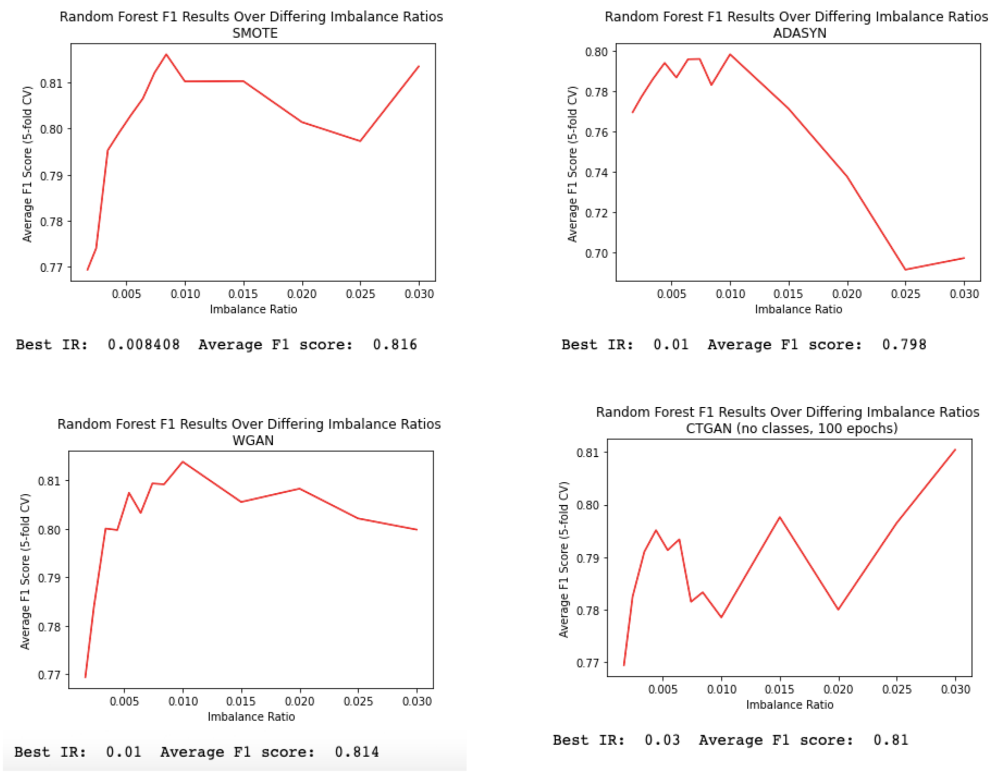
```
Best IR: 0.0054, Average F1 score = 0.769
Best IR: 0.00641, Average F1 score = 0.751
Best IR: 0.084, Average F1 score = 0.695
Best IR:  0.01  Average F1 score:  0.782
```

Figure 4.7: Logistic Regression: Average F1 Scores Across Imbalance Ratios by Oversampling Strategy

Random Forest

Random Forest classifier achieved best results using SMOTE by increasing our imbalance ratio to .0084 (F1 = .8161). Both traditional and GAN methods improved the classifiers average F1 score by increasing the imbalance ratio. Table 5.4 shows that ADASYN reached an optimal imbalance ratio at .01 (F1 = .7982). We observed the same for CTGAN (F1 = .8139). WGAN achieved optimal results using an imbalance ratio of .0074 (F1 = .8094)

Figure 4.8 shows all upsampling methods result in better average F1 scores compared to the original imbalance ratio. SMOTE shows F1 score reaches an initial peak at IR = .0084. Unlike Naive Bayes and Logistic Regression we do not see a sharp decline in performance, rather a weaker negative relationship with performance improving again at IR = .3. ADASYN reaches a maximum at IR = .1 and a subsequent large decline in performance. WGAN shows a similar function shape with performance reaching a maximum at IR = .1. Performance declines at a slower rate compared to ADASYN. CTGAN shows a more volatile relationship across imbalance ratios.



```
Best IR:  0.008408  Average F1 score:  0.816
```

```
Best IR:  0.01  Average F1 score:  0.798
```

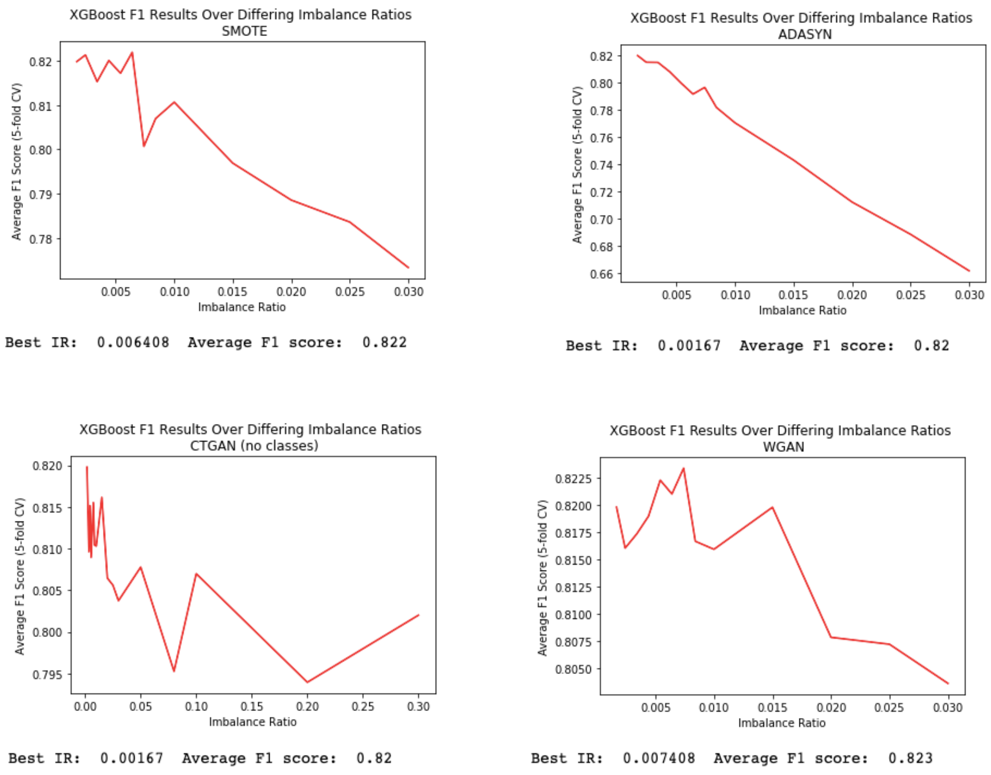
```
Best IR:  0.01  Average F1 score:  0.814
```

```
Best IR:  0.03  Average F1 score:  0.81
```

Figure 4.8: Random Forest: Average F1 Scores Across Imbalance Ratios by Upsampling Strategy

XGBoost

XGBoost did not benefit greatly from the use of oversampling methods. ADASYN and CTGAN both showed the best results using the original imbalance ratio (IR = .0017) with no oversampling. Table 5.5 shows that SMOTE and WGAN exhibited minor performance improvements. WGAN, using an imbalance ratio of .0074, exhibits the best overall performance (F1 = .8234). Figure 4.9 presents that any increases in imbalance ratio over .0074 have a negative relationship. ADASYN shows a near linear negative relationship. GAN methods show more volatile performance, similar to all other classifiers.



```
Best IR:  0.006408  Average F1 score:  0.822
Best IR:  0.00167  Average F1 score:  0.82
Best IR:  0.00167  Average F1 score:  0.82
Best IR:  0.007408  Average F1 score:  0.823
```

Figure 4.9: XGBoost: Average F1 Scores Across Imbalance Ratios by Upsampling Strategy

4.1.5 Distributional Analysis

Using a set of 2359 synthetic samples we investigated the distributional changes that occur when applying our best performing GAN and traditional upsampling methods. Figure 4.10 shows that synthetic data shows near identical distributional properties to the non-synthetic set. WGAN (Figure 4.11) creates synthetic data that follows a more general normal distribution for both V14 and V17. The scatterplot shows a more general shape, creating synthetic data which is more centred compared to SMOTE which retains the specific shape of the original dataset with less empty space (as the method fills in nearest neighbour distances).

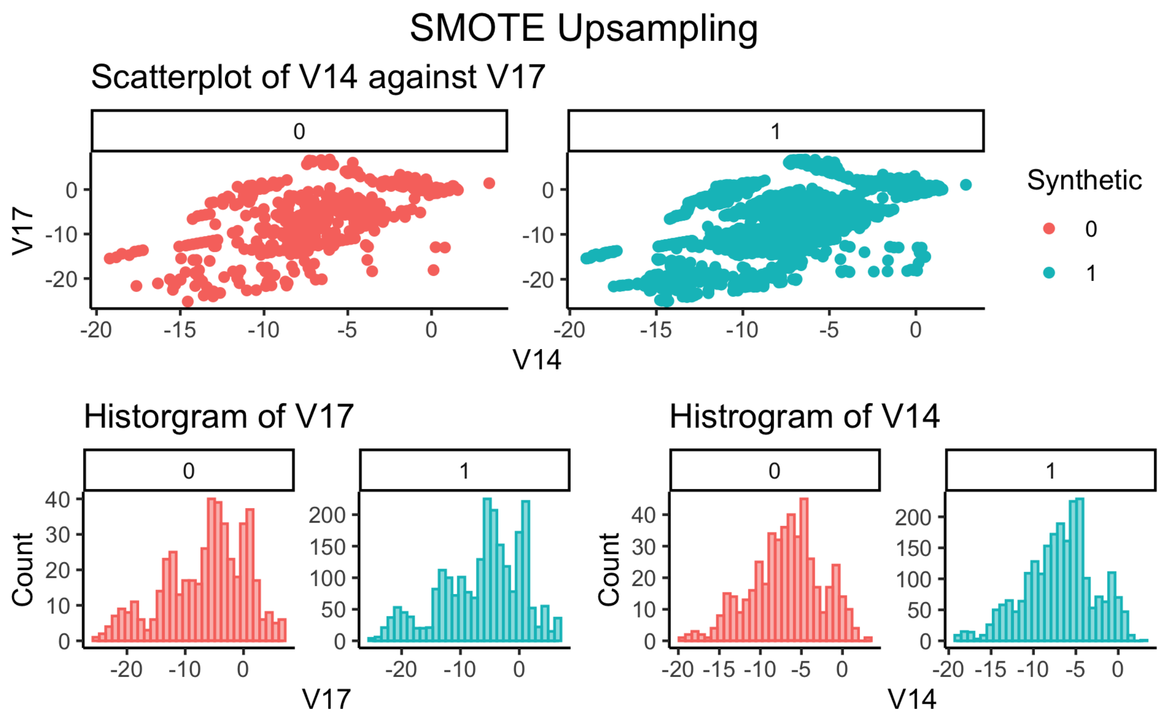


Figure 4.10: SMOTE: Distribution and Relationships of Important Features

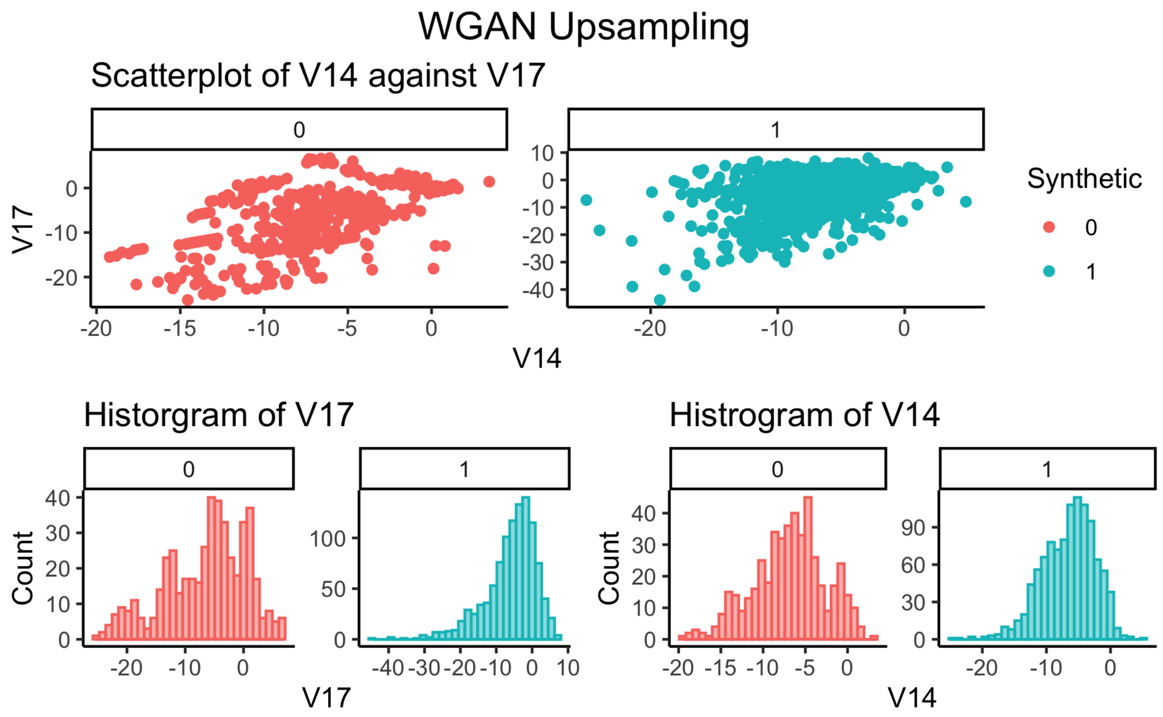


Figure 4.11: WGAN: Distribution and Relationships of Important Features

4.1.6 Ranking Oversampling Methods

Average F1 score were ranked across classifiers for each imbalance ratio where oversampling was conducted. Figure 4.12 shows ranked results across all experiments in a boxplot. A Friedman test was conducted to determine if the oversampling methods were ranked significantly different. Results indicated a differential rank across the 4 oversampling methods, $\chi^2(3) = 23.5, p < .05$. WGAN had the best mean rank ($M = 1.8$) followed closely by SMOTE ($M = 1.85$). ADASYN was most often ranked lowest ($M = 3.42$) with CTGAN most often ranking 3^{rd}(M = 2.93).

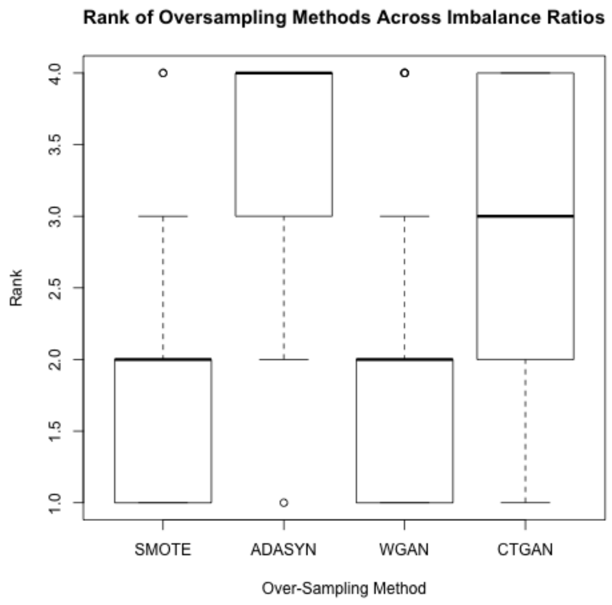


Figure 4.12: Boxplot: Ranked Performance Across Oversampling Methods

4.1.7 Benchmarking - Conditional GAN

To benchmark our GAN results we looked to replicate an experiment completed by Nash (2017). This author used a conditional GAN with the same dataset and scaled and standardised their data. Further they applied a log transformation of the *amount* variable. Firstly we trained non-standardised data using 3 cluster data as defined in section 4.1.1. Observations associated clusters were one-hot encoded to create two binary identifiers. A condition having both at zero value indicated default cluster association. Further, we conditioned our training using 'Peak Time' identifier. Adversarial training was completed using 4000 epochs. The minimum accuracy achieved was 0.9797. Figure 4.13 shows a 10-point rolling average across epochs up to 2000. XGBoost accuracy was minimised close to 700 training epochs.

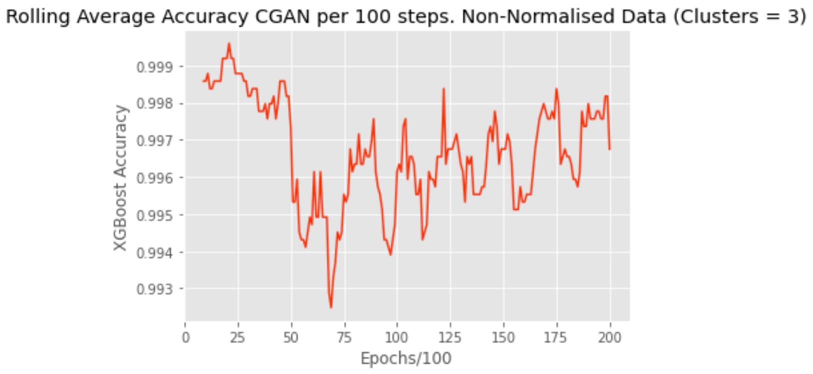


Figure 4.13: XGBoost Accuracy of cGAN using 3 Clusters and Non-Standardised data

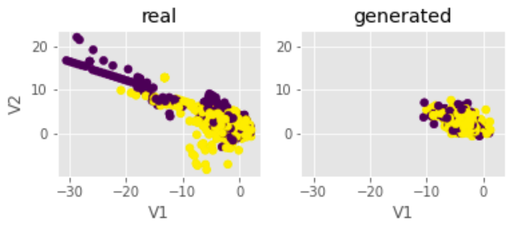


Figure 4.14: Real vs Generated Data at 700 epochs, cGAN Non-Normalised dataset (V1 and V2)

Secondly, we trained our GAN on standardised data. We employed the use of 2 clusters. Observations associated clusters were encompassed by a single categorical variable with two categories (cluster 1 and cluster 2). We used the time variable in this training and omitted *'Peak Time'*. Over 4000 epochs the minimum accuracy achieved was 0.7561. Figure 4.15 shows shows a 10-point rolling average across epochs up to 4000. XGBoost accuracy was minimised close to 2000 training epochs, visually represented in figure 4.16

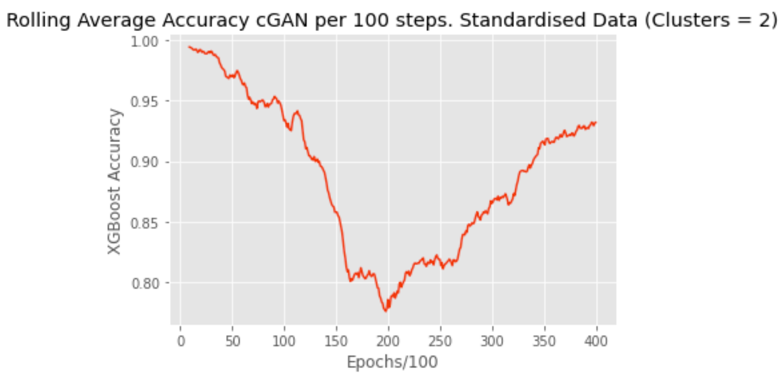


Figure 4.15: XGBoost accuracy with cGAN using 2 clusters and standardised data

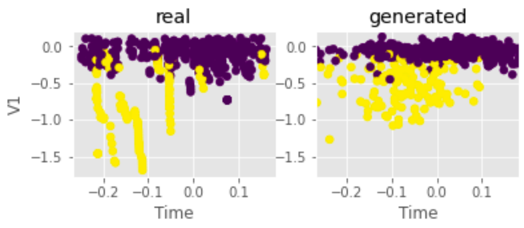


Figure 4.16: Real vs Generated Data at 2000 Epochs, cGAN Non-Normalised dataset (V1 and Time)

4.2 Discussion

4.2.1 CTGAN as an oversampling technique

The experiments give evidence against the null hypothesis, showing that CTGAN regularly ranked low in our upsampling framework across all classifiers. Further, this provides empirical results to give evidence if CTGAN fulfils its purpose on the empirical performance of CTGAN. CTGAN was developed to outperform vanilla GAN and its extension specifically in creating tabular data. It conditionally considers continuous and discrete columns separately. However, our results showed WGAN using a gradient penalty outperforms CTGAN in the task, even though WGAN was built to handle data generation for image processing. This may imply that either GANs do not need to be conditional to their data's structure or CTGAN requires further development.

4.2.2 Evidence of cluster information

To account for confounding factors we replicated our CTGAN 10 times per epoch using both clustered and non-clustered information. We determined that the experiments which used cluster information resulted in significantly higher accuracy. Higher accuracy is synonymous with worse synthetic data. This suggests that the addition of cluster information may not be relevant to data generation. Given the limitations of the dataset, we did not have sufficient information on the individual fraud attacks. There was no way to evaluate the clusters using domain knowledge, for example, to understand if the cluster groups related to similar fraud attacks.

4.2.3 The impact on boosting vs bagging

XGBoost is regarded as a well-performing classifier across supervised classification tasks. We observed that both traditional and GAN upsampling methods did not have a significant impact on performance. Traditional oversampling methods showed a decline in performance as we increased the imbalance ratio above .015. Looking into GAN methods we did not see this occur, with performance relatively stable between .80 and .83. We note that GANS exhibit more textitgeneral synthetic closer to a Gaussian shape compared to traditional methods. This is a potential reason why they may perform better with our gradient boosted classifier, yet further investigation is needed. Oversampling using both GAN and traditional methods showed a significant improvement when using random forest. Although XGBoost achieved the best overal score with WGAN (F1 = .8234, IR = .0074) we saw Random Forest improve to a comparable best score using SMOTE (F1 = .8161, IR = .0084) compared to using no oversampling (F1 = .7694). This highlights the inherent value oversampling can have for bagging techniques.

4.2.4 Decision Tree - how synthetic data is changing our the classifier?

We observed that oversampling the minority class resulted in a change in how our tree classifier creates optimal decision nodes. The most important features remained the same (V17, V14) however their order of importance changed. Decision trees suffer from high variance which could be the main contributing factor. However, this may imply that the addition of synthetic data leads to improved decision boundaries leading to better-performing classifiers.

4.2.5 Distributional Changes

Across classifiers we found WGAN to be the superior synthetic data generation method over CTGAN. Investigating distributional changes, WGAN created a more normal distribution. Traditional methods directly copied the actual data distributional shape. It is interesting to see that the inclusion of synthetic data that is distributionally different still attributed to an improvement in the performance of our classifier.

Our WGAN training had the best mean rank across all classifiers and imbalance ratios. This result was significant. SMOTE outperformed CTGAN, however, CTGAN had a higher mean rank than ADASYN. We hence do not have evidence to reject our research hypothesis. Yet given WGAN as the highest-ranked oversampling method suggests that GAN architectures can outperform traditional methods. Considering WGAN and GAN architectures are initialised from a Gaussian noise vector we are creating synthetic data which is different to the real fraud data.

Chapter 5

Conclusion

Can generative adversarial networks help us fight financial fraud? By completing our research objectives we show that GAN methods can outperform traditional methods to help supervised classifiers performance in classifying fraud. Yet, this depends on the classifier type, the GAN framework and the amount of synthetic data added to the dataset.

The project's main objective was to gather evidence towards our hypothesis (section 3.1). Our hypothesis stated CTGAN as an oversampling method will have significantly better results than traditional methods. We gathered evidence for this by achieving the secondary objective of designing an experimental framework. The results from our experimental framework showed that CTGAN ranked significantly lower compared to WGAN and SMOTE. This provides evidence against our null hypothesis. The completion of this research objective gives insight into the wider scope of the project as we saw WGAN rank highest on average. This contributes to the greater literature that GAN technology can outperform traditional methods. WGAN is potentially more stable than CTGAN implying that the mechanism of optimisation may be more important for synthetic data generation than a framework designed to handle multimodal distributions of discrete and continuous columns.

We completed the objective of engineering a framework for oversampling and completing baseline classifier calculations. Using traditional methods (SMOTE and ADASYN) Random Forest, Logistic Regression and Decision Tree classifiers show a concave relationship between average F1 score and synthetically increasing the dataset's imbalance ratio. This relationship gives evidence toward our sub research question A, *"Does an optimal imbalance ratio threshold exist when adding synthetic data to maximise the performance of a classifier for the detection of financial transactions?"*. Experiment results suggest that there does exist an optimal imbalance threshold for these classifiers. The same was not observed for GANs. Further, we were able to reach a very similar optimal average performance by oversampling using Random Forest compared to XGBoost. This suggests oversampling is well suited for bagging ensemble methods. Our evaluation framework using XGBoost showed little performance improvement across both traditional and GAN frameworks, although we did achieve better than baseline results with WGAN. This indicates oversampling may be less suited for boosting frameworks.

The completion of cluster analysis and GAN training research objectives showed that the inclusion of cluster information attributed to inferior synthetic data generation using CTGAN. We accounted for confounding factors by replicating training 10 times per epoch selection. The significantly lower accuracy for our GANs using no cluster information suggests cluster information may not be useful in aiding synthetic data creation using GANs. We attribute this as evidence for research sub-question B *"Will the use of cluster information aid the creation of synthetic fraudulent data?"*

5.1 Contributions and impact

- We provide evidence that WGAN using a gradient penalty significantly ranks higher than SMOTE and ADASYN for Random Forest. This contributes to the evidence that GAN can be superior in generating synthetic transactional data.
- The addition of clustering information did not show improvement of CTGAN generation. This contributes to the understanding of cluster methods ability to improve synthetic data generation without any domain knowledge.
- We develop an experimental framework for use with SMOTE, ADASYN, CT-

GAN and WGAN. This framework can be easily replicated for other researchers and academics to explore the impact of increasing an imbalanced datasets imbalance ratio on the desired classifiers performance. This is transferrable to multiple domains where data imbalance is an issue e.g., rare disease detection and natural disaster detection.

- We demonstrated that oversampling frameworks do not have significant performance improvements for boosting. We observed significant improvements for bagging techniques, reaching performance close to XGBoost. This shows the inherent value and impact of the framework on the performance of a classifier, considering XGBoost would be considered a state-of-the-art classifier compared to Random Forest.
- For traditional upsampling methods with a Decision Tree, Logistic Regression and Random Forests we showed the existence of a clear concave relationship between classifier performance and oversampling. This contributes to how individuals can optimise the amount of oversampling applied to a given dataset.
- We identified our best performing GAN (WGAN) creating synthetic data with a distribution closer to Gaussian normal compared to SMOTE. This provides empirical evidence that synthetic data whose distribution deviates from the original and is more "general", can still help a learner's performance.
- Benchmarking our conditional GAN performance against a similar approach which used standardised data showed our accuracy considerably higher. This contributes to the idea that scaling and standardising data may improve results as GANs may have less to learn.

5.2 Future Work & recommendations

- There are over 85 SMOTE extensions within the literature. Our experimental design considered only vanilla SMOTE. We would recommend applying our experimental framework across these extensions e.g. Borderline SMOTE (Han et al., 2005), to determine the effect of differing SMOTE techniques on our classifier.

- Our analysis had a small number of duplicate values across the dataset (¡.01%). A recommendation for future work would be to exclude these duplicate values to determine the effect they had on our experimental design.

- Our experiments only considered this technique concerning financial fraud. We recommend replicating the framework on other domains which exhibit imbalance problems e.g. cancer detection.

- When benchmarking our experiments against a similar study, we found that scaling and standardizing showed better performance in data generation. We would recommend replicating our experiments using standardised data to determine if this would have a positive impact on the performance of our classifier.

- GAN networks discriminant network itself is a classifier. If researchers could obtain a dataset with a high number of fraud instances we recommend using transfer learning of the discriminant model to determine how it compares as a classifier.

- Our cluster analysis tested only cluster assignment using k-means, and euclidean distance. There is scope for future testing of different distance metrics e.g. Manhattan distance. Further, one may consider different clustering algorithms.

- We would advise for future work for individuals to evaluate their cluster information with domain knowledge before using them in GAN generation. The main idea behind using clustering for fraudulent data is to indicate similar fraudsters or the same fraudster attempting multiple times. After determining the validity of clusters we recommend comparing GANs with and without cluster information to see if this can aid in improving synthetic data generation.

- Our study showed that WGAN training resulted in a more general distribution compared to traditional methods. We recommend replicating our experiment using higher order of training epochs for GANs. This may contribute to the understanding of the impact of synthetic data distribution on classifier performance.

5.3 Tables

Table 5.1: Decision Tree: Average F1 Results Over Imbalance Ratios

Imbalance Ratio	SMOTE	ADASYN	WGAN	CTGAN
0.0017	0.7802	0.7802	0.7802	0.7802
0.0024	0.7750	0.7770	0.7274	0.7798
0.0034	0.7906	0.7860	0.7465	**0.7864**
0.0044	0.7972	0.7939	0.7883	0.7723
0.0054	0.7854	0.7867	**0.7955**	0.7717
0.0064	0.8031	0.7958	0.7753	0.7794
0.0074	0.7937	0.7959	0.7845	0.7702
0.0084	0.8037	0.7829	0.7664	0.7881
0.0100	**0.8064***	**0.7982**	0.7853	0.7565
0.0150	0.7914	0.7712	0.7695	0.7503
0.0200	0.7719	0.7376	0.7791	0.7512
0.0250	0.7648	0.6912	0.7591	0.7463
0.0300	0.7374	0.6970	0.7438	0.7670

** Best average F1 score across all oversampling methods and imbalance ratios*

Table 5.2: Naive Bayes: Average F1 Results Over Imbalance Ratios

Imbalance Ratio	SMOTE	ADASYN	WGAN	CTGAN
0.0017	**0.1079**	**0.1079**	0.1079	0.1079
0.0024	0.1071	0.0985	0.108	0.1058
0.0034	0.107	0.0932	0.1075	0.1066
0.0044	0.1063	0.0896	**0.1088**	0.1089
0.0054	0.1058	0.0874	0.1067	0.105
0.0064	0.1057	0.0854	0.1081	0.1077
0.0074	0.1061	0.0841	0.1082	**0.1152***
0.0084	0.1057	0.0822	0.1079	0.1073
0.01	0.1053	0.0804	0.1062	0.1075
0.015	0.1043	0.0765	0.1083	0.111
0.02	0.1039	0.0746	0.1061	0.1095
0.025	0.1033	0.0728	0.1065	0.1046
0.03	0.103	0.0717	0.1054	0.1099

** Best average F1 score across all oversampling methods and imbalance ratios*

Table 5.3: Logistic Regression: Average F1 Results Over Imbalance Ratios

Imbalance Ratio	SMOTE	ADASYN	WGAN	CTGAN
0.0017	0.6493	0.6493	0.6493	0.6493
0.0024	0.7045	0.6692	0.6796	0.6602
0.0034	0.7244	0.7225	0.7391	0.6578
0.0044	0.7351	0.73	0.7498	0.6874
0.0054	**0.7489**	0.7478	0.7607	0.6876
0.0064	0.7536	**0.7478**	0.7491	0.6874
0.0074	0.764	0.7426	0.7609	0.6628
0.0084	0.7691	0.7426	0.7641	**0.6951**
0.01	0.7672	0.7337	**0.7821***	0.6933
0.015	0.7692	0.7038	0.7658	0.6906
0.02	0.7656	0.6565	0.7561	0.6375
0.025	0.7564	0.5886	0.7221	0.6635
0.03	0.743	0.5886	0.7305	0.6944

** Best average F1 score across all oversampling methods and imbalance ratios*

Table 5.4: Random Forest: Average F1 Results Over Imbalance Ratios

Imbalance Ratio	SMOTE	ADASYN	WGAN	CTGAN
0.0017	0.7694	0.7694	0.7694	0.7694
0.0024	0.774	0.777	0.784	0.7824
0.0034	0.7953	0.786	0.8001	0.791
0.0044	0.7993	0.7939	0.7998	0.7951
0.0054	0.803	0.7867	0.8074	0.7913
0.0064	0.8065	0.7958	0.8033	0.7934
0.0074	0.8121	0.7959	0.8094	0.7815
0.0084	**0.8161***	0.7829	0.8092	0.7833
0.01	0.8102	**0.7982**	**0.8139**	0.7785
0.015	0.8103	0.7712	0.8055	0.7976
0.02	0.8014	0.7376	0.8083	0.78
0.025	0.7973	0.6912	0.8021	0.7965
0.03	0.8135	0.697	0.7998	**0.8105**

** Best average F1 score across all oversampling methods and imbalance ratios*

Table 5.5: XGBoost: Average F1 Results Over Imbalance Ratios and Upsampling Techniques

Imbalance Ratio	SMOTE	ADASYN	WGAN	CTGAN
0.0017	0.8198	**0.8198**	0.8198	**0.8198**
0.0024	0.8213	0.8149	0.816	0.8096
0.0034	0.8153	0.8148	0.8173	0.8152
0.0044	0.82	0.8079	0.819	0.8089
0.0054	0.8172	0.7994	0.8223	0.8124
0.0064	**0.8219**	0.7914	0.821	0.8155
0.0074	0.8007	0.7964	**0.8234***	0.8104
0.0084	0.8069	0.7816	0.8166	0.8103
0.01	0.8106	0.7703	0.8159	0.8162
0.015	0.7969	0.7429	0.8198	0.8064
0.02	0.7886	0.712	0.8078	0.8056
0.025	0.7836	0.6883	0.8072	0.8037
0.03	0.7734	0.6615	0.8036	0.8078

** Best average F1 score across all oversampling methods and imbalance ratios*

References

AdaBoost. (2009). In S. Z. Li & A. Jain (Eds.), *Encyclopedia of biometrics* (pp. 9–9). Boston, MA: Springer US. doi: 10.1007/978-0-387-73003-5âĆĹ25

Arjovsky, M., Chintala, S., & Bottou, L. (2017, December). Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*.

Ba, H. (2019, July). Improving Detection of Credit Card Fraudulent Transactions using Generative Adversarial Networks. *arXiv:1907.03355 [cs, q-fin]*.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002, June). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. doi: 10.1613/jair.953

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2939672.2939785

Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, *20*(2), 215–232.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., & Bontempi, G. (2014, August). Learned lessons in credit card fraud detection from a practitioner perspective. *Expert Systems with Applications*, *41*(10), 4915–4928. doi: 10.1016/j.eswa.2014.02.026

Dennis Jr, J. E., & Schnabel, R. B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations.* SIAM.

Dimitrakopoulos, G. N., Vrahatis, A. G., Plagianakos, V., & Sgarbas, K. (2018). Pathway analysis using XGBoost classification in biomedical data. In *Proceedings of the 10th hellenic conference on artificial intelligence.* New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3200947.3201029

Douzas, G., & Bacao, F. (2018, January). Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with Applications*, *91*, 464–471. doi: 10.1016/j.eswa.2017.09.030

Fan, S., & Fan, S. (2018, June). *Understanding the mathematics behind Naive Bayes.* https://shuzhanfan.github.io/.

Fernandez, A., Garcia, S., Herrera, F., & Chawla, N. V. (2018, April). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, *61*, 863–905. doi: 10.1613/jair.1.11192

Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019, April). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, *479*, 448–455. doi: 10.1016/j.ins.2017.12.030

Friedman, J., Hastie, T., Tibshirani, R., et al. (2001). *The elements of statistical learning* (Vol. 1). Springer series in statistics New York.

Gangwar, A. K., & Ravi, V. (2019). WiP: Generative Adversarial Network for Oversampling Data in Credit Card Fraud Detection. In D. Garg, N. V. N. Kumar, & R. K. Shyamasundar (Eds.), *Information Systems Security* (pp. 123–134). Cham: Springer International Publishing. doi: 10.1007/978-3-030-36945-3_7

Gao, X., Deng, F., & Yue, X. (2020). Data augmentation in fault diagnosis based on the Wasserstein generative adversarial network with gradient penalty. *Neurocomputing*, *396*, 487–494. doi: 10.1016/j.neucom.2018.10.109

Gee, J., & Button, M. (2019). The financial cost of fraud 2019: The latest data from around the world.

George, D. (2011). *SPSS for windows step by step: A simple study guide and reference, 17.0 update, 10/e.* Pearson Education India.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 2672–2680). Curran Associates, Inc.

Groeneveld, R. A., & Meeden, G. (1984). Measuring Skewness and Kurtosis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, *33*(4), 391–399. doi: 10.2307/2987742

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. (2017). Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*.

Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017, May). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, *73*, 220–239. doi: 10.1016/j.eswa.2016.12.035

Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In D.-S. Huang, X.-P. Zhang, & G.-B. Huang (Eds.), *Advances in Intelligent Computing* (pp. 878–887). Berlin, Heidelberg: Springer. doi: 10.1007/11538059_91

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008, June). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* (pp. 1322–1328). doi: 10.1109/IJCNN.2008.4633969

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. doi: 10.1109/MCSE.2007.55

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Kasa, N., Dahbura, A., Ravoori, C., & Adams, S. (2019). Improving credit card fraud detection by profiling and clustering accounts. In *2019 systems and information engineering design symposium (SIEDS)* (pp. 1–6). doi: 10.1109/SIEDS.2019.8735623

Kassambara, A., & Mundt, F. (2017). *Factoextra: Extract and visualize the results of multivariate data analyses* [Manual].

Keselj, V. (2009). *Speech and Language Processing Daniel Jurafsky and James H. Martin (Stanford University and University of Colorado at Boulder) Pearson Prentice Hall, 2009, xxxi+ 988 pp; hardbound, ISBN 978-0-13-187321-6, $115.00.* MIT Press.

Liu, Y., Zhou, Y., Liu, X., Dong, F., Wang, C., & Wang, Z. (2019, February). Wasserstein GAN-Based Small-Sample Augmentation for New-Generation Artificial Intelligence: A Case Study of Cancer-Staging Data in Biology. *Engineering*, *5*(1), 156–163. doi: 10.1016/j.eng.2018.11.018

Maimon, O. Z., & Rokach, L. (2014). *Data mining with decision trees: Theory and applications* (Vol. 81). World scientific.

Mirza, M., & Osindero, S. (2014, November). Conditional Generative Adversarial Nets. *arXiv:1411.1784 [cs, stat]*.

Mosteller, F., & Tukey, J. W. (1968). Data analysis, including statistics. *Handbook of social psychology*, *2*, 80–203.

Nash, C. Z. (2017, November). *Codyznash/GANs_for_Credit_Card_Data.* https://github.com/codyznash/GANs_for_Credit_Card_Data.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Prati, R. C., Batista, G. E., & Monard, M. C. (2004). Learning with class skews and small disjuncts. In *Brazilian symposium on artificial intelligence* (pp. 296–306).

Randhawa, K., Loo, C. K., Seera, M., Lim, C. P., & Nandi, A. K. (2018). Credit card fraud detection using AdaBoost and majority voting. *IEEE Access*, *6*, 14277–14284. doi: 10.1109/ACCESS.2018.2806420

Reynolds, D. A. (2009). Gaussian mixture models. *Encyclopedia of biometrics*, *741*.

Rish, I., et al. (2001). An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, pp. 41–46).

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., & Chen, X. (2016). Improved Techniques for Training GANs. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 29* (pp. 2234–2242). Curran Associates, Inc.

Shapiro, S. S., & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, *52*(3/4), 591–611.

Shen, A., Tong, R., & Deng, Y. (2007). Application of classification models on credit card fraud detection. In *2007 international conference on service systems and service management* (pp. 1–4). doi: 10.1109/ICSSSM.2007.4280163

Sisodia, D. S., Reddy, N. K., & Bhandari, S. (2017, September). Performance evaluation of class balancing techniques for credit card fraud detection. In *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)* (pp. 2747–2752). doi: 10.1109/ICPCSI.2017.8392219

Sohony, I., Pratap, R., & Nambiar, U. (2018). Ensemble learning for credit card fraud detection. In *Proceedings of the ACM india joint international conference on data science and management of data* (pp. 289–294). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3152494.3156815

Tahir, M. A., Kittler, J., Mikolajczyk, K., & Yan, F. (2009). A multiple expert approach to the class imbalance problem using inverse random under sampling. In *International workshop on multiple classifier systems* (pp. 82–91).

Tanaka, F. H. K. d. S., & Aranha, C. (2019, April). Data Augmentation Using GANs. *arXiv:1904.09135 [cs, stat]*.

Thorndike, R. L. (1953, December). Who belongs in the family? *Psychometrika*, *18*(4), 267–276. doi: 10.1007/BF02289263

Wang, Q., Zhou, X., Wang, C., Liu, Z., Huang, J., Zhou, Y., ... Cheng, J. (2019). WGAN-Based Synthetic Minority Over-Sampling Technique: Improving Semantic Fine-Grained Classification for Lung Nodules in CT Images. *IEEE Access*, *7*, 18450–18463. doi: 10.1109/ACCESS.2019.2896409

Wickham, H. (2016). *Ggplot2: Elegant graphics for data analysis.* Springer-Verlag New York.

Wickham, H., François, R., Henry, L., & Müller, K. (2018). *Dplyr: A grammar of data manipulation* [Manual].

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). Modeling tabular data using conditional GAN. In *Advances in neural information processing systems.*

Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018). Random forest for credit card fraud detection. In *2018 IEEE 15th international conference on networking, sensing and control (ICNSC)* (pp. 1–6). doi: 10.1109/ICNSC.2018.8361343

Zhou, Z.-H., & Liu, X.-Y. (2010, July). ON MULTI-CLASS COST-SENSITIVE LEARNING. *Computational Intelligence*, *26*(3), 232–257. doi: 10.1111/j.1467-8640.2010.00358.x

Zhu, J., Yang, G., & Lio, P. (2019, April). How Can We Make Gan Perform Better in Single Medical Image Super-Resolution? A Lesion Focused Multi-Scale Approach. In

2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019) (pp. 1669–1673). doi: 10.1109/ISBI.2019.8759517

Appendix A

Additional content

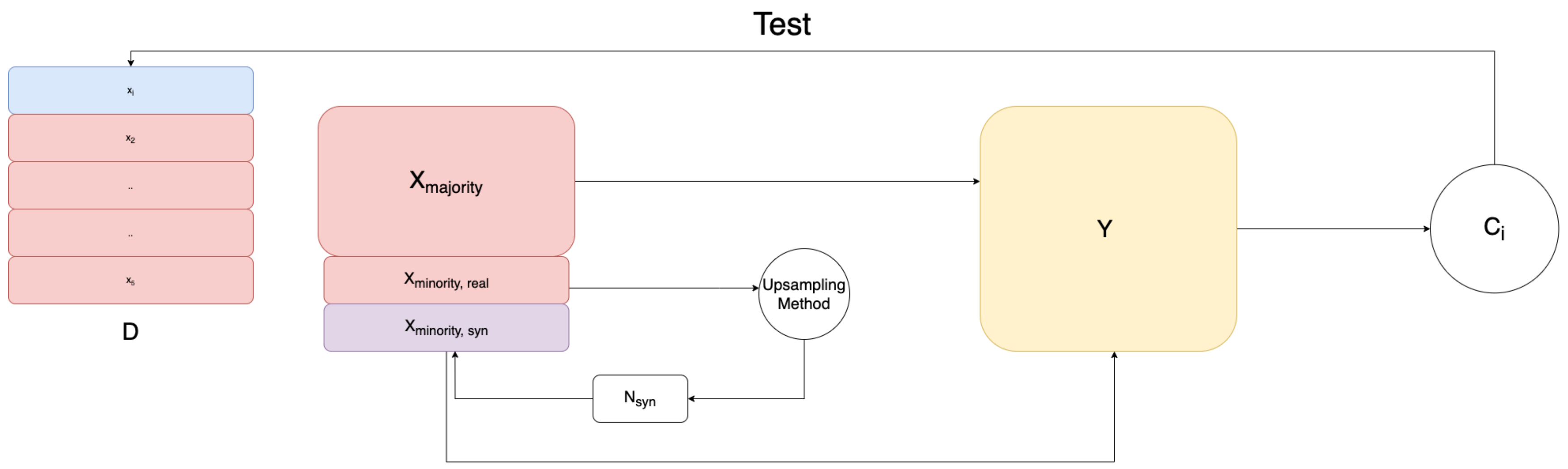


Figure A.1: Upsampling Framework Using 5-Fold Cross Validation

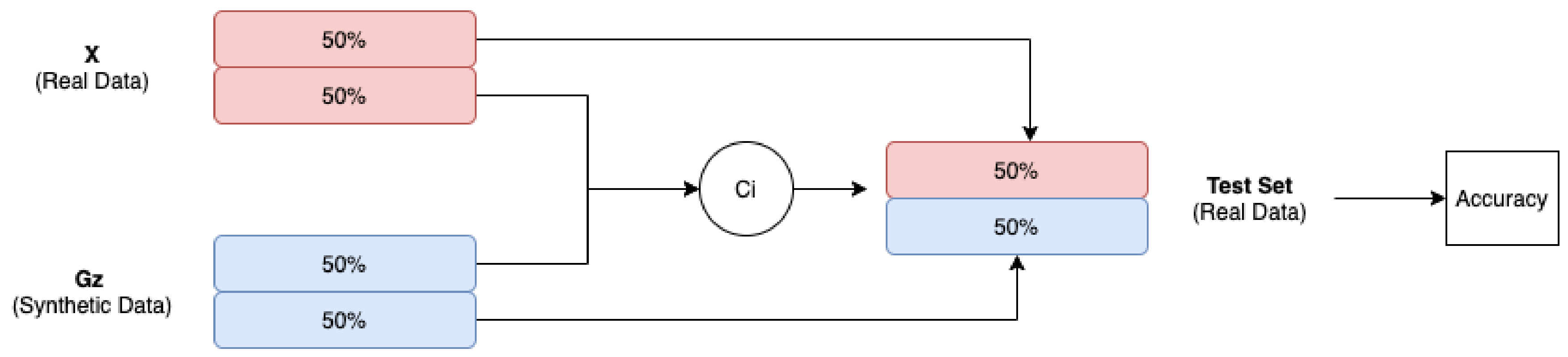


Figure A.2: GAN Testing Framework