EFFECTIVE DETECTION OF PURCHASING INTENTION FOR ONLINE SHOPPING

KANG SHU YI

UNIVERSITI TUNKU ABDUL RAHMAN

EFFECTIVE DETECTION OF PURCHASING INTENTION FOR ONLINE SHOPPING

KANG SHU YI

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

April 2023

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	
Name	:	KANG SHU YI
ID No.	:	1903794
Date	:	28/04/2023

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"EFFECTIVE DETECTION OF PURCHASING INTENTION FOR ONLINE SHOPPING"** was prepared by **KANG SHU YI** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	kckhor
Supervisor	:	Khor Kok Chin
Date	:	18/5/2023

Signature	:	mehao
Co-Supervisor	:	Hoo Meei Hao
Date	:	18/5/2023

ACKNOWLEDGEMENTS

I would like to express my gratitude to my research supervisor, Dr. Khor Kok Chin, for providing me with invaluable advice, input, and encouragement throughout the duration of this research endeavour. His knowledge, wisdom, and constructive feedback were essential in guiding the direction of my research and improving the standard of my work.

I would like to thank Dr. Hoo Meei Hao, my co-supervisor, for her support and encouragement throughout this research endeavour. I would also like to express gratitude to Professor Masaomi Kimura for his suggestions and contributions to enhancing the research's quality. I would also like to acknowledge Dr. Wong Whee Yen for the sincere advice and inspiration she provided from the very beginning of the research.

Throughout the duration of this research endeavour, my family and friends have provided me with support, and encouragement. Their confidence in my abilities and constant encouragement has provided me with fortitude and motivation. I appreciate their understanding and tolerance as I balanced between the responsibilities of this research endeavour with other commitments.

ABSTRACT

The main issue with the below expectations in detecting purchasing intention is caused by the unbalanced data set and its overlapping class problem. To identify a sampling method that best improves the detection rate, this project performed four categories of sampling experiments, resulting in 2,011 experiments in total. To improve the detection results, a hybrid of undersampling and oversampling was applied to reduce and increase the size of the majority and minority classes of the unbalanced data set used in this project, respectively. Undersampling rates from 10% to 80%, and oversampling rates from 10% to 90% are used in combinations to achieve effective detections for the class "Buy", which is the minority in the data set. Random undersampling and five variants of Synthetic Minority Oversampling Techniques (SMOTE): Standard SMOTE, ADASYN, ANS, Borderline SMOTE, and SVM SMOTE, were utilised on the data set. Then, the resulting data sets were cross-validated and tested with five classifiers: Decision Tree, Logistic Regression, Naïve Bayes, Random Forest and SVM. The result indicated that applying Random Forest with the random undersampling rate of 80% and oversampling rate (ANS) of 80% yielded the best recall in detecting the majority and minority classes overall.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

$\hat{r}_i$	density distribution
k	number of nearest neighbours
y	the class label
x	the features
n	the number of features
e	base of natural log
L	likelihood function for logistic regression
A	accuracy
P	precision
$R1$	"Buy" class recall
$R0$	"No Buy" class recall
F_1	F1-score

SMOTE	Synthetic Minority Oversampling Technique
ADASYN	Adaptive Synthetic Sampling
ANS	Adaptive Neighbour Synthetic Sampling
B-SMOTE	Borderline Synthetic Minority Oversampling Technique
SVM-SMOTE	Support Vector Machine Synthetic Minority Oversampling Technique
DT	Decision Tree
RF	Random Forest
SVM	Support Vector Machine
NB	Naïve Bayes
LR	Logistic Regression
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
ROC	Receiver Operating Characteristic

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Globally, the e-commerce market has made a total sale of 4.938 trillion in 2021, a 16% growth from the previous year's sales (Bernhardt, 2022). The e-commerce landscape is extremely competitive; it's a zero-sum game. The increasingly demanding customer base raises the entry barrier for new players to join the e-commerce arena. As for the existing major e-commerce companies, much capital is being injected to compete with other e-commerce rivals (Philips, 2016).

While most companies focus on building brand awareness, some companies compete by creating personalised e-commerce experiences so that the services or products accommodate the needs of users (Philips, 2016). This is where machine learning comes into place. Many companies invest in a salesperson-like behavioural prediction system. These systems collect and analyse users' behavioural data to extract the pattern of consumption/purchasing by users (Sakar et al., 2019).

However, their efforts do not equate to low conversion rates. The average e-commerce conversion rate was 1.75% in June 2022 (IRP Commerce, 2022). This is because of the prediction system's inability to effectively interpret user behaviours due to the rare class problem. The rare class problem occurs when the prediction system cannot effectively predict buyers with purchasing intention. Two core factors are causing the rare class problem: unbalanced data sets and overlapping problems. These two factors are common in the machine learning field as they significantly reduce the effectiveness of conventional machine learning algorithms. The first factor exists whenever there is an unequal distribution of the data training set. In this case, the data set consists of the majority of samples with low-purchase intention rather than the opposite. The second factor exists when the majority and minority samples overlap in the data space.

1.2 Importance of the Project

This project strives to benefit e-commerce companies by producing predictions with high accuracy when it comes to buyer intentions. With a more efficient machine learning model, there will be a high accuracy in identifying buyers with purchasing intentions. Therefore, more users can be converted into buyers by targeting personalised marketing strategies, such as targeted advertisements, discounts, personalised recommendations, and cart notifications. These will, in turn, make the return worth the price tag of the companies' investment.

This project also strives to contribute to the research field by helping researchers to compare the application of several oversampling techniques in solving the rare class problem. Other than contributing to the research field, this project also aspires to assist practitioners in deciding on variants of oversampling methods to apply, especially when dealing with user purchasing intention.

1.3 Problem Statement

1.3.1 Unbalanced Data Set Problem

Major e-commerce companies usually invest huge sums of money into machine learning for customer behaviour analysis, but the results are usually unsatisfactory. The main issue is that in the training data, the portion of low purchasing intention samples surpasses high purchasing intention samples. In this case, the samples with high purchasing intention are the minority class whereas the samples with low purchasing intention are the majority class. This, in turn, causes machine learning algorithms to favour the data samples with low purchasing intention, resulting in low accuracy for predicting data samples with high purchasing intention (Weiss, 2004). There are various methods to overcome this issue: algorithm-level, data-level, and ensemble classification (Kurniawan et al., 2020; Fernandez et al., 2018; Dongre and Snehlata, 2017; Sun et al., 2009). This project focuses on the data-level method, which is pre-processing the unbalanced data set before constructing classification models.

Data-level methods are further broken down into under-sampling, oversampling, and hybrid sampling (Kumar et al., 2021). Under-sampling reduces the number of majority instances, whereas oversampling generates and adds more minority class instances to balance the data set. Under-sampling risks removing the majority of instances which are significant (Prachuabsupakij, 2015). Contrarily, oversampling increases the chance of overfitting. Hybrid sampling combines the oversampling of the minority instances and the under-sampling of the majority instances to balance the data set.

1.3.2 Overlapping Class Problem

The overlapping class issue complicates the process of machine learning. The issue arises when instances of multiple classes occupy the same region in the data space (Vuttipittayamongkol, Elyan, and Petrovski, 2021). Instances that overlap share similar feature values but belong to distinct classes (Figure 1.1). When a data set is unbalanced and overlapping, the decision boundary shifts towards the majority class (Vuttipittayamongkol, Elyan, & Petrovski, 2021). Classifying minority groups is more difficult because separating rules are difficult to implement. This complicates the process of predicting target classes based on features that are extremely similar.

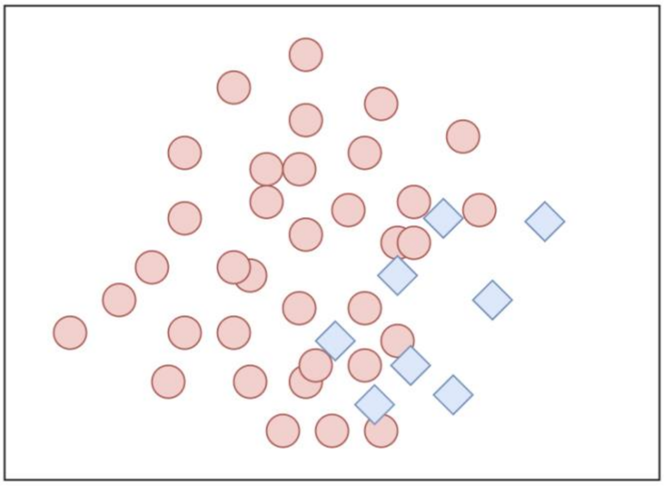


Figure 1.1: An unbalanced data set with an overlapping class problem. The red circles represent the majority class instances, whereas the blue rhombuses represent the minority class instances.

1.4 Aim and Objectives

This project aims to solve the rare class problem of an unbalanced and overlapped data set related to online buyers' purchasing intention.

The objectives of this project are:

- To identify the problems that cause the low detection rate for the online shoppers purchasing intention
- To identify sampling techniques that improve the detection rate for online shoppers purchasing intention

1.5 Proposed Solution

To tackle unbalanced data sets, it is most common to use resampling techniques specifically under-sampling, oversampling, and hybrid sampling. In this project, we focused on undersampling with Random Under Sampling (RUS), oversampling, specifically, five variants of the Synthetic Minority Oversampling Technique (SMOTE) and hybrid sampling. Over the years, scholars have presented many variants of SMOTE. Among the five variants of SMOTE used in this project are Standard SMOTE, Adaptive Synthetic Sampling (ADASYN), Adaptive neighbour synthetic sampling (ANS), B-SMOTE (B-SMOTE) and SVM-SMOTE. Hence, in this project, the effectiveness of five SMOTE variants in predicting purchasing intention using conventional machine learning algorithms was compared.

1.6 Research Approach

Figure 1.2: The flow of the proposed method.

The proposed research approach consists of six phases: Data Pre-processing, Train-Test Set Split, Data Sampling, Model Training and Cross-Validation, Model Testing and Model Evaluation. Data were pre-processed to transform data into a format so that it could be understood by the machine. Then the data set was applied with or without the relevant sampling techniques as shown in Figure 1.2. The five classification algorithms: Decision Tree, Naïve Bayes, Support Vector Machines, Random Forest, and Logistic Regression were used to build a model. The sampled or non-sampled training set was fed to the models, and the models were trained and validated using the k-fold cross-validation method. The performance of the model was evaluated using the test set. The performance of the models was evaluated based on the following evaluation metrics: Accuracy, Recall, and F1 score.

1.7 Scope of the Project

This project focused on the unbalanced and overlapped data set that exists in the data set related to online buyers' purchasing intention. The project utilises RUS as the undersampling technique. The project covers five variants of SMOTE:

- Synthetic Minority Oversampling Technique (SMOTE)
- Adaptive Synthetic Sampling (ADASYN)
- Adaptive neighbour synthetic sampling (ANS)
- Borderline-SMOTE (B-SMOTE)
- SVM-SMOTE.

The project also aims to compare the performances of the classifiers with undersampling, oversampling, hybrid sampling or without any sampling.

- Decision Tree
- Naïve Bayes
- SVM
- Random Forest
- Logistic Regression

Python is the programming language for this project. This project uses a range of libraries and tools, including pandas, NumPy, Matplotlib, Scikit-learn, Imbalanced-learn, smote_variants and seaborn.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In many cases, the rare class in the data set often carries more information. Being the minority in the data set and without performing any balancing techniques, the minority class is often ignored. Examples of cases in which the rare class is essential are cancer detection, fraud detection, bankruptcy detection, etc. The presence of overlapping instances of different classes further complicates the data mining process. When both unbalanced and overlapping data problems are present in the data set, the decision boundary tends to favour the majority class and ignore the minority class.

2.2 E-Commerce

E-commerce is the trading of goods or services on an internet vendor's website (Jain, Malviya and Arya, 2021). There are six types of business models in eCommerce: Business-to-Business (B2B), Business-to-Consumer (B2C), Consumer-to-Consumer (C2C), Consumer-to-Business (C2B), Business-to-Administration (B2A), Consumer-to-Administration (C2A). Huang et al. (2018) state that consumers are less likely to abandon their carts if they are satisfied with the choice process, even though they are hesitant at the point of checkout. Bell et al. (2020) explored the possible factors of cart abandonment categorized into motivational and emotional factors. The motivational factors mainly consist of the lack of primary motivation to buy, and the lack of external motivation, whereas emotional factors are customer irritation and disappointment, security fears, trust, and brand loyalty. Grouping and analysing components of user behaviour help to map buyers' intents. Customer journeys that span multiple sessions and sites over a long period consist of multiple intents map labels better to buyers' intents (Tsagkias et al., 2020).

The e-commerce industry has been steadily growing since it began, but the breakout of the pandemic accelerated it. Consumers who were already shopping online increased their spending, while some late adopters were prompted to learn how to shop online (Kim, 2020).

2.3 Rare Class Problem

The rare class simply indicates that the instances of that class occupy a much lesser composition in the data set compared to other classes. The recall and precision values for the minority class are significantly lower than the majority class. According to Weiss (2004), many practitioners have observed that the recall for the minority class is usually 0, which means no classification rules are computed for the minority class. Weiss (2004) has identified a list of problems associated with rarity, including inappropriate evaluation metrics, absolute rarity, relative rarity, data fragmentation, inappropriate inductive bias, and noise. In an unbalanced data set, instances of the minority class have a higher tendency to be misclassified since they are often ignored by the model (Sun et al., 2009). The rare class problem has garnered wide attention from scholars mainly due to it being a prominent issue in important data sets across domains and the inadequacy of many popular algorithms to overcome it (Sun et al., 2009). Research related to rare class problems usually surrounds three aspects: 1. The nature of the rare class problem, 2. The possible strategies to overcome the rare class problem as well as 3. The best evaluation metrics for the performance of the model.

2.3.1 Nature of the Problem

Studies have presented that unbalanced data sets are not the only issue influencing the performance of models. Among other factors are small sample size, class separability, and within-class concepts. Weiss and Provost (2003) have shown that balanced data sets tend to perform better than unbalanced data sets. However, to what degree of the unbalanced data set does it take to affect the performance of classification algorithms is yet to be determined. As for sample size, according to Japkowicz and Stephen (2002), the larger the data set, the more information on the minority class can be obtained. This will be especially helpful in distinguishing the instances between the two target classes. Though an unbalanced data set may influence the performance of models, that is not always the case when there is class overlapping in some feature space at different levels (Prati, Batista & Monard, 2004). Unbalanced class and class overlapping tend to occur together as misclassification usually occurs near the class boundaries where the overlap of classes happens (Kotsiantis et al., 2005). Within-class unbalance worsens the performance in two aspects, it increases the learning concept complexity, and it is usually implicit (Sun et al., 2009).

2.3.2 Possible strategies to overcome the rare class problem

The discussion of strategies covers two categories: the machine learning algorithm and the pre-processing techniques, where oversampling will be discussed in depth in the upcoming sections. According to Sun et al. (2009), SVM is reported to be less impacted by the unbalance data set problem. In contrast, Akbani, Kwek and Jakowicz (2004) and Wu & Chang (2003) find out otherwise. The authors mentioned that when the skewness of the unbalance data set is too extreme, SVM can be ineffective in determining the class boundary. Most research on unbalanced data set problems focuses on the decision tree algorithm, C4.5 (Sun et al., 2009).

Relying on the standard machine learning algorithms is insufficient to overcome the unbalanced data set problem. To improve the performance of the algorithms, rebalancing of the data set needs to be done, and there are several techniques to achieve this. In general, these techniques are known as resampling techniques and are categorised into two classifications, basic sampling methods, and advanced sampling methods. Under-sampling and oversampling are examples of the basic sampling method. Advanced sampling methods, on the other hand, may combine under-sampling and oversampling or apply intelligence to the basic methods. Oversampling will be further discussed in the next section.

2.4 Random Undersampling

Random Undersampling randomly eliminates data instances from the majority class in a data set based on a predefined undersampling rate. Due to its simplicity, random undersampling is a popular undersampling technique. Most often, researchers opt for Random Undersampling of the majority data instances to mitigate the class unbalance problem, reduce computational costs, and reduce training time. The undersampling rate may be increased to a point to achieve the desired class distribution. Zuech et al. (2021) investigated the correlation between classification performance in detecting web attacks and the application of eight random undersampling ratios and seven distinct classifiers. Four ensemble classifiers—Random Forest, CatBoost (CB), Light Gradient Boosted Machine (LightGBM), and XGBoost—generated better AUC scores when random undersampling was applied. Hasanin et al. (2019) compared the effects of six different data-level sampling techniques: Random Undersampling, Random

Oversampling, Standard SMOTE, SMOTE-borderline1, SMOTE-borderline2 and ADASYN on the class unbalance problem in Big Data. The authors concluded that RUS is the best data-level sampler among the six sampling techniques because classifiers with the highest AUC and GM were generated when RUS was used in the SlowlorisBig case study. Xiao et al. (2021) investigated the effect of resampling methods and classification models on credit-scoring issues involving an unbalanced data set. According to the authors, there was no statistically significant difference in TPR, F-measure, G-mean and AUC scores between using RUS versus SMOTE + ENN at the 95% confidence level. Tantithamthavorn et al. (2020) investigated the effect of data-sampling methods on the efficacy of defect prediction models. RUS enhances Recall by the greatest margin among the five compared sampling techniques: Random Oversampling (ROS), Random Undersampling (RUS), SMOTE, and the Bootstrap Random Oversampling Examples Technique (ROSE).

However, RUS has a fatal flaw in removing crucial data from the data set (Shamsudin et al., 2020; Devi et al., 2020). RUS is incapable of selecting a data point according to its significance (Xiao et al., 2021). Consequently, this may exacerbate the classification procedure, as the decision boundary between the minority and majority classes may become ambiguous. As a result, researchers favour oversampling over undersampling most of the time (Ali et al., 2019). Koziarski (2021) contrasted the efficacy of RUS and Combined Synthetic Oversampling and Undersampling Technique (CSMOUTE) on unbalanced binary data sets from the KEEL repository. Combining SMOTE and SMUTE, the proposed CSMOUTE method is a hybrid sampling technique. In terms of the F-measure, CSMOUTE outperformed RUS statistically.

2.4.1 Advantage of Combining RUS with Oversampling Techniques

Shamsudin et al. (2020) analysed the effect of combining RUS with five oversampling techniques (SMOTE, ADASYN, Borderline, SVM-SMOTE, and ROS) on a data set of fraudulent credit card transactions using the Random Forest classifier. The study's findings indicate that applying Random Forest combined oversampling techniques and RUS enhances the Precision, Recall, and F1-score compared to using RUS or oversampling techniques alone. For handling highly unbalanced data classes in Big

Data, Johnson and Khoshgoftaar (2020) suggest the ROS-RUS hybrid sampling method. The authors compared the effects of ROS, RUS, and the hybrid ROS-RUS with Deep Learning on data with a significant unbalance. Based on the AUC scores obtained, the hybrid ROS-RUS performed as good as or better than ROS or RUS individually in their investigation. Mahadevan and Arock (2020) enhanced ensemble learning by combining RUS and SMOTE in a hybrid sampling technique. The hybrid sampling technique proposed managed to avoid introducing bias and losing crucial information from the majority class. Compared to the other models, the suggested approach attained the highest G-mean, F1-score, and ROC_AUC scores. Lee and Kim (2020) compared the impact of data sampling on the prediction of nuclear receptor toxicity. The results of the study indicate that hybrid sampling (RUS + ROS) enhanced the Accuracy, ROC-AUC score, and Recall of the SCFP model in comparison to when sampling techniques were not utilised.

2.5 Oversampling Techniques

Oversampling is the technique where synthetic data of the minority class is created to reduce the skewness of unbalance of the data set. The most basic oversampling technique is random oversampling. In random oversampling, no heuristics are applied. This makes them simple to implement and fast in executing, hence desirable in huge and complex data sets. To perform random oversampling, instances of the minority class are randomly chosen with replacement, duplicated, and added to the training set. However, a trade-off is the model is likely to overfit since all instances of the minority class are replicated as it is (Peng et al., 2019; Chawla et al., 2002). The random sampling results in a substantial bias towards the majority class, whereas the effect of bias is less evident in SMOTE (Mahadevan and Arock, 2020).

2.5.1 SMOTE

In SMOTE, the minority class is oversampled by obtaining samples from the minority class and generating synthetic instances along the line of its k-nearest neighbours (Chawla et al., 2002). To generate the synthetic samples, a number between 0 and 1 is multiplied by the difference between the feature vector and its adjacent neighbour. Consequently, a random point along the line between the two features is chosen (Figure 2.1). In the end, the region within the decision boundary of the minority class turns more general. Compared to random oversampling, SMOTE mitigates the overfitting problem since this technique generates new samples instead of duplicating existing ones (Kotsiantis et al., 2005). SMOTE establishes a correlation between a selected number of instances, but not between variables (Mahadevan and Arock, 2020). The newly generated synthetic instances influence the classifying algorithm to establish a larger and more general decision region rather than smaller and more specific decision regions. The classifying algorithm can generalize better since minority class instances have occupied more general regions. Koziarski (2020) stated that SMOTE does not account for the distribution of the majority instances, causing newly generated minority instances to overlap with the cluster of majority instances. The SMOTE algorithm has the algorithmic complexity of O(nlogn)fe. Arafat et al. (2019) stated that SMOTE performs better in most data sets than ensemble classifiers.

```
Function SMOTE
    if N < 100
        then Randomise the T minority class samples
        T ← (N/100) * T
        N ←100
    end
    N ← (N/100)
    k ← Number of nearest neighbors
    nummattrs ←number of attributes
    Sample[][]: array for original minority class samples
    newindex ← 0
    Synthetic[ ][ ]: array for synthetic samples
    for i ←1 to T
        knn_array ← indices of k nearest neighbours of i
        Populate(N, i, knn_array)
    end
    Function Populate(N, i, knn_array)
    while N≠0
        nn ← random number between 1 to k
        for attr ← 1 to numattrs
        dif ← Sample[nnarray[nn]][attr] − Sample[i][attr]
        gap ← random number between 0 and 1
        Synthetic[newindex][attr] ←Sample[i][attr] + gap ∗
      dif
      end
      newindex += 1
      N ← N−1
    end
    return
```

Algorithm 2.1: The algorithm of Synthetic Minority Oversampling Technique (SMOTE) (Chawla, 2002; Sridhar and Sanagavarapu, 2021).

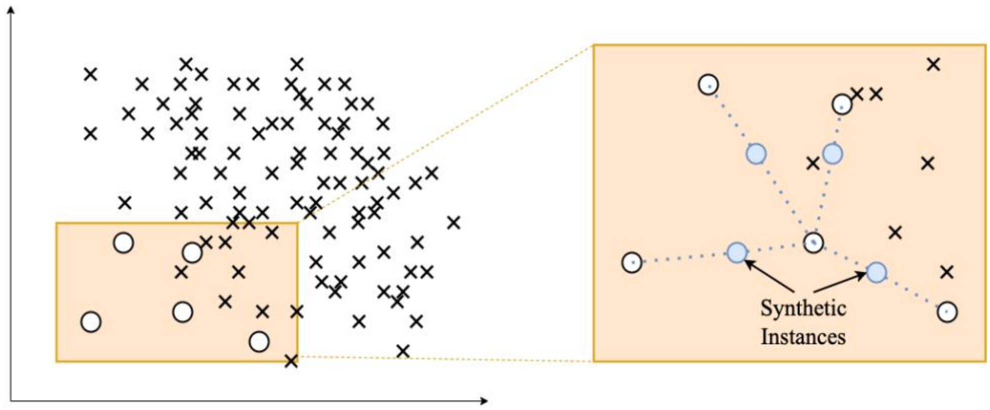


Figure 2.1: The Generation of Synthetic Instances using SMOTE.

2.5.2 ADASYN

Adaptive Synthetic Sampling (ADASYN) focuses on generating instances for minority instances that are more difficult to learn rather than those that are easier to learn. Based on the density distribution, $\hat{r}_i$, ADASYN automatically determines the number of synthetic instances required for each minority class. R is the measurement of the distribution of weights for each minority class instance based on their degree of difficulty in learning. In addition to balancing the data distribution, ADASYN also forces the machine learning algorithm to learn difficult-to-learn samples by adaptively shifting the decision boundary to emphasise those samples (He et al., 2008). The ADASYN algorithm has the algorithmic complexity of *O(nlogn)*. ADASYN approach shares similarities SMOTEBoost and DataBoost-IM except for the way the distribution function is updated. To update the distribution function, SMOTEBoost and DataBoost-IM use the evaluation of hypothesis performance, whereas ADSYN adaptively updates the distribution function according to the data distribution characteristics. As a result, ADASYN shows better efficiency than SMOTEBoost and DataBoost-IM since no hypothesis evaluation is required for generating synthetic instances (He et al., 2008).

```
Function ADASYN
    N ← Amount of oversampling
    G ← (Smax/Smin) * N
    k ← Number of nearest neighbors
    Sample[][]: array for original minority class samples
    newindex ← 0
    Synthetic[ ][ ]: array for synthetic samples
    for i ←0 to Smin
        k_array ← Find k nearest neighbours of i
        Hi ← Number of Majority class instances in k_array
        ri ← Hi / k
        r̂i ¬ r_i / (\sum_{i=1}^{S_{min}} r_i)
        gi ←r̂I × G
            for j ← 1 to gi
                Synthetic[newindex] ←Sample[i]
                newindex += 1
            end
        end
    return
```

Algorithm 2.2: The algorithm of Adaptive Synthetic Sampling (ADASYN) (Sridhar and Sanagavarapu, 2021).

2.5.3 B-SMOTE

Borderline-SMOTE (B-SMOTE) focuses on minority instances on the borderline and nearby since there is a higher possibility of misclassifying them than those away from the borderline (Han et al., 2005); hence, making them more significant for classification. B-SMOTE first identifies the instances of the minority class along the border between the majority and minority classes. New instances are then generated from the identified minority class instances and appended to the original training set. The B-SMOTE algorithm has an algorithmic complexity of $O(n^2)$.

Based on the Figure 2.2, B-SMOTE only emphasises oversampling the borderline samples and their nearby instances.

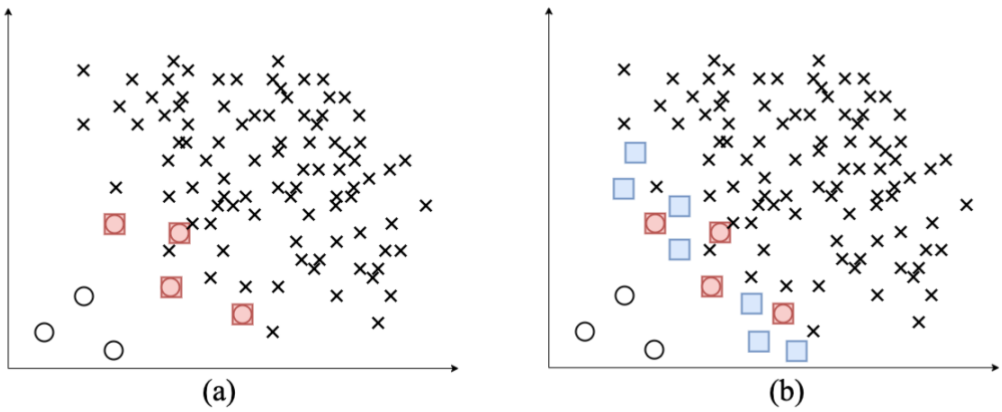


Figure 2.2: Illustration of generating borderline synthetic instances. (a) The borderline minority instances are indicated by circles highlighted by red squares. (b) The borderline synthetic minority instances are indicated by the blue squares.

```
Function BorderlineSMOTE
N ← Amount of oversampling
k ← Number of nearest neighbors
danger[][]: array for minority class samples near/on borderline
for i ← Minority Class Instances
        k_array ← Find k nearest neighbours of i
        Hi ← Number of Majority class instances in k_array
        if Hi = k OR  0 <= Hi < k/2
                continue
        else if k/2 <= Hi < k
                danger.add(i)
        end
end
for j ← danger
        k_array ← Find k nearest neighbours of j
        Populate(N, j, k_array)
end
return
```

Algorithm 2.3: The algorithm of B-SMOTE (Sridhar and Sanagavarapu, 2021).

2.5.4 ANS

The two objectives of Adaptive neighbour synthetic sampling (ANS) are first, to override the relying upon the value of K which is by default, 5 in SMOTE and second, to preserve the significance of minority outcasts without generating synthetic instances. ANS first identifies and excludes the minority outcasts in the data set by using the C-nearest neighbour algorithm (Siriseriwan and Sinapiromsaran, 2017). The minority instances are identified as minority outcasts when all of their C-nearest neighbours are negative. The identified minority outcasts are isolated from the set while the rest of the minority instances are used in generating synthetic instances by using the SMOTE algorithm. For each minority instance, the longest distance between the two nearest neighbours is taken as the radius. Using this radius value, the number of minority instances within the circumference of this radius value is taken as the K value for each minority instance. SMOTE is later performed based on the K value determined, which varies for each minority instance. Regions with a higher density of the minority instances will have a more scattered distribution of the synthetic instances, whereas regions with a higher density of the majority class will have lesser synthetic instances generated (Siriseriwan and Sinapiromsaran, 2017). The minority outliers are later included in a set of majority instances as a sub-data set and build a 1-nearest neighbour model. A small positive region is generated around each outlier, causing any undetermined instances that occur within this region to be classified as members of the minority class regardless of the trained classifier's output.

```
t ← 1;
(Pused, OC, E) = OutcastExtraction(D, P, C)
ε ← maxE
for p_i in Pused
    Np_i ← {p_j in Pused | d(p_i, p_j) < ε }
end
while t < the roundup value of |N|/| Pused | do
    for p_i Pused
            Randomly select np_i from Np_i
            gap ←  a random number between 0 and 1
            p'←  p_i + gap × ( np_i - p_i)
            Add p' into S.
    end
    t += 1
end
Function OutcastExtraction(D, P, C)
    C_max ← 0.25*|D|
    Perform C_max-nearest neighbour of P in D
    fp_i ← first positive nearest neighbour of p_i in P as
    out_border_i ← number of negative neighbours of p_i with
    smaller radius than d(fp_i, p_i)
    for c ←  1 to C_max
            for p_i in P
                    if out_border_i > c
                            p_i is the outcast for this c
                    end
            end
            n_oc_c ← the number of outcast in this c
            if | n_oc_c - n_oc_c-1 | = 0, set C = c
end
OC ← { p_i in P | out_border_i > C }
Pused ← { p_i in P | out_border_i < C }
ε_i in E_Pused ← the distance between p_i in Pused and its nearest
    positive neighbour
return {Pused , OC, E_Pused }
```

Algorithm 2.4: The algorithm of Adaptive neighbour synthetic sampling (ANS) (Siriseriwan and Sinapiromsaran, 2017).

2.5.5 SVM-SMOTE

SVM-SMOTE focuses on generating synthetic minority instances along the decision boundary. (Wang, 2008) has proven that emphasising minority instances along the borderline results in better performance compared to sampling the entire minority class. SVM-SMOTE uses a standard SVMs classifier to approximate the decision boundary between the majority and the minority classes using support vectors. New instances are then randomly generated along the support vectors joining the minority class instances with their neighbours using either interpolation or extrapolation based on the density of the majority instances around the chosen instance. New instances will be generated with the extrapolation technique if the number of majority instances is not more than half of its nearest neighbours to extend the minority class region towards the majority class (Nguyen et al., 2011). However, similar to SMOTE, if the number of majority class instances counts from more than half of its nearest neighbours, the boundary area of the minority class will be merged, except new instances are generated in order of first to the k^{th} nearest neighbour instead of randomizing (Nguyen et al., 2011). The algorithmic complexity of SVM-SMOTE is O(n2).

```
Function SVM-SMOTE
N ← Amount of oversampling
k ← Number of nearest neighbours
danger[][]: array for minority class samples near/on borderline
Synthetic[ ][ ]: array for synthetic samples
p ← random(0, 1)

for i ← 0 to len(sv)
    k_array ← Find k nearest neighbours of sv[i]
    Hi ← Number of Majority class instances in k_array
    if 0 <= Hi < k/2
            for j ← 0 to amount [i]
                    Synthetic[i] ←sv[i] + p × (sv [i] – ksvarr[i][j])
            end
    else if k/2 <= Hi < k
            for j ← 0 to amount [i]
            Synthetic[i] ←sv[i] + p × (ksvarr [i][j]) – sv [i])
            end
    end
end
return
```

Algorithm 2.5: The algorithm of SVM-SMOTE (Sridhar and Sanagavarapu, 2021).

2.5.6 Summary of SMOTE variants

All SMOTE variants are “ordinary” sampling except for ADASYN. “Ordinary” sampling refers to the implementation of the same sampling method in SMOTE where all instances along the line connecting the minority instances to their neighbours belong to the minority class. ADASYN, B-SMOTE and SVM-SMOTE methods share the similarity of implementing the process of identifying minority instances on and along the borderline and generating new instances near them. Both ADASYN and SVM-SMOTE use a supervised classifier in their algorithm. Out of five SMOTE variants, only ANS is a density-based technique.

Table 2.1: Comparison of SMOTE variants (Kovács, 2019).

	Density based	Ordinary sampling	Borderline	Uses classifier
SMOTE		x		
ADASYN			x	x
B-SMOTE		x	x	
ANS	x	x		
SVM-SMOTE		x	x	x

2.6 Classification algorithm

2.6.1 Decision Tree

The Decision Tree is a popular classifier used in vast fields of study. A decision tree generates rules which are used to classify the tuples. The decision tree has a tree-like shape. Each node of a decision tree is the test against a rule, each branch is the result of the test, and each leaf is the target class (Figure 2.3). Each path originating from the root to the leaf represents the classification rules. The deeper the tree branches, the less general the decisions made by the classifier. When the rules become more complex, the fitter the model is.

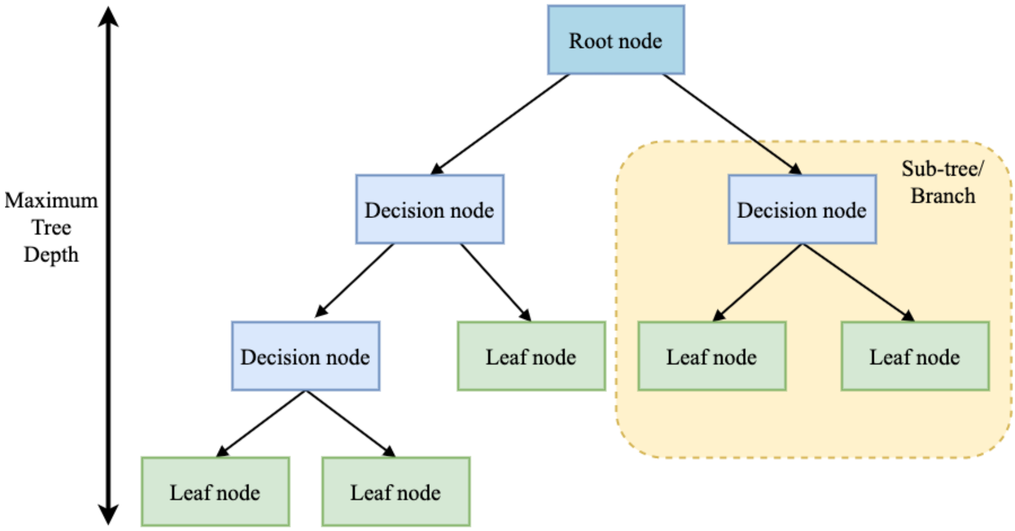


Figure 2.3: Illustration of the Decision Tree logic.

2.6.2 Naïve Bayes

The Naïve Bayes classifier applies the Bayes' theorem to classify tuples. Bayes' Theorem is a formula to calculate the conditional probability of an event occurring provided that another event has occurred. This classifier assumes that each feature contributes equally to the outcome and is independent of the others. The prediction of a class is determined by the probability of the function value corresponding to that class. It calculates the prior probability of each instance feature. The posterior probability for each class is then calculated. Finally, the outcome of the prediction is the class with the highest posterior probability. This classifier is extremely fast compared to more complex algorithms. Below are the equations for Naïve Bayes classification:

$$p(y \mid x_1, \ldots, x_n) \tag{2.1}$$

$$p(y|x) = \frac{p(y)p(x|y)}{p(x)} \tag{2.2}$$

$$p(y) = p(y)\prod_{i=1}^{n} p(x_1|y) \tag{2.3}$$

where

y = the class label

x = the features

n = the number of features

2.6.3 SVM

The support vector machine (SVM) is a form of deep-learning algorithm that groups data instances by mapping them to a high-dimensional feature space. The objective of SVM is to draw a hyperplane that can distinctively distinguish instances of different classes. New instances are predicted to belong to which side of the hyperplane they should be. SVM finds the instances closest to the line and maximizes the margin between the support vectors and the line. When the margin is maximum, then the optimum hyperplane has been found (Figure 2.4).

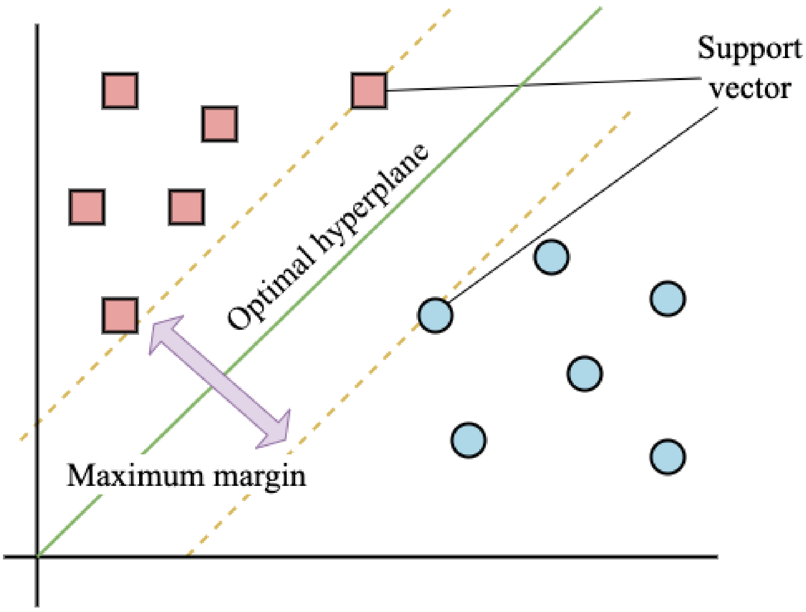


Figure 2.4: The support vector machine mechanism.

2.6.4 Random Forest

As an ensemble learning method, random forest is built from multiple decision trees trained with the bagging method. The method is made up of multiple decision trees built on the same data set. In the random forest model, decision trees run in parallel without interacting with each other (Figure 2.5). The random forest classifier produces a decision based on majority votes by the decision trees. Hence, the random forest classifier tends to generate better results and is a complex model.

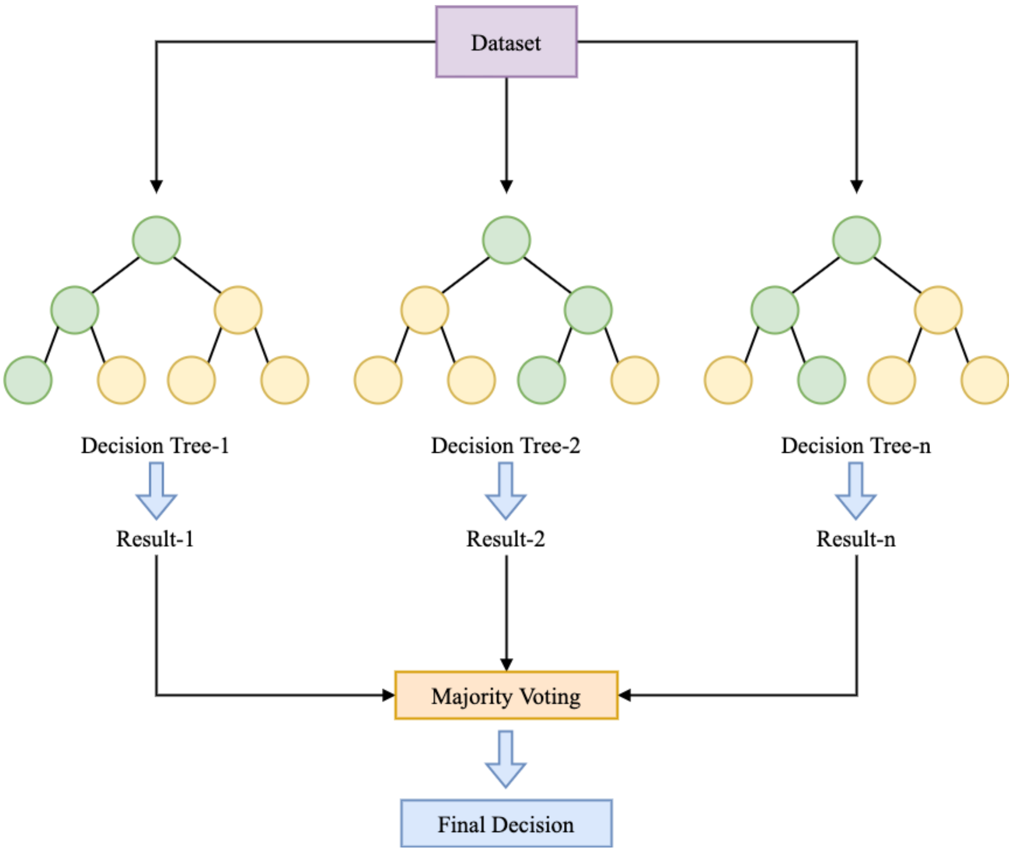


Figure 2.5: The mechanism of the random forest classifier.

2.6.5 Logistic Regression

Based on a set of independent variables, a logistic regression classifier predicts the probability of a binary outcome. Logistic regression shows the correlation between the binary class label and the features which can be nominal, ordinal or ratio. This classifier assesses the correlation between the class label and one or more features by approximating the probability using a logistic function.

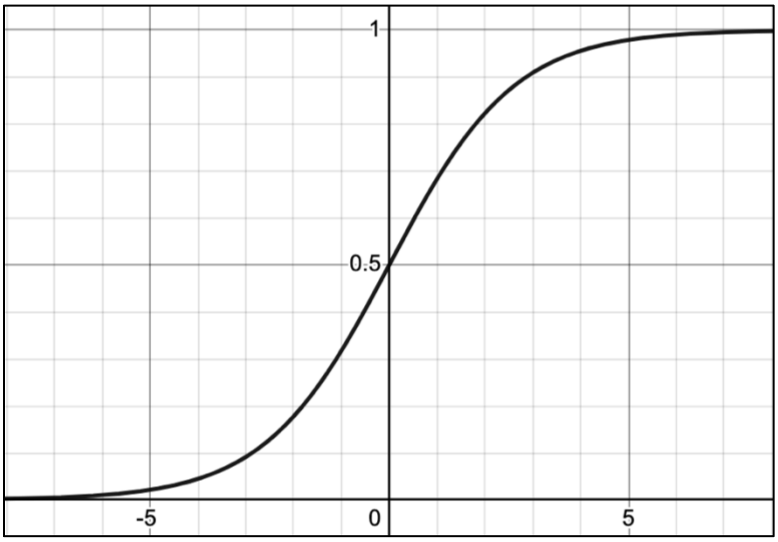


Figure 2.6: The logistic regression curve.

$$logistics\ (p) = \frac{1}{1 + e^{-p}} \tag{2.4}$$

$$p = \beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i \tag{2.5}$$

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_1)^y (1 - p(x_1)^{1-y}) \tag{2.6}$$

where

logistics (p) = an output between 0 and 1 (probability estimate)

p = input to the function (formula prediction)

e = base of natural log

L = likelihood function for logistic regression

2.6.6 Summary of Classification Algorithms

Table 2.2 below summarises the strategy, advantages, and limitations of the five classification algorithms used in this project. The table is self-explanatory.

Table 2.2: Comparison between classification algorithms.

Classification Algorithm	Strategy	Advantage	Limitations
Decision Tree	Classify instances by testing against rules	Requires minimal feature transformation.	Performs poorly on unbalanced data sets.
Naïve Bayes	Predicts based on Bayes' theorem	Fast and highly scalable.	The assumption of dependency makes it unrealistic. Features in data sets are not completely independent of each other.
Support Vector Machine	Generates an optimal hyperplane with maximum margin	Works well on high dimension data sets.	Does not perform well when data set is noisy. Not suitable for large data sets.
Random Forest	Classify based on majority votes of	Better classification result compared	Large number of trees slows

	decision trees in the forest	to simpler algorithms.	down the algorithm.
Logistic Regression	By estimating the probability using a logistic function	Has low variance. (Akkaya and Çolakoğlu, 2019)	Does not perform well when correlated attributes are present. (Akkaya and Çolakoğlu, 2019)

2.7 Related Work

Many researchers were involved in the research of predicting online shoppers' purchasing intention. This section discusses previous relevant studies working on predicting online shoppers' purchasing intention. The following studies are based on the Online Shoppers Purchasing Intention data set from Sakar et al. (2018).

Yap & Khor (2022) applied all three categories of sampling methods, SMOTE for oversampling, random under-sampling, and hybrid sampling to the data set. The authors oversampled from 10% to 150% of the minority class, under-sampled 10% to 80% of the majority class, and applied the same rates for hybrid sampling. Before applying sampling techniques, the authors compared six classification algorithms: K-Nearest Neighbour (KNN), C4.5, Support Vector Machine (SVM), Sequential Minimal Optimization (SMO), Naïve Bayes (NB), and Multilayer Perceptron (MLP). The result shows that the C4.5 algorithm performed with the highest accuracy of 89.6%. However, the NB algorithm (67.6%) has shown a True Positive Rate (TPR) higher than C4.5 (58.7%). After applying sampling methods, the highest accuracy was given by the combination of hybrid sampling and the C4.5 algorithm of 87.0% where the TPR is 84.2% and TNR is 87.5%. Yap & Khor (2022) concluded that performing hybrid sampling gave a better result than SMOTE and under-sampling techniques.

Sakar et al. (2019) compared Random Forest (RF) with C4.5, Support Vector Machine (SVM), and Multilayer Perceptron (MLP) with and without applying the random oversampling technique. MLP showed the highest accuracy of 87.24% and an F1 score of 0.86 followed by RBF SVM with an accuracy of 84.88% and an F1 score of 0.82. Overall, the F1 score has increased by 49% on average after applying the oversampling technique.

Baati and Mohsil (2020) focused on comparing Naïve Bayes (NB), C4.5, and random forest classifiers for the prediction before and after applying the oversampling SMOTE technique. In contrast to the work done by Sakar et al. (2019), the authors selectively used features related to session and user information only. Among the features used are day, operating systems, browser, region, traffic, visitor, weekend, month, and revenue, which are all categorical features except for day. Hence, the author discretized day into five discrete levels to homogenize it as a categorical feature as well. Random forest with oversampling obtained the highest accuracy of 86.78% and an F1-score of 0.60.

Prayogo and Karimah (2021) propose using information gain and correlation feature selection to identify the most significant features and ADASYN as the resampling technique along with random forest as the classifier. Based on the feature selection approach, five features that are highly correlated with the class labels: Page value, Exit rate, Bounce rate, Product related, Product related duration, are selected. In Prayogo and Karimah (2021), the random forest classifier with ADASYN performs better than without ADASYN in the aspect of its accuracy, precision, recall and F1-score. The performance achieved with the proposed approach is an accuracy of 93.78%.

Kek et al. (2021) use z-score standardization to scale the data to result in a mean of 0 and a standard deviation of 1. The authors applied three classifiers to the data set in the study: Logistic Regression, Support Vector Machine and Decision Tree. The authors then choose the best-performing classifier by using the ensemble method. The oversampling technique applied by the authors is SMOTE, where 10,422 synthetic minority instances are generated to balance the data set. The top 10 most important features are ordered by calculating the F-score using the XGBoost model. Among the top 10 selected features are PageValues, Nov, OperatingSystems, VisitorType, TrafficType, Administrative, Region, ProductRelated_Duration, ProductRelated and Weekend. Random forest outperformed the other ensemble methods achieving an accuracy of 94.3%.

Muda et al. (2020) applied SMOTE to overcome the issue of unbalanced data sets. The authors performed a Chi-square test for feature selection. The best-performing model in Muda et al. (2020) is the random forest classifier paired with a 5-fold cross-validation where the accuracy is 88.35% and the AUC value is 80.04%.

Obiedat (2020) compared three classifiers: multilayer perceptron (MLP), decision tree and random forest; and five oversampling techniques: SMOTE, ADASYN, SMOTE-Borderline, SVM-SMOTE and SMOTE-NC. Among the three classifiers, the random forest classifier had the highest performance in accuracy, precision and F1-score, with 89.1%, 69% and 60.7%. The author also states that the most optimal oversampling percentage was 466.26% as it gave the highest F1-score. SMOTE-NC produced the best result for the positive class (91.5%), whereas SVM-SMOTE gave the best result for the negative class (92.3%).

Aside from the online purchase intention data set, some authors used other e-commerce data sets to study the unbalanced distribution of data as well. Esmeli et al. (2021) trained five machine learning models, Decision Tree (DT), Random Forests (RF), Bagging, K-Nearest Neighbour (KNN) and Naive Bayes (NB) on the YooChoose RecSys data set in their study. Comparing under-sampling and SMOTE as the sampling method, the best performance is achieved by applying the under-sampling technique with the decision tree classifier where the AUC value is 97.08%. Mokryn et al. (2019) studied using two data sets, the primary one being the YooChoose RecSys data set and the Zalando data set as the secondary data set. To overcome the unbalanced data set problem in each data set, the authors applied the SMOTE technique. Four classifiers were compared: logistic regression, bagging, NBTree and XGBoost. In general, for both data sets, the bagging classifier performed better than the other three classifiers. The authors also stated that applying SMOTE to the data sets provided significant improvements in the results compared to under-sampling techniques. Geene (2020) uses the Tooso fashion clickstream data set, which consists of 203,958 user sessions with only 4.21% being of the positive class. The author applied random oversampling and random under-sampling with different sampling ratios: 0.1, 0.25, 0.5, 0.75, and 1.0. The author compared five models: Naïve Bayes, Markov chain, gradient boosted machine, autoencoder and Long Short-Term Model. Overall, the LSTM model performed the best when the oversampling ratio was 0.1 where the F1 score achieved was 62.9%, accuracy was 96.2% and the AUC value was 86.4%.

Numerous authors have concluded that the sampling methods contributed to overcoming the unbalanced data set problem, as evidenced by the enhanced evaluation metrics. Therefore, sampling methods are utilised together with classification algorithms in this project as a solution to the unbalanced data set problem that exists in the online purchase intention data set.

Table 2.3: Comparison Between Related Works.

No.	Author (Year)	Title	Sampling Technique	Data Set	Result / Findings	Research methods/ variables
1.	Yap and Khor (2022)	Utilising Sampling Methods to Improve the Prediction on Customers' Buying Intention	• SMOTE • Random under-sampling • Hybrid sampling	UCI Online shoppers' purchasing intention data set 12,330 instances (84.5% negative class, 15.5%	Hybrid Sampling shows the best performance among all three sampling methods (oversampling, undersampling and	**Classification Algorithms:** K-Nearest Neighbour (KNN), Naïve Bayes (NB), C4.5, Support Vector Machine (SVM), Sequential

				positive class)	hybrid sampling). C4.5 algorithm showed the highest accuracy compared to the other six algorithms. Hybrid sampling consumes less computational cost compared to MLP.	Minimal Optimization (SMO) and Multilayer Perceptron (MLP) **Validation:** 10-fold cross-validation
2.	Prayogo and Karimah (2021)	Feature Selection and Adaptive Synthetic Sampling Approach for Optimizing Online Shopper Purchase	ADASYN	UCI Online shoppers' purchasing intention data set 12,330 instances (84.5% negative class, 15.5%	**Best Performance:** ADASYN + Random Forest Accuracy: 93.27% Weighted Average Precision: 93.3% Weighted	**Classifier:** Random Forest

		Intent Prediction		positive class)	Average Recall: 93.3% Weighted Average F1-score: 93.3% **Improvement with ADASYN:** Accuracy: +2.906% Weighted Average Precision: +3.5% Weighted Average Recall: +2.9% Weighted Average F1-score: +3.3%	
3.	Kek et al. (2021)	Comparisons Of Data Mining Classification Algorithms For Customers' Shopping	SMOTE	UCI Online shoppers' purchasing intention data set 12,330 instances	**Best Performance:** Random Forest Specificity: 0.936 Accuracy: 0.943	**Classification Algorithms:** Decision Tree, SVM, Logistic Regression, Random Forest

		Intention In E-Commerce		(84.5% negative class, 15.5% positive class)	Precision:0.938 Recall: 0.950 F1:0.944 Receiver Operating Characteristic Curve: 0.943	**Other pre-processing techniques:** z-score normalization, **Validation:** 5-fold cross-validation
4.	Muda et al. (2020)	Prediction of Online Shopper's Purchasing Intention Using Binary Logistic Regression, Decision Tree, and Random Forest	SMOTE	UCI Online shoppers' purchasing intention data set 12,330 instances (84.5% negative class, 15.5% positive class)	**Best Performance:** Random forest Cross-validation: 5-fold Accuracy: 88.35% AUC: 80.04%	**Classification Algorithms:** Decision Tree, Logistic Regression, Random Forest **Other pre-processing techniques:** Chi-square test, Multicollinearity test **Validation:** 5-fold cross-validation, 10-fold cross-validation

5.	Baati and Mohsil (2020)	Real-Time Prediction of Online Shoppers' Purchasing Intention Using Random Forest	SMOTE	UCI Online shoppers' purchasing intention data set 12,330 instances (84.5% negative class, 15.5% positive class)	• Random forest with oversampling obtained the highest accuracy of 86.78% and F1 score of 0.60. • The implementation of oversampling technique significantly increased the accuracy of Random Forest by 3.14% and F1 score by 0.50	**Classification Algorithms:** Naïve Bayes Classifier(NBC), Random Forest(RF) with CART, and C4.5 **Sampling method:** oversampling - SMOTE **Other pre-processing techniques:** Feature selection
6.	Obiedat (2020)	A Comparative Study of Different Data Mining Algorithms with Different Oversampli	• SMOTE • ADASYN • B-SMOTE • SVM-SMOTE • SMOTE-NC	UCI Online shoppers' purchasing intention data set 12,330 instances (84.5%	**Best classifier: Random Forest** Accuracy: 89.1% F1 score: 60.7% Precision: 69%	**Classification Algorithms:** Decision Tree, Multilayer Perceptron, Random Forest

		ng Techniques in Predicting Online Shopper Behaviour		negative class, 15.5% positive class)	**Best oversampling technique:** SVM-SMOTE with Random Forest F-measure: 92.3%	
7.	Geene (2020)	The Effects of an Imbalanced Dataset on Online Customer Intent Prediction	• Random Oversampling • Random Undersampling	Tooso fashion clickstream data set	**Best Performance:** LSTM with Oversampling ratio of 0.1 F1: 0.629 Accuracy: 0.962 AUC: 0.864	**Classification Algorithms:** Naïve Bayes, Markov Chain, Gradient Boosted Machine, Long Short-Term Model
8.	Esmeli et al. (2020)	Towards early purchase intention prediction in online session based retailing systems	• SMOTE • Random Undersampling	YooChoose RecSys data set	**Best performance:** Decision Tree with Undersampling AUC: 97.08%	**Classification Algorithms:** Naïve Bayes, Random Forest, Bagging, Decision Tree, K-

						nearest neighbours
9.	Sakar et al. (2019)	Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks	Oversampling	UCI Online shoppers' purchasing intention data set 12,330 instances (84.5% negative class, 15.5% positive class)	• MLP showed higher accuracy and F1 score than RF and SVM • Combining clickstream data obtained from the navigation path followed during the online visit with session information-based features that possess unique information about the purchasing	**Classification Algorithms:** Random Forest(RF) with C4.5, Support Vector Machine (SVM), and Multilayer Perceptron (MLP) **Other pre-processing techniques:** Feature selection **Validation:** 10-fold cross-validation

					interest improves the success rate of the system • MLP showed highest accuracy of 87.24% and F1 score of 0.86 followed by RBF SVM with accuracy of 84.88% and F1 score of 0.82.	
10.	Mokryn et al. (2019)	Will this session end with a purchase? Inferring current purchase intent of anonymous visitors	SMOTE	• YooChoose RecSys data set • Zalando data set	• applying SMOTE provides better results than undersampling	**Classification Algorithms:** logistic, Bagging, NBTree, XGBoost

2.8 Evaluation Metrics

Guo et al. (2017) highlighted that accuracy might be biased towards the majority class. This may lead to the Accuracy Paradox, where accuracy has a high value but other metrics have low values (Sridhar and Sanagavarapu, 2021). AUC, G-mean, and F1-score are often used as evaluation metrics for comparing and selecting models (Guo et al., 2017; Kotsiantis et al., 2005). Since AUC, G-mean, and F1-score take class distribution into account, hence it is not biased against the minority class (Guo et al., 2017; Kotsiantis et al., 2005).

To measure and analyse the performance of each classification algorithm paired with different SMOTE techniques, six basic model evaluation indicators are proposed: accuracy A, precision P, recall R, F1-score, ROC curve and AUC value.

	Predicted: True	Predicted: False
Actual: True	True Positive (TP)	False Negative (FN)
Actual: False	False Positive (FP)	True Negative (TN)

Figure 2.7: A confusion matrix for positive and negative instances.

(1) Accuracy, A is the proportion of instances that are correctly classified by the classifier:

$$A = \frac{TP + TN}{TP + FP + FN + TN} \quad (2.7)$$

(2) Recall, R is the proportion of positive instances that are correctly classified:

$$R = \frac{TP}{TP + FN} \quad (2.8)$$

(3) F1-score represents the harmonic mean between the precision and recall of the classifier:

$$F_1 = 2 \bullet \frac{Precision \bullet Recall}{Precision + Recall} \qquad (2.9)$$

2.9 Data Set Overview

In this study, the data set used is the Online Shoppers Purchasing Intention data set from Sakar et al. (2018). This data set contains 12,330 instances, of which 84.5% of the instances belong to the negative class and 15.5% are from the positive class. The target class in this data set is the binary attribute "Revenue", where the values are either "True" or "False". The unbalanced distribution of instances is shown in Figure 2.8.

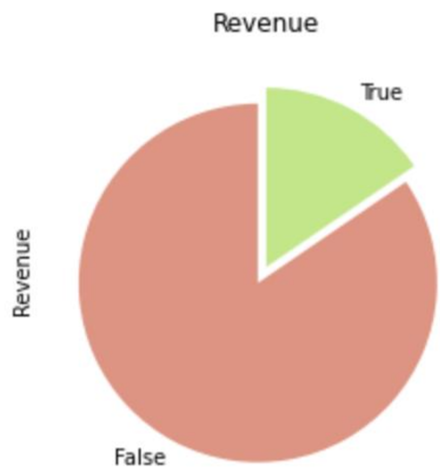


Figure 2.8: The pie chart of the proportion of true and false instances in the target class, Revenue.

There are 18 attributes in the data set consisting of ten numerical and eight categorical attributes. One significant analysis is the number of successful transactions made per month is the highest in November.

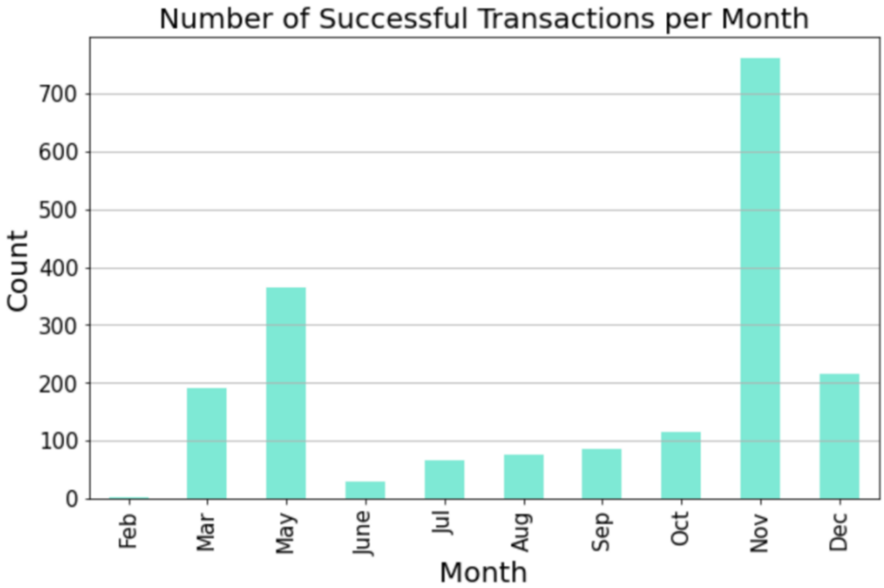


Figure 2.9: Bar graph of the number of successful transactions per month.

Another point worth noting is the number of successful transactions surges outstandingly on special days.

Figure 2.10: The bar graph shows the number of successful transactions when a special day is near.

Returning visitors have a higher tendency to be shoppers with low purchasing intent than new visitors.

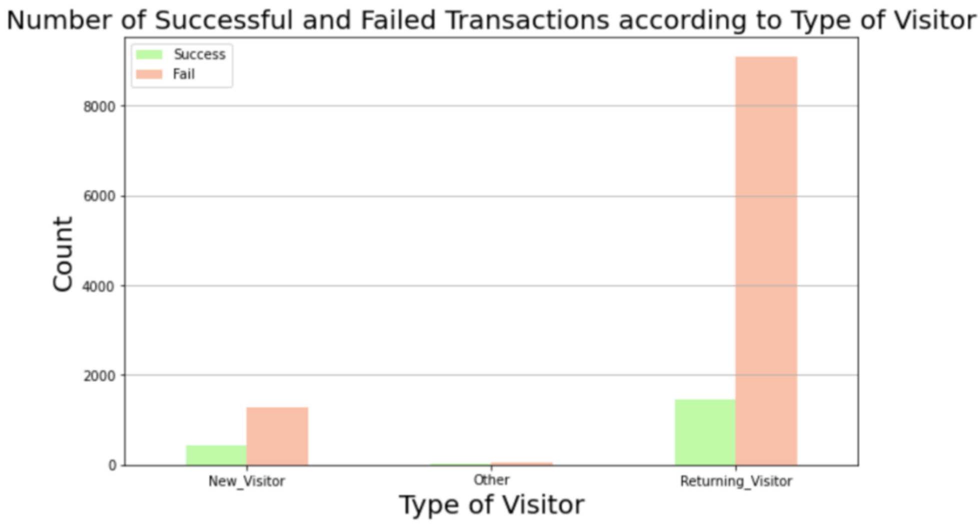


Figure 2.11: Bar graph of the number of successful and failed transactions for each type of visitor.

Table 2.4: The Numerical Features of Online Purchasing Intention Data Set by Sakar et al. (2019)

Administrative	The number of unique page categories visited by the visitor during the session.
Administrative Duration	Represent the total duration spent in each of these page categories.
Informational	Number of pages visited by the visitor regarding the purchasing site's Web site, communication, and address information.
Informational Duration	Total quantity of time (in seconds) spent on informative pages by visitors.
Product Related	Quantity of pages visited by a visitor related to a product
Product Related Duration	Total time (in seconds) spent on product-related pages by a visitor.
Bounce rate	Average bounce rate value of the visitor's frequented pages
Exit rate	Average exit rate value of the visitor's visited pages
Page value	Average page value of the pages that a visitor views
Special day	The proximity of the site visit to a memorable day

Table 2.5: The Categorical Features of Online Purchasing Intention Data Set by Sakar et al. (2019)

OperatingSystems	Operating system of the visitor
Browser	Browser used by the visitor
Region	Geographic region from which the visitor initiated the session.
TrafficType	The traffic source that led the visitor to the website (e.g., banner, SMS, direct).
VisitorType	Visitor type as ‘‘New Visitor,’’ ‘‘Returning Visitor,’’ and ‘‘Other’’
Weekend	Weekend value signifying whether the visit date is a weekend.
Month	Month value of the visit date
Revenue	Class label signifying whether a transaction was completed during the visit.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This section consists of the workflow summary, the detailed workflow and the research tools used.

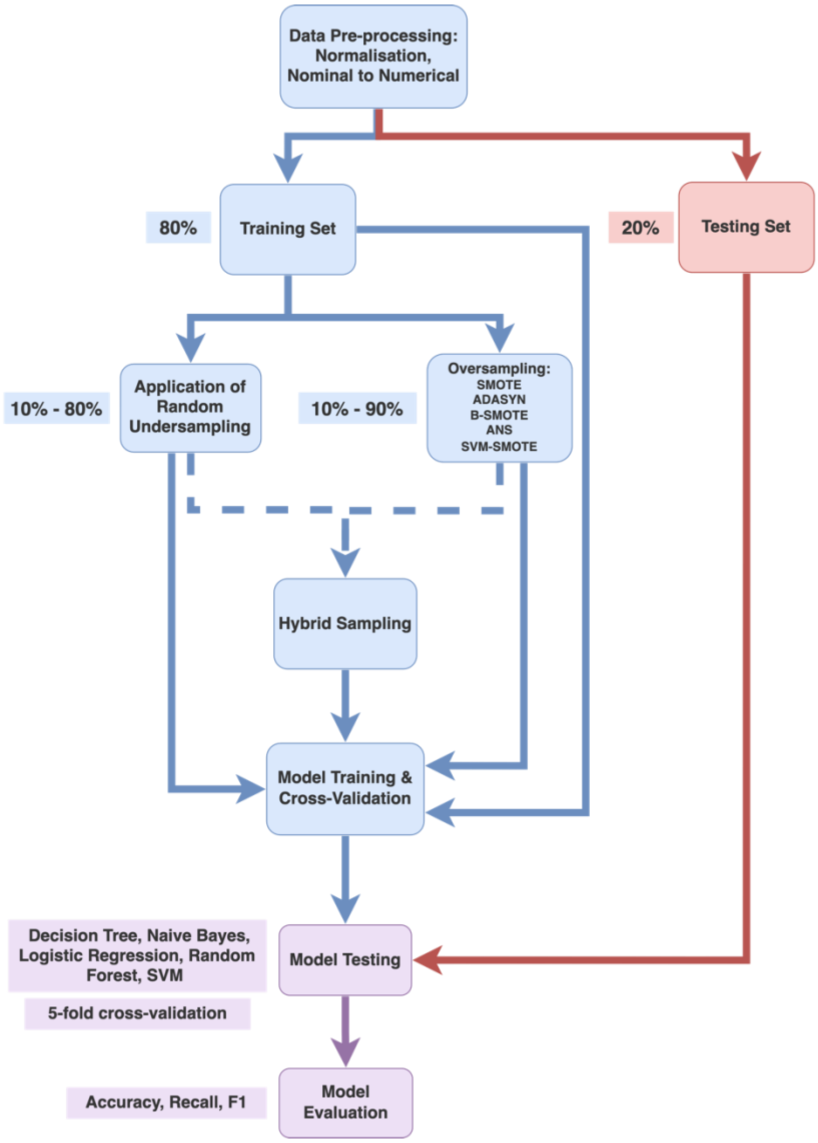


Figure 3.1: The workflow of the project.

3.2 Summary of Workflow

The project executed the following steps: data pre-processing, trainset and test set splitting, data sampling on the train set, model training and cross-validation on the train set, model testing on the test set and model evaluation.

3.3 Detailed Workflow

3.3.1 Data Pre-processing

Since this data set does not contain any null values, data cleaning is not required. However, there are two issues to be overcome: the existence of nominal features in the data set and the influence of the scale of variables on the models.

The first issue is caused by two nominal features in the data set: “Month” and “VisitorType”. This issue is addressed by applying one hot encoder to transform the features: “Month” and “VisitorType” into the numerical format. One hot encoder is applied instead of a label encoder because the features are not ordinal. In the following code snippet, firstly, “Month” and “VisitorType” were transformed into indicator variables and added to the data frame. The nominal “Month” and “VisitorType” were later dropped from the data frame.

```
df_ohe1 = pd.get_dummies(df[['Month']])
df = df.join(df_ohe1)

df_ohe2 = pd.get_dummies(df[['VisitorType']])
df = df.join(df_ohe2)

df = df.drop(['Month','VisitorType'], axis=1)
df = df.reset_index(drop=True)
```

Two boolean data type columns “Weekend” and “Revenue” were converted into integers.

```
df['Weekend'] = df['Weekend'].astype(int)
df['Revenue'].astype(int)
```

The second issue, the influence of the scale of variables on the models, might create a bias in models. To address the second issue, a normalisation method Min-Max scaling, was applied to the data set. By applying Min-Max scaling, all features in the data set were transformed to within the range of [0,1]. With a normalised scale for all features, the tendency of bias caused by the scale of variables is prevented.

In the code snippet below, the features in the training set and test set were normalised with a MinMaxScaler separately.

```
X_train_minmax = []

from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
for i in range (0,10):
    X_train_minmax.append(min_max_scaler.fit_transform(X_samp[i]))

X_test_minmax = min_max_scaler.fit_transform(X_test)
```

3.3.2 Train-Test Splitting

To standardise the training set and test set, train-test-split was performed on the data set, and the train set and test set were exported into two separate CSV files. This way, the size of the majority and minority classes in the training set and test set were the same for every experiment.

Firstly, a train-test-split was performed with a test size of 0.20. 80% of the data was used for training, while the remaining 20% was reserved for testing.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42)
```

Then, the training set data frame was created by merging the training features (X_train) and the training target column (y_train). The data frame was later exported into a CSV file to be used in the model training phase.

```
train_df= X_train.merge(y_train.to_frame(), left_index=True,right_index=True)
train_df.to_csv('train_df.csv', index=False)
```

Similarly, the test set data frame was created by merging the test features (X_test) and the testing target column (y_test) and was later exported into a CSV file to be used in the model testing phase.

```
test_df= X_test.merge(y_test.to_frame(), left_index=True,right_index=True)
test_df.to_csv('test_df.csv', index=False)
```

3.3.3 Sampling Methods

Four categories of experiments were carried out in this project. The four categories of experiments were differentiated by whether the training set had been applied without any sampling, with undersampling only, with oversampling only or with hybrid sampling.

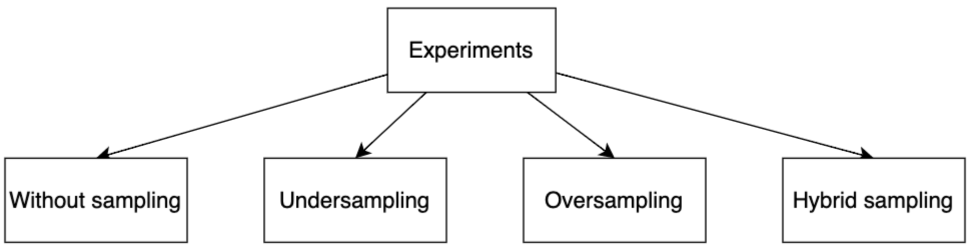


Figure 3.2: The category of experiments performed.

The first category of experiments was carried out on the training set without applying sampling methods.

In the second category of the experiments where only undersampling is applied to the training set. Random undersampling was applied to the training set by applying undersampling rates from 10% to 80% by an increment of 10%.

The third category was carried out with only oversampling applied to the training set. Five variants of SMOTE: (i) Standard SMOTE, (ii) ADASYN, (iii) ANS, (iv) B-SMOTE and (v) SVM-SMOTE, were applied to training sets separately. Each variant of SMOTE was applied to the training set with oversampling rates from 10% to 90% by an increment of 10%.

The fourth category was carried out by applying the hybrid sampling method to the training set. In this method, a hybrid of undersampling and oversampling was applied to reduce and increase the size of the majority and minority classes of the unbalanced data set used in this project. Undersampling rates from 10% to 80% and oversampling rates from 10% to 90% were used in combinations on the training set.

Prior to oversampling, undersampling is performed by using a data mining tool Waikato Environment for Knowledge Analysis (WEKA). The count of majority instances was calculated according to the undersampling ratio and used as the input to undersample the majority instances using the “SpreadSubsample” function in WEKA. The undersampled data sets were exported as CSV files to be used in the later phases.

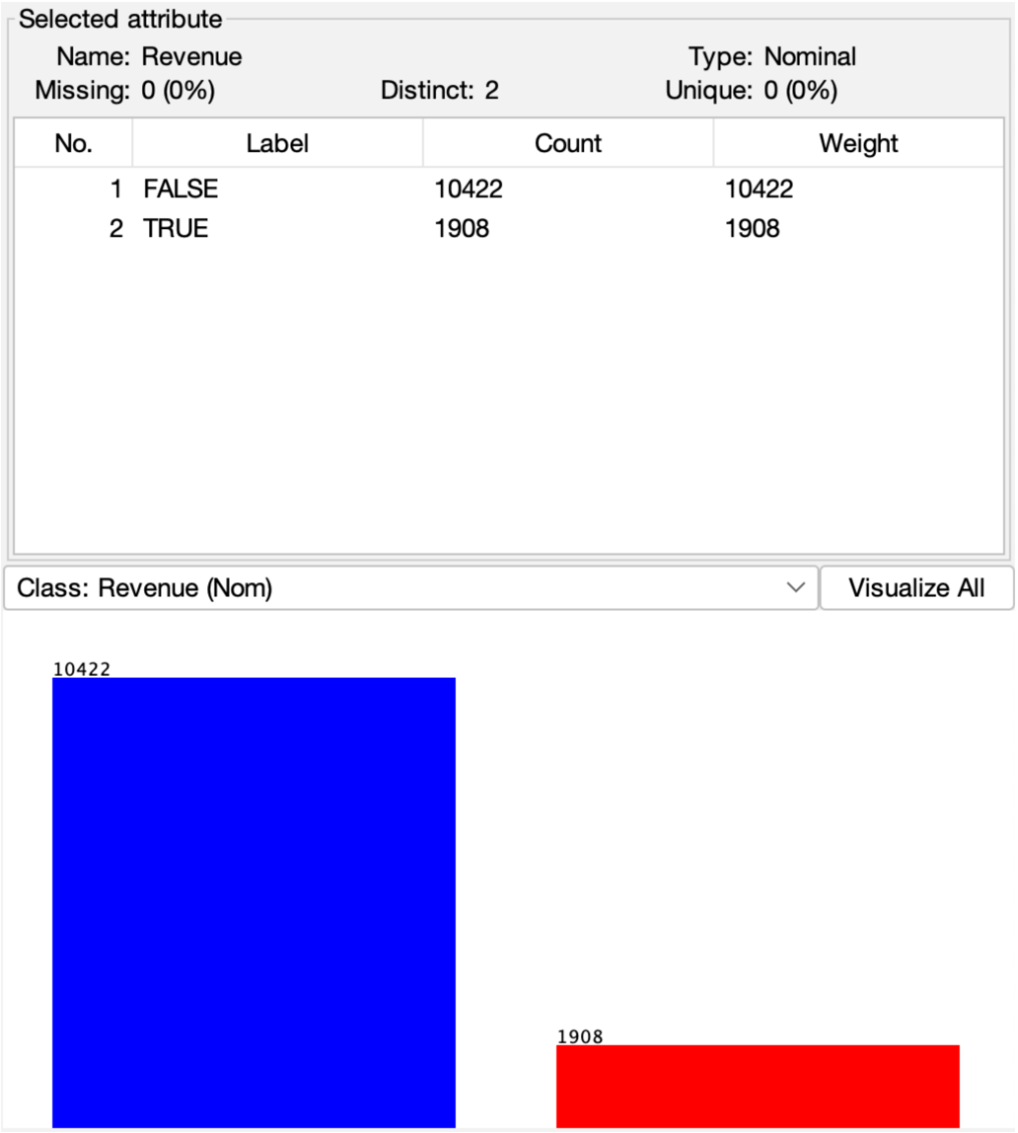


Figure 3.3: Proportion of data classes before applying undersampling.

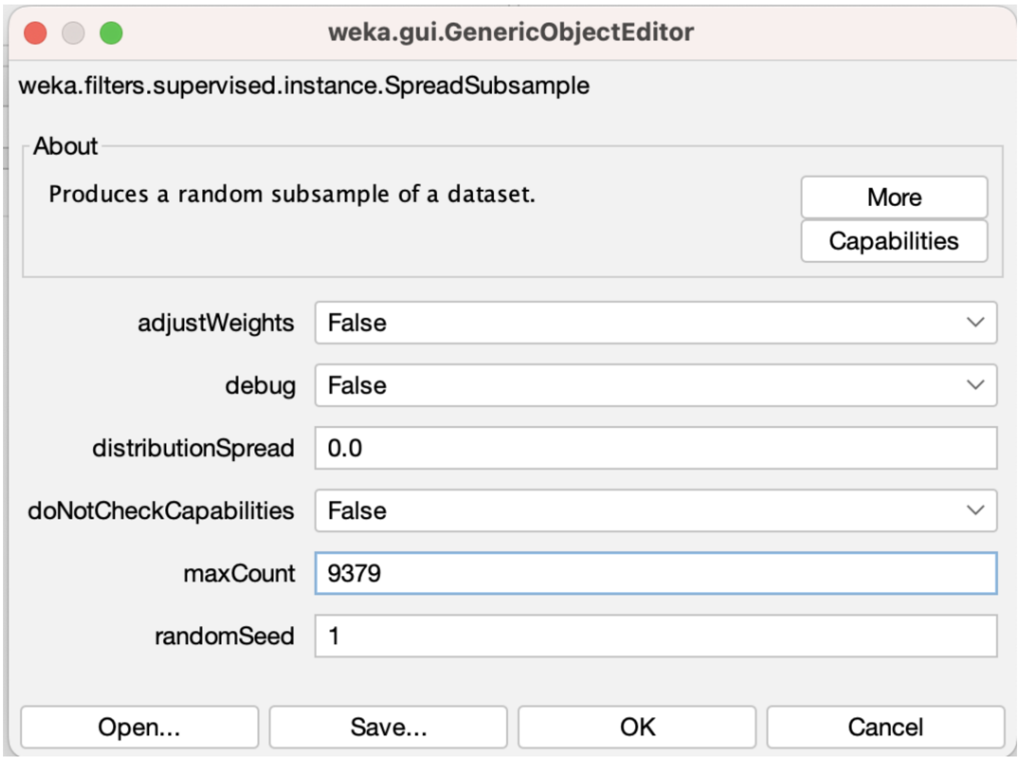


Figure 3.4: Demonstration of undersampling using WEKA's "SpreadSubsample" function.

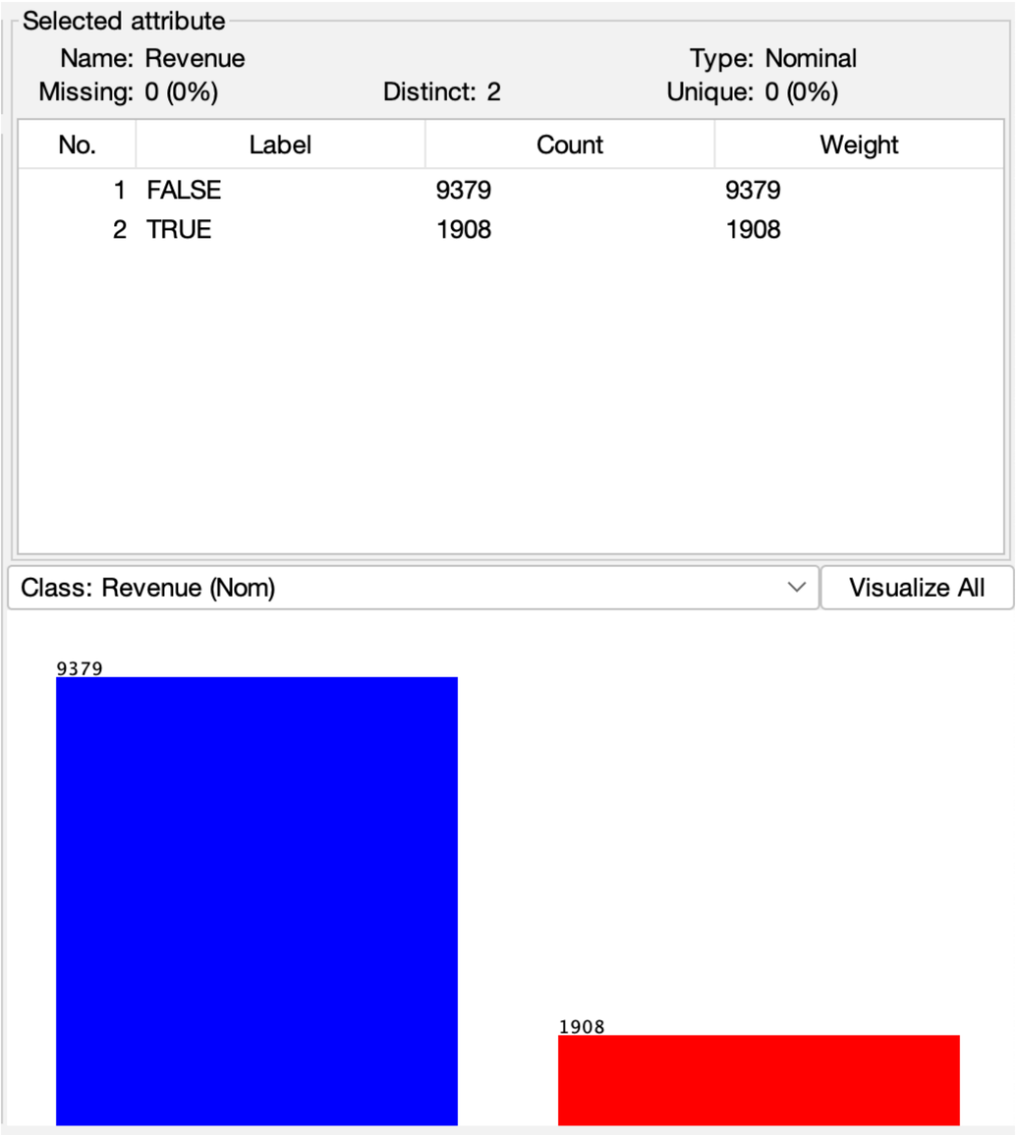


Figure 3.5: Proportion of data classes after applying 10% undersampling.

The following code snippet demonstrates the application of oversampling technique. In this example, Standard SMOTE was applied to produce nine arrays of training data oversampled at the rate from 0% to 90%, with an increment of 10%. Note that only Standard SMOTE, ADASYN, ANS, and Borderline_SMOTE used are from the smote_variants library.

```
for i in range (1,10):
    ratio = 0.1 * (i*0.2486)
    print(ratio)
    oversampler_list.append(sv.SMOTE(proportion=ratio))
    xi,yi= oversampler_list[i-1].sample(X_array, y_array)
    X_samp.append(xi)
    y_samp.append(yi)
```

SVM-SMOTE used is from Imbalanced-learn's oversampling library. The data set was oversampled by applying SVM-SMOTE to produce nine arrays of training data at the rate from 0% to 90%, with an increment of 10%.

```
for i in range (1,10):
    ratio = 0.1 * (((i-1)*0.199)+2.186)
    print(ratio)
    oversampler_list.append(os.SVMSMOTE(sampling_strategy=ratio))
    xi,yi= oversampler_list[i-1].fit_resample(X_array, y_array)
    X_samp.append(xi)
    y_samp.append(yi)
```

3.3.4 Model Training and Cross-Validation

After applying the sampling methods, the processed training set was passed to a classifier. Five classifiers: Decision Tree, Naïve Bayes, Logistic Regression, Random Forest and SVM, were included in this project. The formation of experiments with the classifiers and sampling methods is explained by category.

The training sets of the first category, without applying sampling methods, was fed into five classifiers respectively. As a result, five sets of experiments were formed and prepared for model testing and evaluation. Same as the first category, the training sets for the second category, which contains eight undersampling rates: 10%, 20%, 30%, 40%, 50%, 60%, 70%, and 80%, were fed into the five classifiers separately. This resulted in 40 experiments being formed. The training sets for the third category, with nine sampling rates for each of the five SMOTE variants, was fed into the five classifiers. As a result, 225 more experiments were formed. The training sets of the fourth category, applying undersampling rates from 10% to 80% and oversampling rates from 10% to 90%, were fed into the five classifiers. These combinations resulted in 1,745 experiments formed. 55 experiments were not performed when the undersampling rate is 70% and oversampling rate is above 60% as well as when the undersampling rate is 80% and the oversampling rate is above 10% when SVM-SMOTE is applied. This is due to the fact that SVM-SMOTE is from a different library and is unable to generate synthetic instances when the proportion of majority instances is minimal.

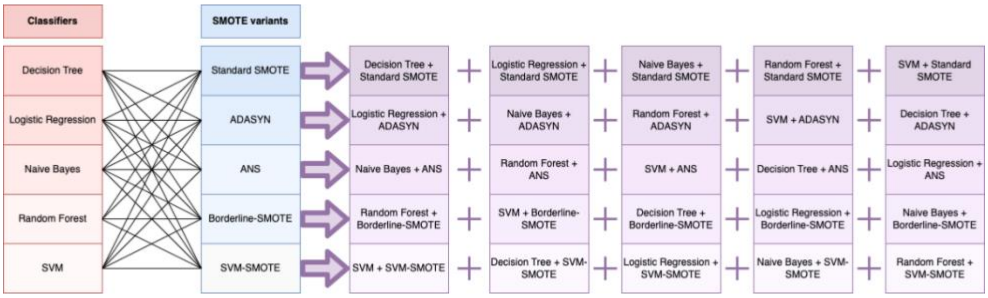


Figure 3.6: Combination of Classifier and SMOTE variants formed.

Combining all categories of experiments formed above, the total number of experiments constructed was 2,011.

The code snippet below demonstrates the process of fitting the nine arrays of oversampled training set into the Decision Tree classifier. A for loop is used to loop through the training set of different oversampling ratios.

```
from sklearn.tree import DecisionTreeClassifier
tree_model = []
for i in range (0,10):
    tree_model.append(DecisionTreeClassifier(random_state=42))
    tree_model[i].fit(X_train_minmax[i],y_samp[i])
```

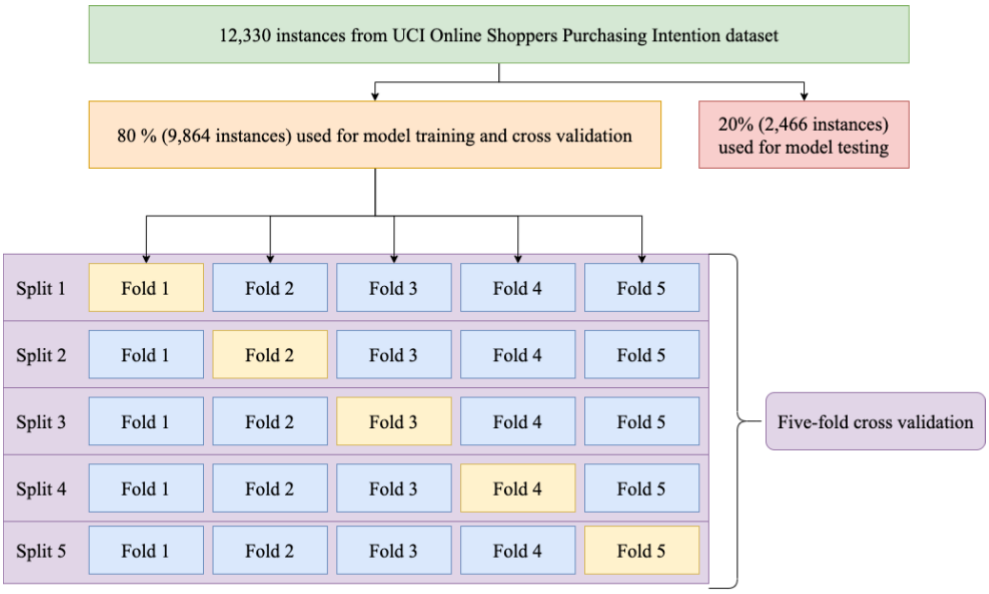


Figure 3.7: The illustration of the splitting of the data set.

In this project, to validate the model, k-fold cross-validation is chosen. The training sets and classifiers formed for the experiments above were used in k-fold cross-validation. Since positive instances are rare in this data set, 5-fold cross-validation is selected to ensure that each portion of data still contains a relatively significant proportion of the positive instances.

In the code snippet below, cross-validation was performed on the Decision Tree model with nine arrays of oversampled training sets. The result of the cross-validation was displayed based on the oversampling rate.

```
for i in range (0,10):
    print("\nOversample "+ str(i) +"0% Result:")
    decision_tree_result = cross_validation(tree_model[i], X_train_minmax[i],
y_samp[i], 5)
    print(decision_tree_result)
```

3.3.4 Model Testing and Evaluation

After fitting the models in each experiment, the test set is inputted into the models to test the models’ detection rate.

Model testing was performed by looping through the array of Decision Tree models which were fitted to different oversampling ratios.

```
y_pred = []
for i in range (0,10):
    y_pred.append(tree_model[i].predict(X_test_minmax))
```

After performing model testing, the detection rates of models in each experiment were evaluated. The detection rate was evaluated based on several metrics, including accuracy, recall, and F1-score.

The performance metrics were generated by looping through the prediction results of the model. The following code snippets demonstrate the process of generating and displaying the performance metrics.

i. Accuracy

```
from sklearn.metrics import accuracy_score
for i in range (0,10):
    print('Model '+ str(i) +'0% oversampling accuracy score:
{0:0.4f}'.format(accuracy_score(y_test, y_pred[i])))
```

ii. TPR / R1/ Minority Recall

```
def calc_recall (TP, FN):
    recall = TP / float(TP + FN)
    return recall


for i in range (0,10):
    print(str(i)+"0% oversampling",'Recall or Sensitivity :
{0:0.4f}'.format(calc_recall(matrix[i][0],matrix[i][3])))
```

iii. TNR / R0 / Majority Recall

```
def calc_recall (TF, FP):
    recall = TF / float(TF + FP)
```

```
    return recall

for i in range (0,10):
    print('{0:0.4f}'.format(calc_recall(matrix[i][1],matrix[i][2])))
```

iv. F1-score

```
def calc_f1 (TP, FP, FN):
    f1 = 2 * (TP / float(TP + FN) * TP / float(TP + FP) / float(TP / float(TP + FP) +
TP / float(TP + FN)))
    return f1

for i in range (0,10):
    print(str(i+1)+"0% oversampling",'f1 score :
{0:0.4f}'.format(calc_f1(matrix[i][0], matrix[i][2], matrix[i][3])))
```

3.4 Evaluation Metrics

The recall was used as the metric on the majority and minority classes of the data set for choosing the best-performing classifier and the right combination of hybrid sampling. Besides recall, accuracy and F1 were also used if a tie in the recall occurred. The undersampling technique applied in this project was Random Undersampling. On the other hand, the oversampling techniques used in this project were Standard SMOTE, ADASYN, ANS, B-SMOTE and SVM-SMOTE. The non-sampling, undersampling and oversampling and hybrid sampling results are described in Section 4.1, Section 4.2 and Section 4.3, and Section 4.4, respectively.

The best Classifier + Hybrid Sampling was chosen based on the following criteria. Firstly, the majority recall is compared among each Classifier + Hybrid Sampling at different undersampling and oversampling rates. Those with a recall below 0.80 for minority classes are filtered out unless none has a recall of at least 0.80. If that is the case, the consecutive highest recall shall be considered. Secondly, when there is a tie comparing the recall for the minority class, then the next criterion is to look at the recall for the majority class. Same for the majority class, Classifier + Hybrid Sampling with a recall lower than 0.80 are filtered unless no better options are left.

Thirdly, if a tie persists, then accuracy is taken into account to break the balance; the Classifier + Hybrid Sampling with a higher accuracy shall be selected. In the end, if the accuracy criterion does not break the tie, the F1 score shall be used to select the best Classifier + Hybrid Sampling.

3.5 Python and Libraries

This project was conducted by implementing the Python language which is supported by Jupyter Notebook. Python contains a wide range of library resources which eases machine learning. The libraries used for this project consist of the following:

Table 3.1: Python Libraries used and their usage.

Library	Usage
NumPy	To manipulate data using mathematical and logical operations.
Pandas	To perform data cleaning and analysis.
Matplotlib	To create visualisations such as pie charts, bar plots, and scatter plots
Scikit-learn (Imbalanced-learn)	To use tools such as classifying algorithms, scalers, evaluation metrics and more. The imbalanced-learn library is also used in this project to use oversamplers.
smote_variants	To use the ANS oversampler not available in Imbalanced-learn
seaborn	To create visualisations such as scatterplot matrix and the heatmap

3.6 Gantt Chart

This section describes the project timeline for this project. The tasks were completed in accordance with the planned project schedule to ensure the project's timely completion. The project schedule is depicted in Figure 3.9. Figures 3.10 and 3.11 depict the Gantt charts for the FYP 1 and FYP 2 projects. In the appendix section (Appendix A and B), a detailed version of the Gantt Charts is documented. During

FYP 1, the main focus was to construct a project proposal whereas FYP 2 mainly focused on implementing and realising the project.

Figure 3.8: Overall Gantt Chart for this project.

Figure 3.9: Gantt Chart for FYP 1.

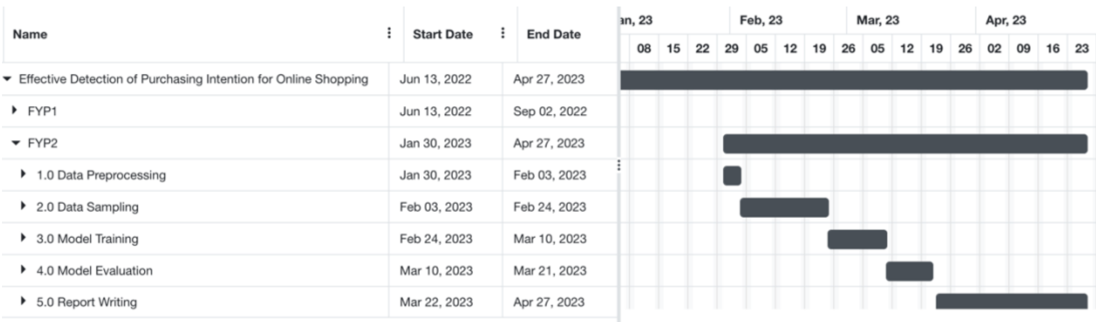


Figure 3.10: Gantt Chart for FYP 2.

3.7 Work Breakdown Structure

FYP 1

1.0 Problem Formulation and Project Planning

1.1 Review Background of Problem

1.2 Determine Problem Statement

1.3 Define Project Objectives

1.4 Determine Proposed Solution and Research Approach

1.5 Define Scope of Project

2.0 Literature Review Writing

2.1 Study on E-Commerce and its Rare Class Problem

2.2 Identify and Review Sampling Techniques

2.3 Identify and Review Classifiers

2.4 Study on Related Works and Compare them

2.5 Identify and Review Evaluation Metrics

3.0 Methodology Writing

3.1 Determine Workflow of Project

3.2 Determine Evaluation Criteria

4.0 Prototyping

4.1 Construct a Model for each Classifier

5.0 Improvisation on FYP 1

5.1 Check Flow and Continuity of Report

5.2 Amend Report Issues

FYP 2

1.0 Data Preprocessing

1.1 Data Transformation and Normalisation

2.0 Data Sampling

2.2 Apply Undersampling

2.3 Apply Oversampling

2.4 Apply Hybrid Sampling

3.0 Model Training

3.1 Construct Decision Tree Models

3.2 Construct Logistic Regression Models

3.3 Construct Naive Bayes Models

3.4 Construct Random Forest Models

3.5 Construct SVM Models

4.0 Model Evaluation

4.1 Test on Trained Models

4.2 Generate Result for Test

4.3 Collect and Organise Result in Excel

5.0 Report Writing

5.1 Revise Methodology

5.2 Generate Graphs for Collected Results

5.3 Analyse Results and Trends in Graphs

5.5 Prepare FYP Poster

5.4 Improvise Report

5.5 Prepare FYP Presentation

CHAPTER 4
RESULTS AND DISCUSSION

4.1 Introduction

This section presents the results of the four categories of experiments conducted by section.

4.2 Non-sampling Results

Table 4.1: Comparison of Results for each Classifier without pre-processing

(Notes : **A** indicates **Accuracy**, $\mathbf{R_o}$ indicates **Majority recall,** $\mathbf{R_1}$ indicates **Minority recall, F1** indicates **F1 score**, and the bolded rows indicate the best results for the classifier in the data set.)

Classifier	A	R_o	R_1	F1
Decision Tree	0.8524	0.9075	0.5766	0.5656
Logistic Regression	0.8723	0.9742	0.3625	0.4861
Naïve Bayes	0.8382	0.9971	0.0438	0.0828
Random Forest	0.8958	0.9664	0.5426	0.6344
SVM	0.8690	0.9835	0.2968	0.4303

Five experiments were performed on the data set without performing any pre-processing steps: undersampling and oversampling. In all five experiments, five classifiers were used: Decision Tree, Logistic Regression, Naïve Bayes, Random Forest and SVM. These experiments yield recall and accuracies for the majority class (R0) that are above 0.80 and 0.90, respectively. However, the recall for the minority class (R1) generated by all five experiments is less than 0.60, with Naïve Bayes producing the lowest R1 value of 0.0438. This observation demonstrates that the classifiers have a greater bias towards the "Do Not Buy" category, as it is the most numerous category. Classifiers tend to perform poorly in the presence of an unbalanced data set and its overlapping classes problem.

4.3 Undersampling Results

40 experiments were conducted on data sets with only undersampling applied. Each classifier is involved in eight of the 40 experiments. The R1 for all classifiers is less than 0.60 when the undersampling rate is 0%, with the Naïve Bayes classifier having the lowest R1 at 0.0438.

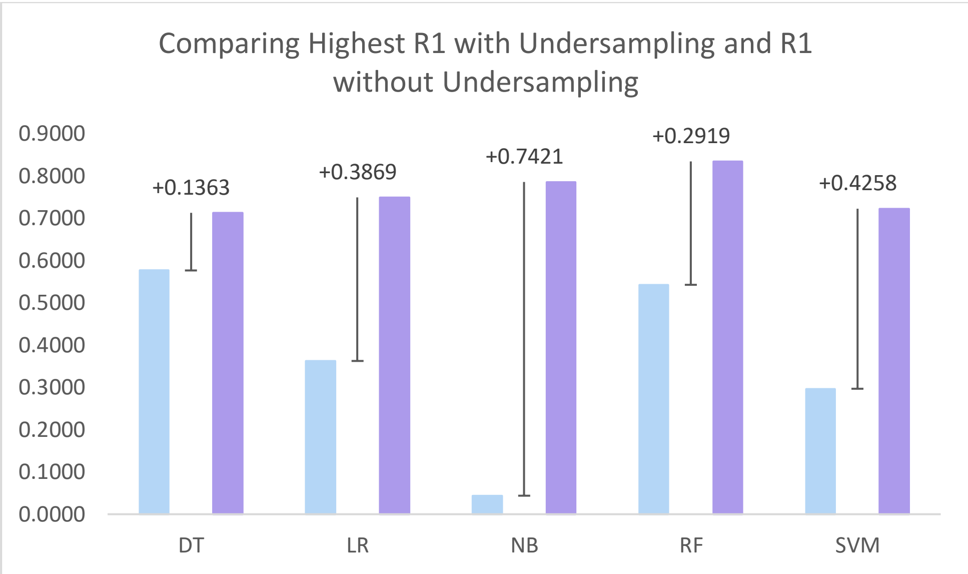


Figure 4.1: Comparing the highest R1 with undersampling (purple) and R1 without undersampling (blue).

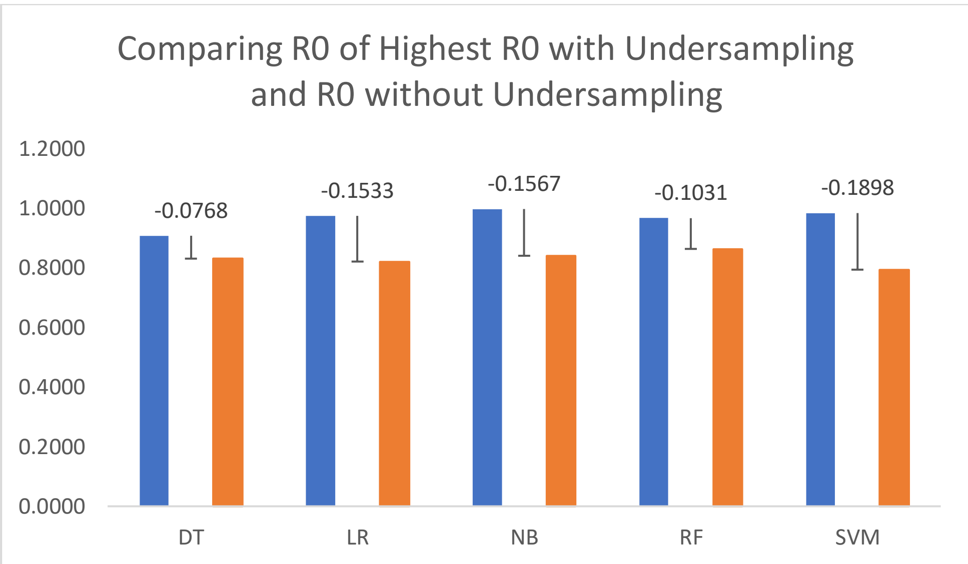


Figure 4.2: Comparing R0 of the highest R0 with undersampling (blue)and R0 produced without undersampling (orange).

As the rate of undersampling increases, so does the R1 for all classifiers. The Naïve Bayes classifier demonstrated the greatest improvement in R1 with an increase of 0.7421. At an undersampling ratio of 80%, the Random Forest classifier generated a maximum R1 of 0.8345. In contrast, R0 decreases as the undersampling rate increases. However, the decline in R0 is not as noticeable as the rise in R1. To support this claim, SVM produced the largest decrease at 0.1898, which is substantially less than R1's largest increase.

4.4 Oversampling Results

4.4.1 Decision Tree

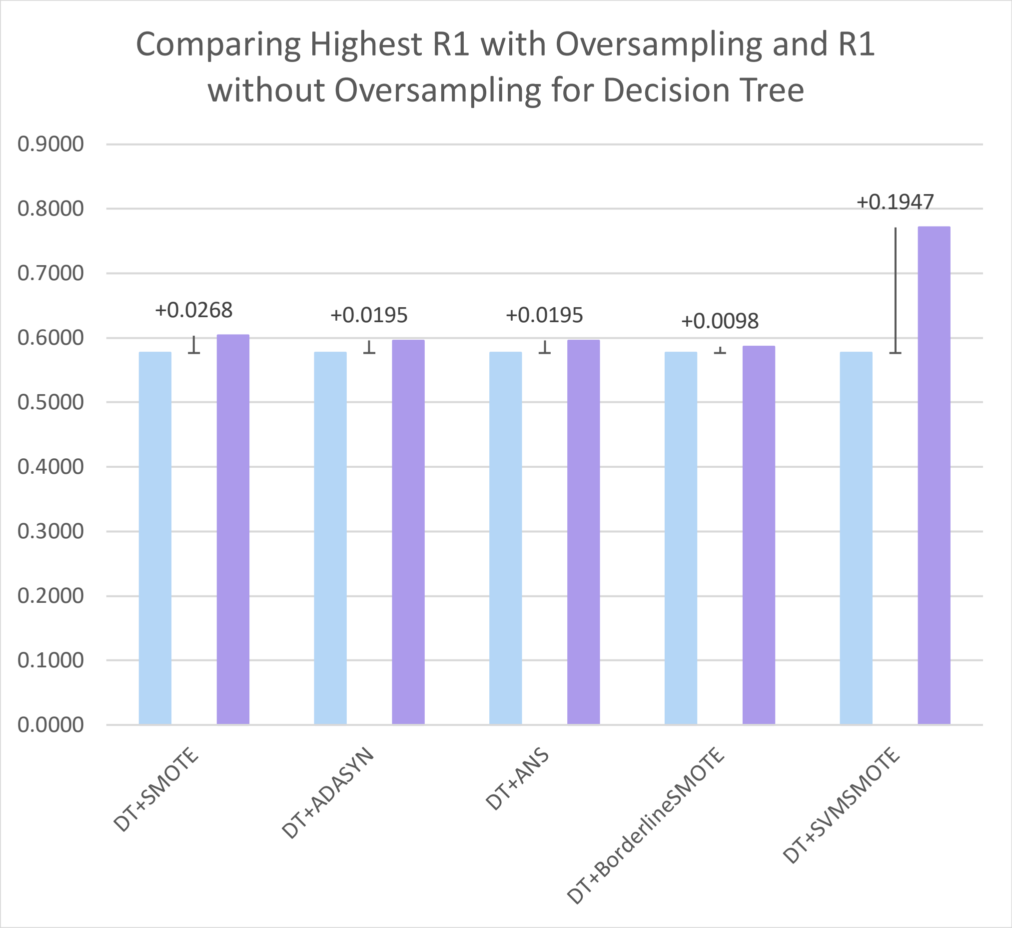


Figure 4.3: Comparing the highest R_1 with oversampling (purple) and R_1 without oversampling (blue) for the Decision Tree classifier.

With the exception of SVM-SMOTE, the differences between the highest R1(s) and R1 without oversampling for the Decision Tree are generally subtle. Compared to the other combinations, the Decision Tree + SVM-SMOTE has the greatest minority class recall difference. In general, the range of the other Decision Tree + oversampler pairs does not exceed 0.0268. The range for Decision Tree + SVM-SMOTE is 0.1947, which is seven times greater than the maximum range for other combinations.

4.4.2 Logistic Regression

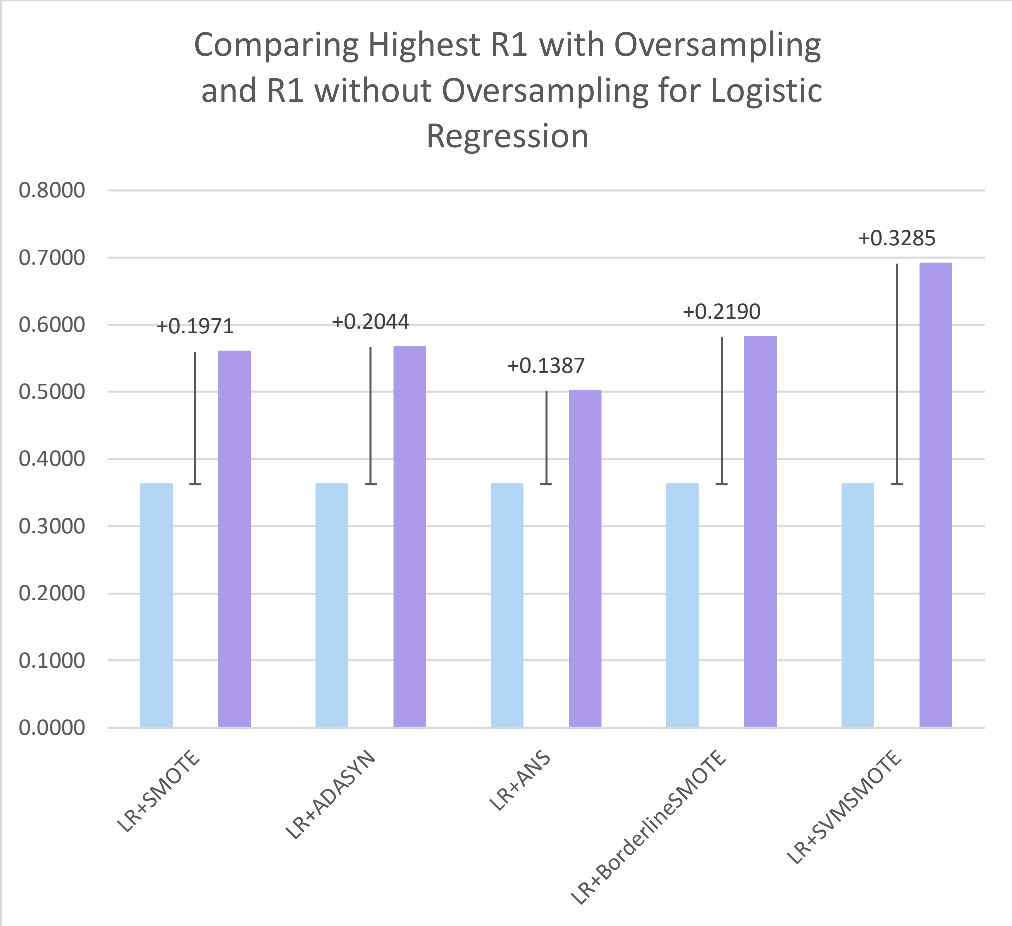


Figure 4.4: Comparing the highest R_1 with oversampling (purple) and R_1 without oversampling (blue) for the Logistic Regression classifier.

In general, the range for the difference between the highest R1 and R1 without oversampling for Logistic Regression + oversampler pairs is between 0.1387 and 0.2190. Among the Logistic Regression + oversampler pairs, LR + SVM-SMOTE appears to be the outlier. The exceptional range of R1 for LR + SVM-SMOTE is 0.3285, which is half as high as the maximum range of R1 or the other Logistic Regression + oversampler pairs.

4.4.3 Naïve Bayes

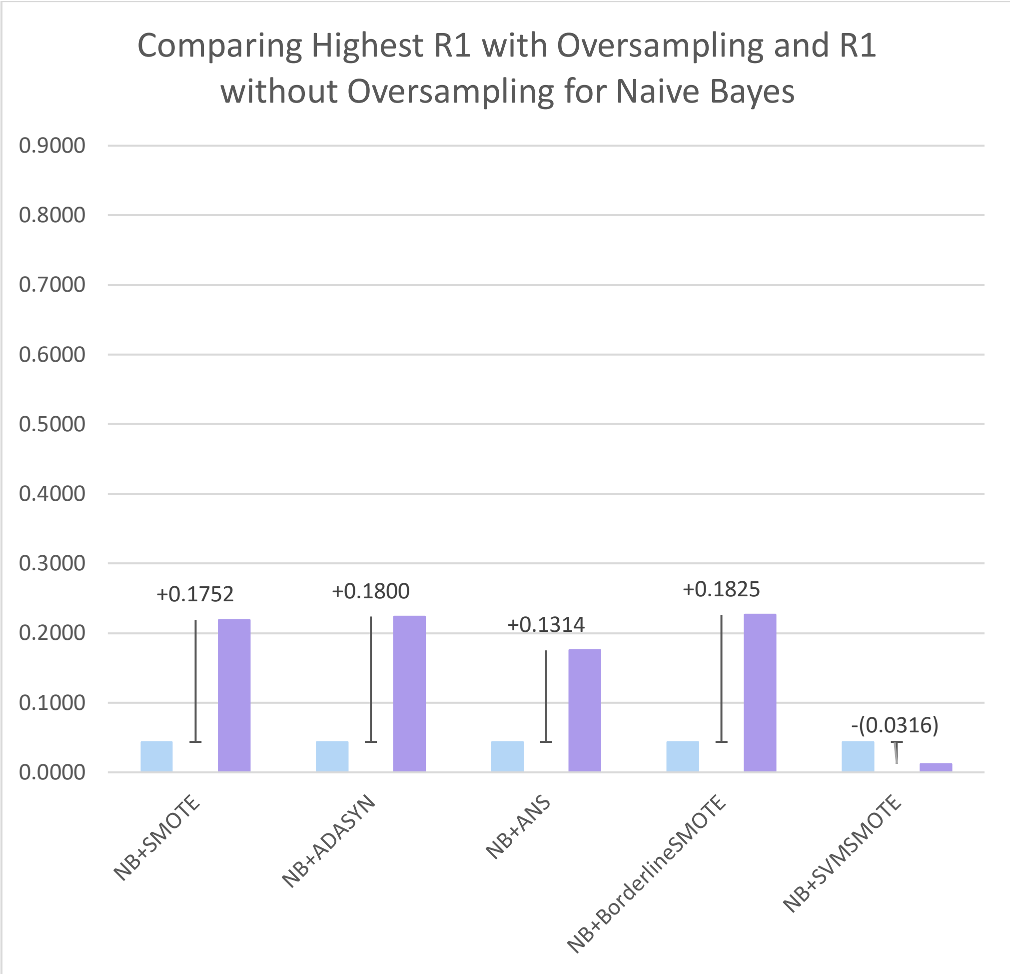


Figure 4.5: Comparing the highest R_1 with oversampling (purple) and R_1 without oversampling (blue) for the Naïve Bayes classifier.

The R1 generated by the Naive Bayes + oversampler pairs ranges from 0.0438 to 0.2263, with 0.2263 being the greatest R1 produced by ANS as the oversampler. Overall, the performance of all Naïve Bayes plus oversampler pairs in predicting the "Buy" class appears to be subpar. Worse yet, the NB + SVM-SMOTE pair has generated a turnover result in which the highest R1 is in fact lower than the R1 produced without the use of an oversampler.

4.4.4 Random Forest

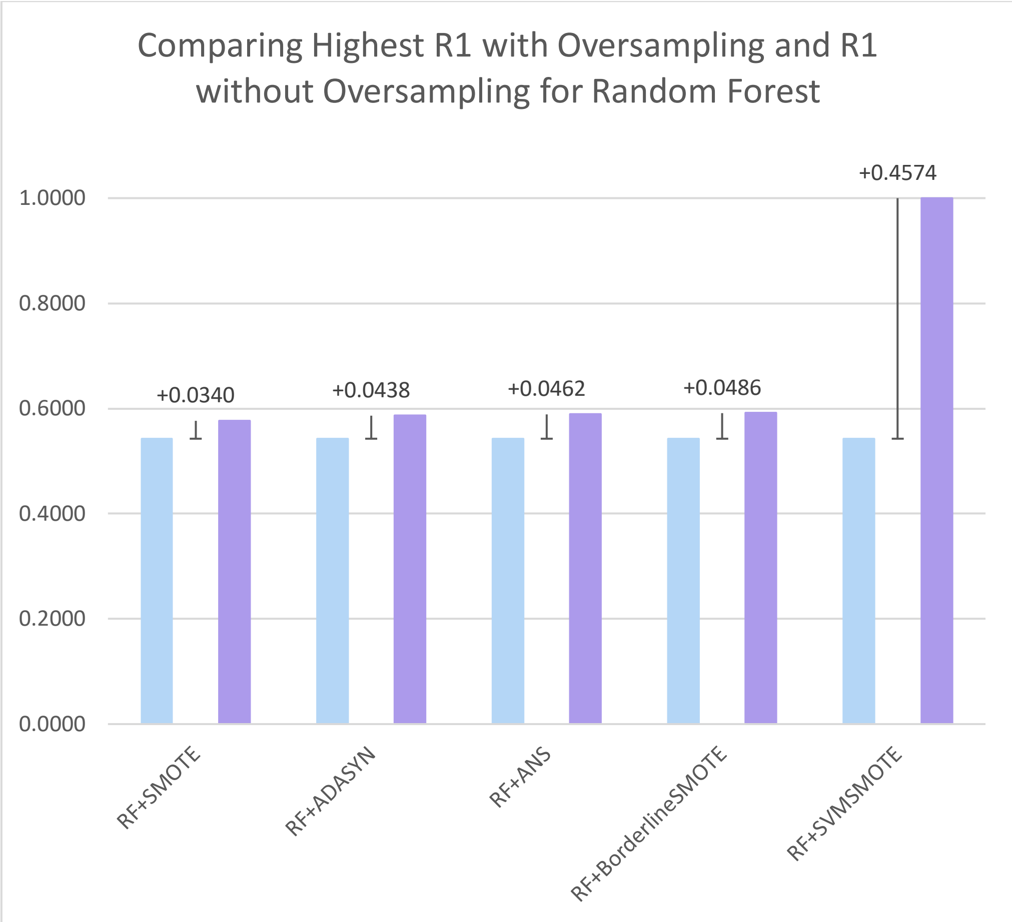


Figure 4.6: Comparing the highest R_1 with oversampling (purple) and R_1 without oversampling (blue) for the Random Forest classifier.

Without any sampling, the Random Forest generates an R1 of 0.5426. In general, the difference between the greatest R1(s) and the R1 without oversampling is minimal. Typically, the difference ranges between 0.0340 and 0.0486. Nevertheless, similar to other classifiers, SVM-SMOTE remains an anomaly when integrated with Random Forest. The maximum R1 produced by RF + SVM-SMOTE when the oversampling rate is 40% or greater is 1.0000. The difference between the maximum R1 generated by RF + SVM-SMOTE and the R1 without oversampling was a remarkable 0.4574.

4.4.5 SVM

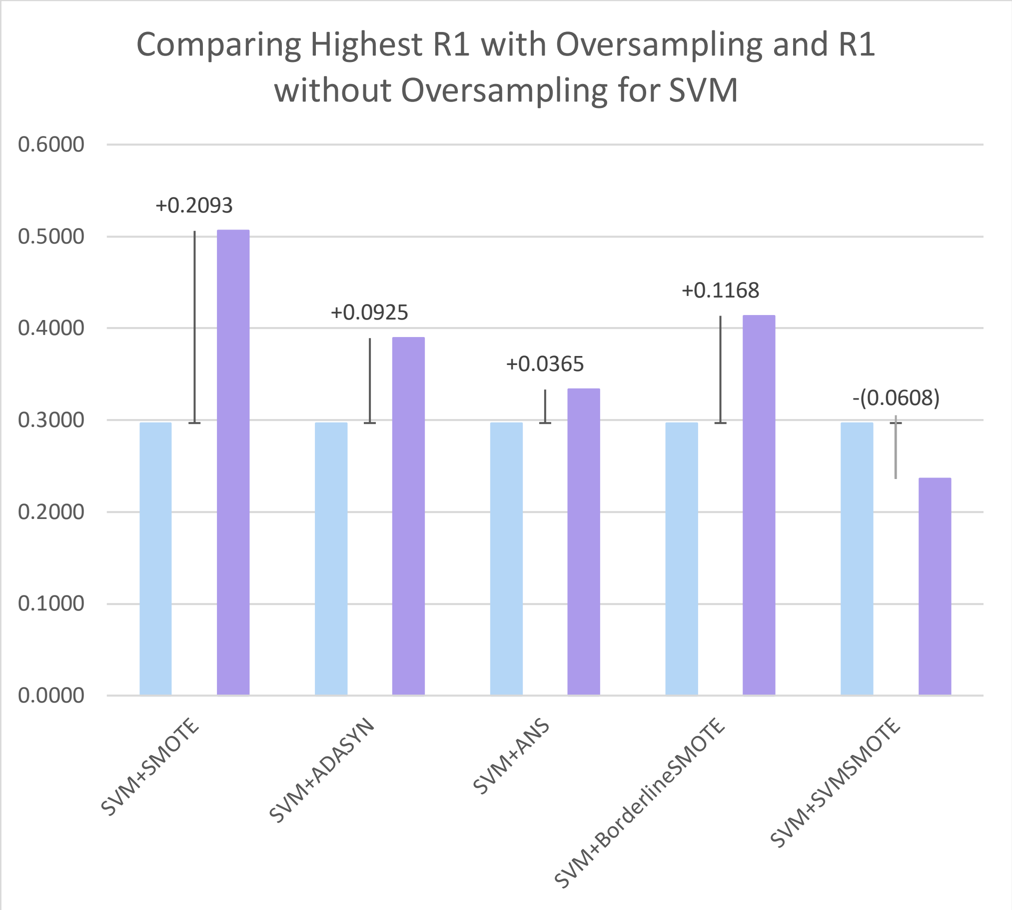


Figure 4.7: Comparing the highest R_1 with oversampling (purple) and R_1 without oversampling (blue) for the SVM classifier.

Without sampling, the SVM yields an R1 value of 0.2968. Each SVM + oversampler appears to have a vastly distinct R1 range. SVM + ADASYN produced the smallest difference (0.0365), while SVM + B-SMOTE produced the largest difference (0.2093). While other SVM + oversampler combinations generated positive differences regardless of magnitude, SVM + SVM-SMOTE generated a negative R1 difference. This indicates that the R1 produced by SVM + SVM-SMOTE at all oversampling rates is less than it would be without oversampling.

4.4.6 Summary of Oversampling Results

In comparison to other classifiers, Decision Tree + oversamplers (excluding SVM-SMOTE) produced the smallest variances in R1. Random forest, comparable to Decision Tree, positioned second for the fewest generated R1 differences. With or without oversampling, Naïve Bayes generated the lowest R1 values among all classifiers. With the exception of SVM, the other classifiers, including Decision Tree, Logistic Regression, Naïve Bayes, and Random Forest, when combined with oversamplers (with the exception of SVM-SMOTE), exhibit a similar trend for the range of R1. In contrast, the range of R1 generated by SVM plus oversamplers varies more widely. As far as classifiers + SVM-SMOTE comparisons are concerned, the classifiers appear to fall into two categories: positive range of R1 and negative range of R1 generated. Random Forest, Logistic Regression, and Decision Tree are examples of classifiers that generate a positive R1 range. Another noteworthy observation is that the generated positive range of R1 is always extraordinarily significant, distinguishing it from other oversampler combinations. On the other hand, Naive Bayes and SVM are classifiers that produced a negative R1 range. Negative ranges of R1 are generated by Naive Bayes and SVM integrated with SVM-SMOTE, but they are comparatively low. Negative R1 values generated by Naïve Bayes and SVM with SVM-SMOTE are -0.0316 and 0.0608, respectively.

When only oversampling is applied, both majority and minority recall remain plateaus with little increments and decrements that show little to no effect in improving the detection rate. From this observation, it is shown that using the oversampling technique alone does not lead to significant improvements in majority recall or minority recall.

4.5 Hybrid Sampling Results

Table 4.2: Comparison of the Best Results for each Classifier + Hybrid sampling set

(Notes: **UR** indicates **Undersampling ratio**, **OR** indicates **Oversampling ratio**, **A** indicates **Accuracy**, **R_0** indicates **Majority recall, R_1** indicates **Minority recall, F1** indicates **F1 score**, and the bolded rows indicate the best results for the classifier in the data set.)

Classifier	UR	OR	A	R_0	R_1	F1
Decision Tree	80%	30% (Standard SMOTE)	0.8147	0.8204	0.7859	0.5857
	80%	50% (ADASYN)	0.8021	0.8015	0.8054	0.5757
	80%	20% (ANS)	0.8106	0.8136	0.7956	0.5834
	80%	50% (B-SMOTE)	0.8240	0.8248	0.8200	0.6083
	30%	40% (SVM-SMOTE)	0.7944	0.8083	0.7251	0.5403
Logistic Regression	80%	50% (Standard SMOTE)	0.7307	0.7061	0.8540	0.5139
	70%	80% (ADASYN)	0.7656	0.7538	0.8248	0.5398
	80%	60% (ANS)	0.7311	0.7080	0.8467	0.5121
	70%	40% (B-SMOTE)	0.7283	0.7056	0.8418	0.5081
	40%	50% (SVM-SMOTE)	0.7855	0.7908	0.7591	0.5412
	80%	40%	0.6241	0.5839	0.8248	0.4224

Naïve Bayes		(Standard SMOTE)				
	70%	0% (ADASYN)	0.7859	0.8404	0.7859	0.4442
	80%	30% (ANS)	0.6367	0.6019	0.8102	0.4264
	80%	30% (B-SMOTE)	0.6342	0.6005	0.8029	0.4225
	80%	0% (SVM-SMOTE)	0.6764	0.6715	0.7007	0.4192
Random Forest	80%	80% (Standard SMOTE)	0.8528	0.8521	0.8564	0.6598
	80%	90% (ADASYN)	0.8512	0.8501	0.8564	0.6573
	80%	80% (ANS)	0.8589	0.8603	0.8516	0.6679
	80%	80% (B-SMOTE)	0.8483	0.8467	0.8564	0.6531
	80%	0% (SVM-SMOTE)	0.8585	0.8633	0.8345	0.6628
SVM	80%	30% (Standard SMOTE)	0.7076	0.6856	0.8175	0.4824
	80%	80% (ADASYN)	0.7129	0.6910	0.8224	0.4884
	80%	80% (ANS)	0.7271	0.7114	0.8054	0.4959
	80%	60% (B-SMOTE)	0.7291	0.7105	0.8224	0.5030
	70%	60% (SVM-SMOTE)	0.8049	0.8161	0.7494	0.5615

Based on Table 4.2, 19 out of the 25 sets of Classifier + Hybrid sampling sets utilised a random undersampling rate of 80%. This demonstrates that the majority of classifiers generate a higher detection rate when the majority of data in the training set is 80% smaller than its original size. When 80% of the training set is undersampled with the random undersampler, the data instances of the majority class in the training set decrease from 8,367 to 1,673. In comparison to the other undersampling ratios, the proportion of majority data instances is the closest to the proportion of minority data instances at this point. When the proportion of the two data classes is similar, the "Not Buy" and "Buy" classes have equal weight. Consequently, the propensity of classifiers to be biased towards the "Not Buy" class decreases substantially.

4.5.1 Decision Tree

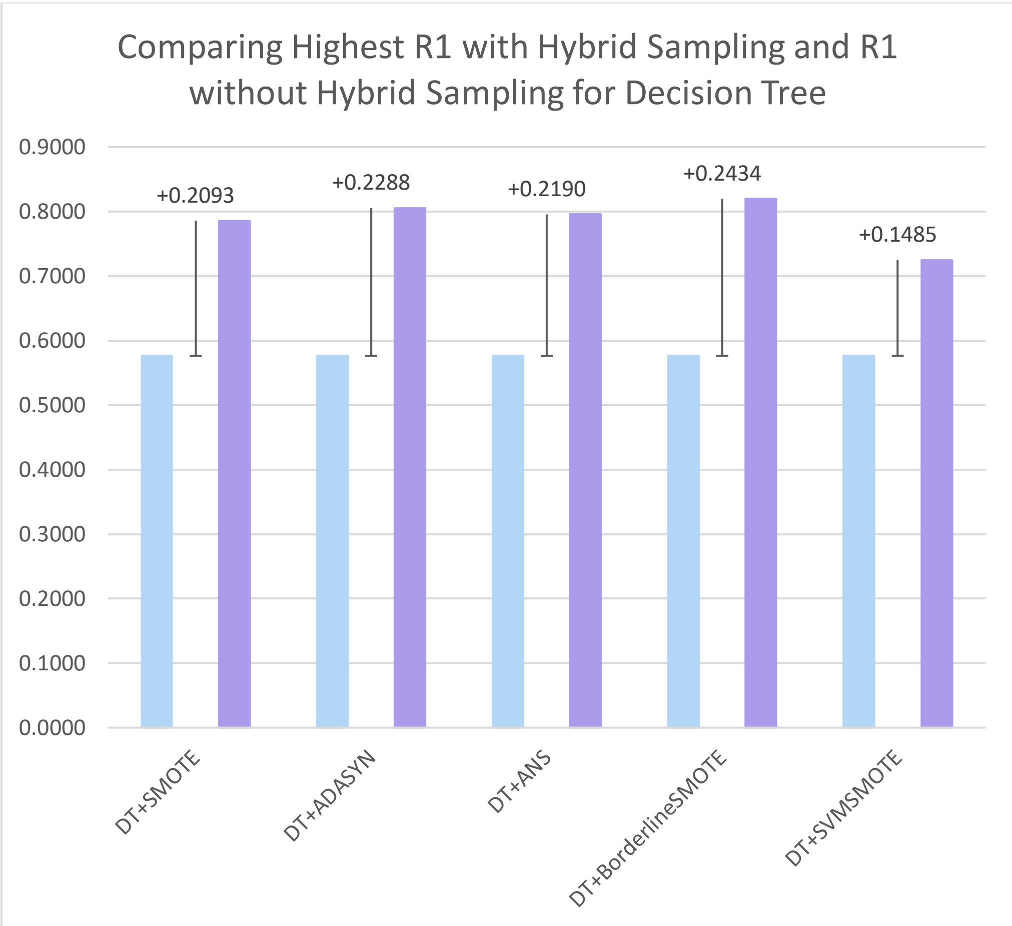


Figure 4.8: Comparing the highest R_1 with hybrid sampling (purple) and R_1 without sampling (blue) for the Decision Tree classifier.

Overall, hybrid sampling enhanced the Decision Tree classifier's detection rate. The integration of B-SMOTE and Decision Tree yields the maximum R1 (0.8200), producing a difference of 0.2434. With the exception of SVM-SMOTE, the range of R1 generated by Decision Tree + oversamplers is generally above 0.20. Combining Decision Tree with SVM-SMOTE produced the narrowest range in R1; the range was only 0.1485.

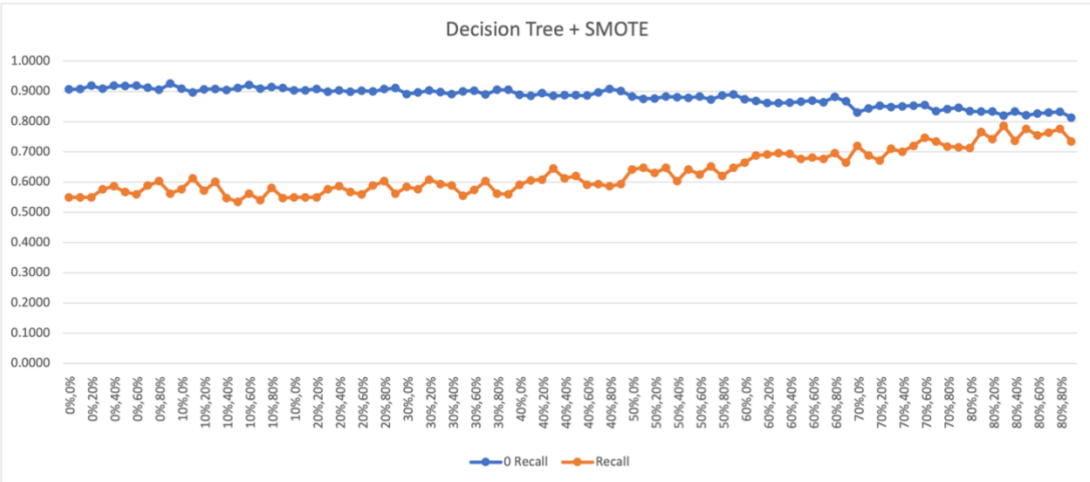


Figure 4.9: Majority and minority recall of Decision Tree combined with Standard SMOTE.

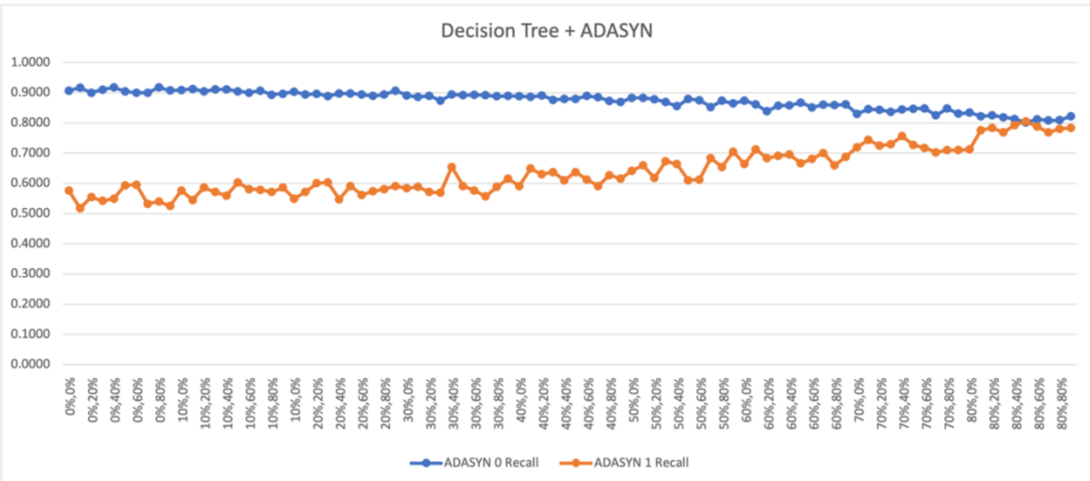


Figure 4.10: Majority and minority recall of Decision Tree combined with ADASYN.

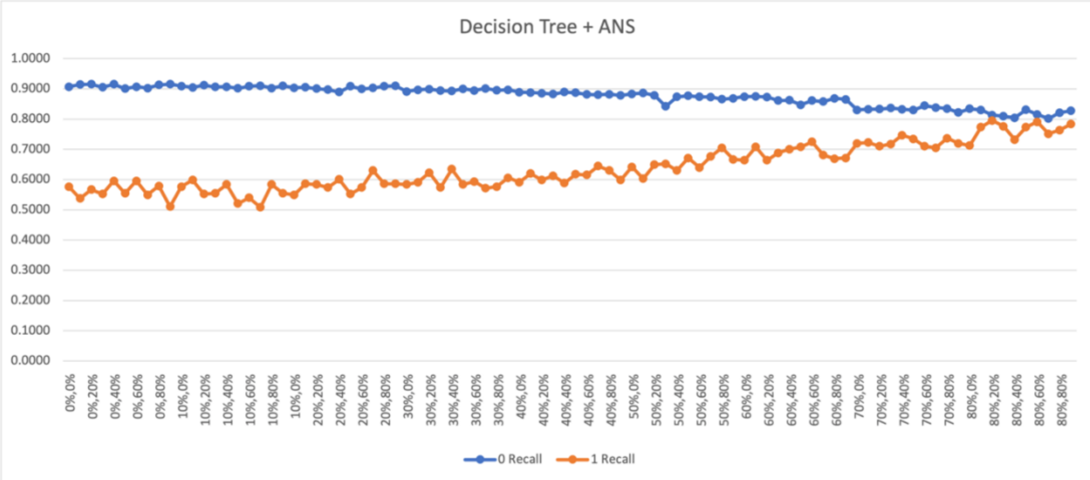


Figure 4.11: Majority and minority recall of Decision Tree combined with ANS.

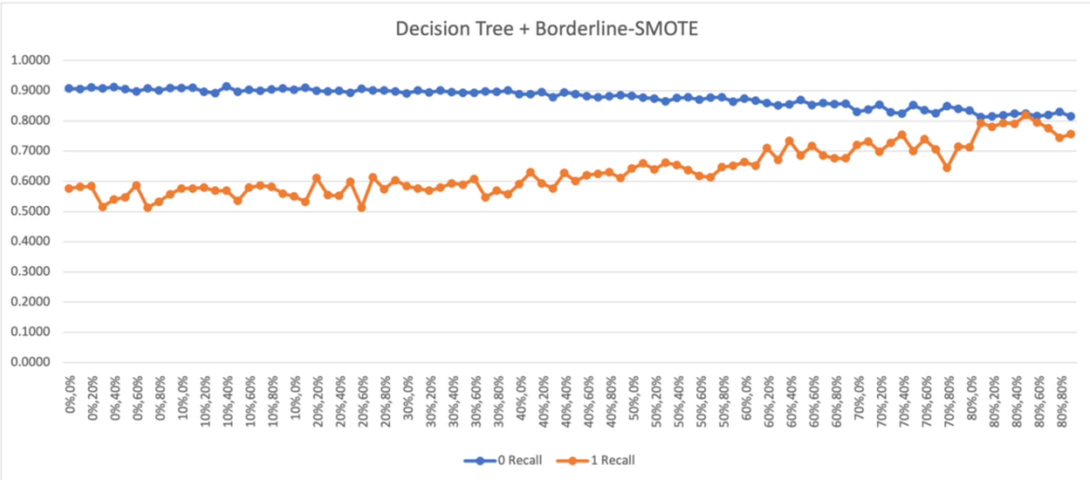


Figure 4.12: Majority and minority recall of Decision Tree combined with B-SMOTE.

The Standard SMOTE, ADASYN, ANS, and B-SMOTE oversampled data with Decision tree classifier experiments reveal several common patterns. All four experiments involving an oversampler exhibited converging graphs. When the undersampling rate decreases, both accuracy and R0 decline but remains above 0.80. However, R1 of the combinations above increased overall, with a few minor declines at specific instances. F1 is relatively stable and ranges from 0.5 to 0.6.

4.5.2 Logistic Regression

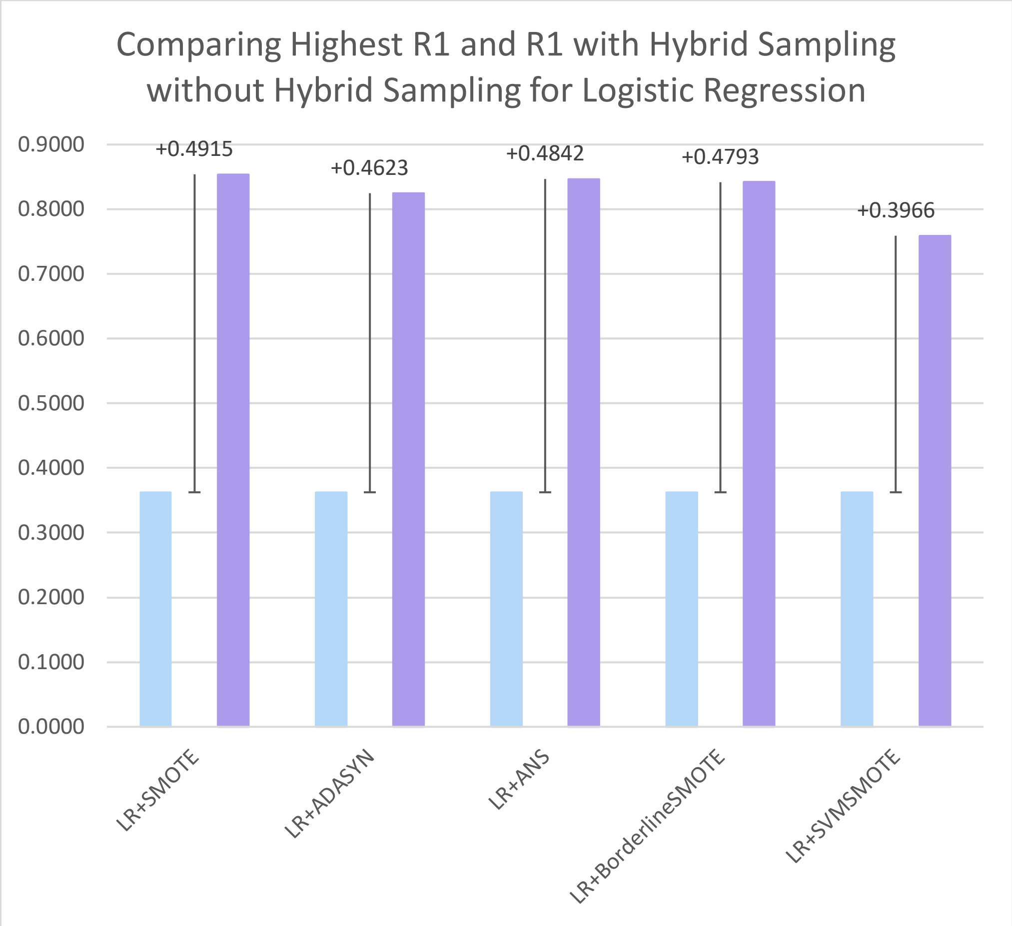


Figure 4.13: Comparing the highest R_1 with hybrid sampling (purple) and R_1 without sampling (blue) for the Logistic Regression classifier.

The application of the hybrid sampling technique with the Logistic Regression classifier resulted in a substantial improvement in the detection rate of purchasing intention. Random Undersampling (RUS) and Standard SMOTE with the Logistic Regression classifier yield the highest R1, 0.8540. This combination increased the R1 value by 0.4915 percentage instances, the greatest increase among all Logistic Regression + hybrid sampling combinations. The range of improvement for all Logistic Regression + hybrid sampling methods except SVM-SMOTE is greater than 0.45. Logistic Regression + RUS + SVM-SMOTE improved R1 by 0.3966, which was marginally less than the other Logistic Regression + hybrid sampling combinations.

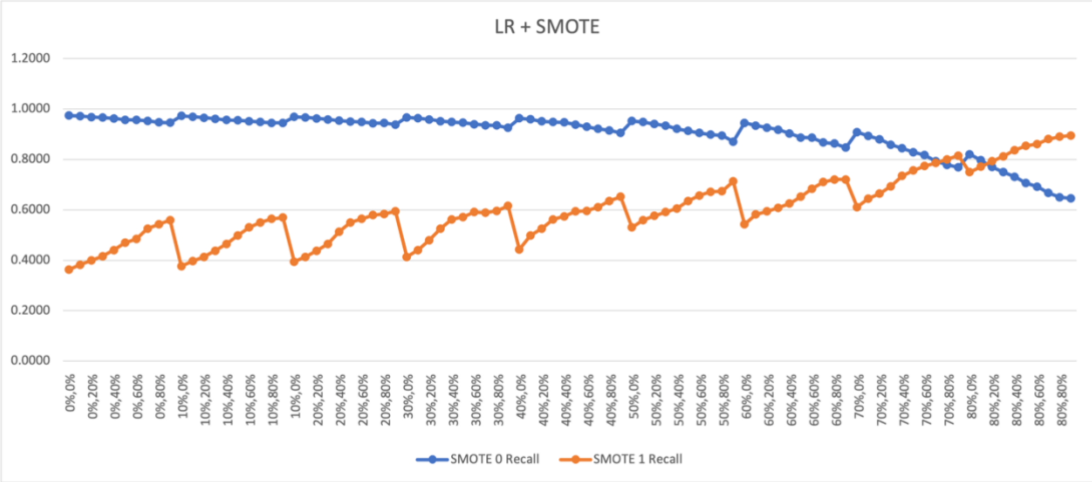


Figure 4.14: R0 and R1 of Logistic Regression combined with Standard SMOTE.

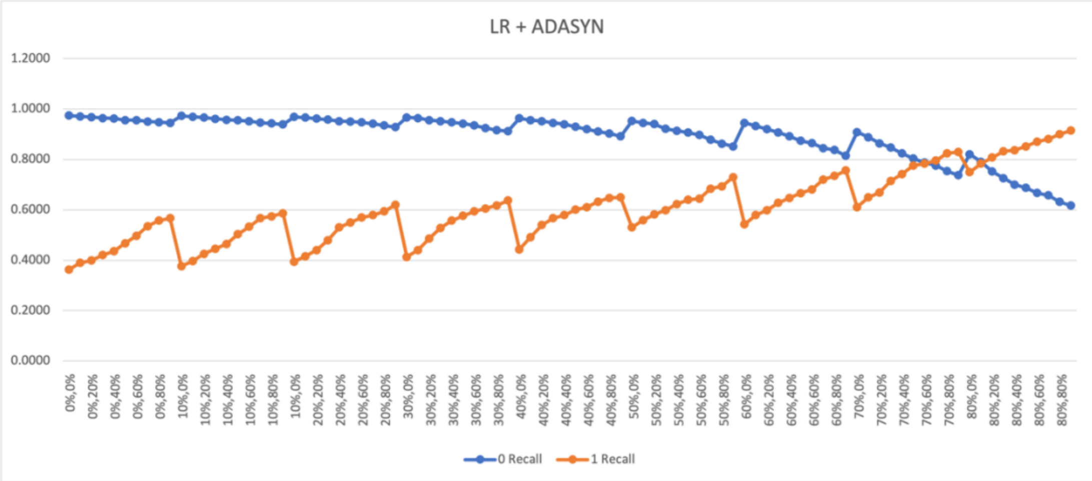


Figure 4.15: R0 and R1 of Logistic Regression combined with ADASYN.

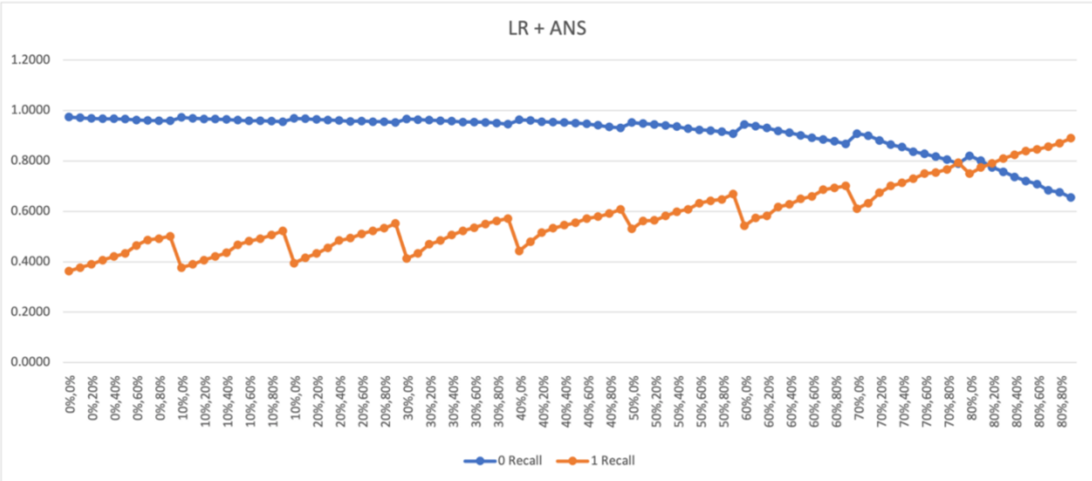


Figure 4.16: R0 and R1 of Logistic Regression combined with ANS.

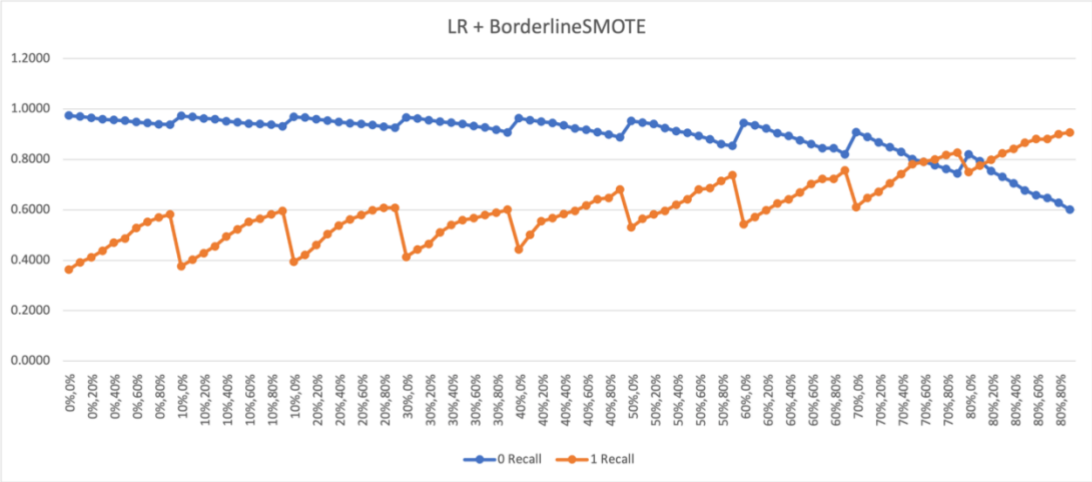


Figure 4.17: R0 and R1 of Logistic Regression combined with B-SMOTE.

With the exception of SVM-SMOTE, the remaining four smote variants exhibited a number of similar tendencies in terms of R0, R1, and accuracy. As the undersampling ratio decreases, accuracy and R0 decrease, but remain above 0.65 and 0.60, respectively. As for R0, the graph resembled upward stairs with eight dips. Something worth noting is that the dips occur when the undersampling ratio is at 0%. In contrast, when the undersampling ratio is 0%, R0 exhibited a downward stair-shaped line graph with eight abrupt increases. This indicates that undersampling may have a greater impact on the data set's recalls.

4.5.3 Naïve Bayes

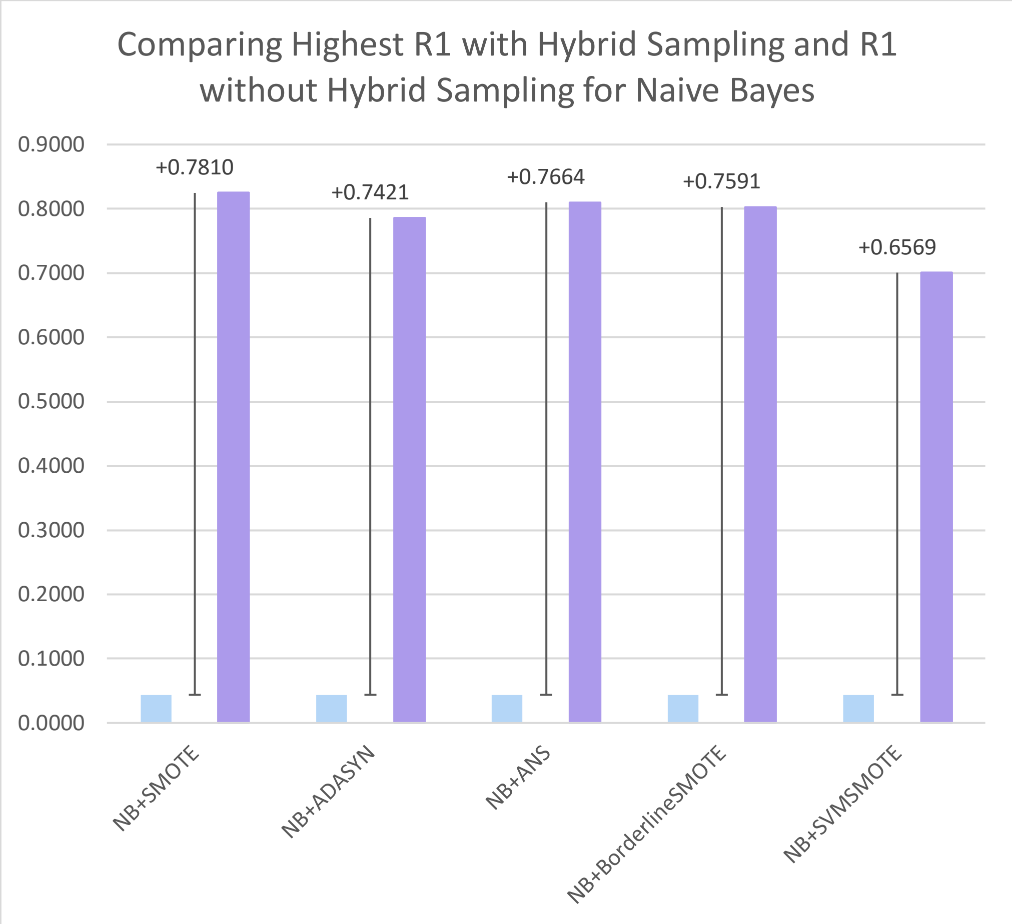


Figure 4.18: Comparing the highest R_1 with hybrid sampling (purple) and R_1 without sampling (blue) for the Naïve Bayes classifier.

Naive Bayes + hybrid sampling demonstrated the most remarkable gain in the detection rate of purchasing intention, as it has the lowest R1 score when no sampling methods are applied. With SVM-SMOTE excluded, the R1 improved by at least 0.74 on average. Utilising RUS, Standard SMOTE, and the Naive Bayes classifier produced the greatest R1 value of 0.8248. The combination of Naive Bayes + RUS + Standard SMOTE yielded the greatest improvement, 0.7810, as a result. The exception, SVM-SMOTE, also substantially increased the detection rate; however, its improvement rate of 0.6569 is slightly lower than the average improvement rate of the other Naive Bayes + hybrid sampling combinations.

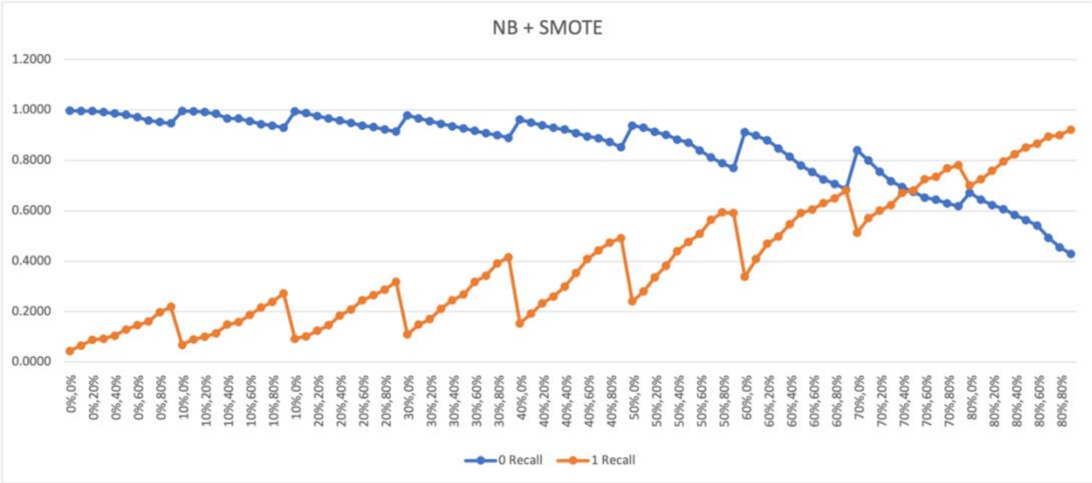


Figure 4.19: Majority and minority recall of Naïve Bayes combined with Standard SMOTE.

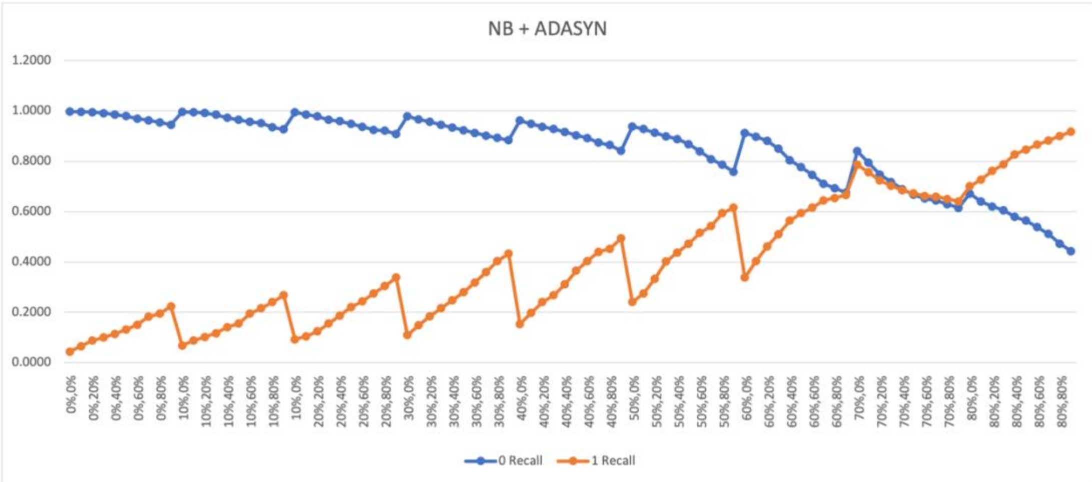


Figure 4.20: Majority and minority recall of Naïve Bayes combined with ADASYN.

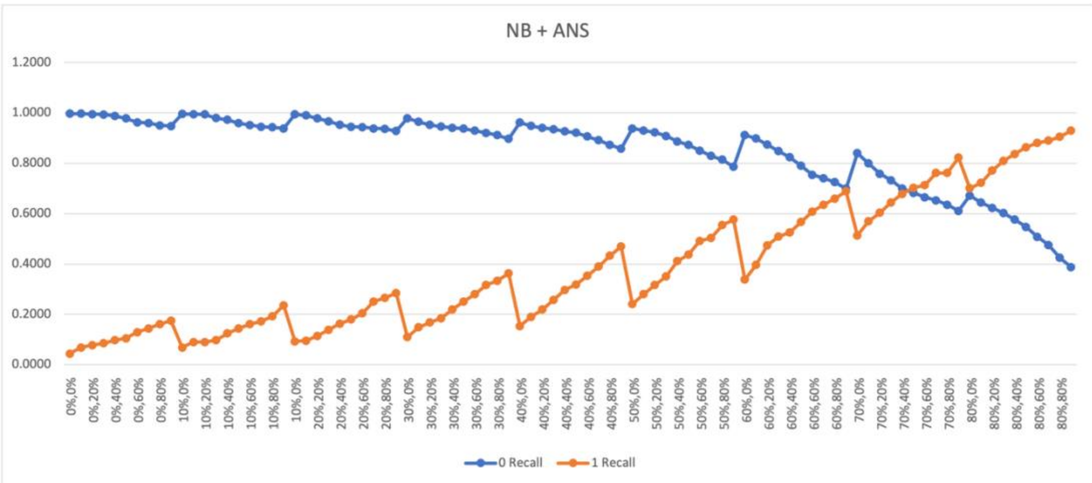


Figure 4.21: Majority and minority recall of Naïve Bayes combined with ANS.

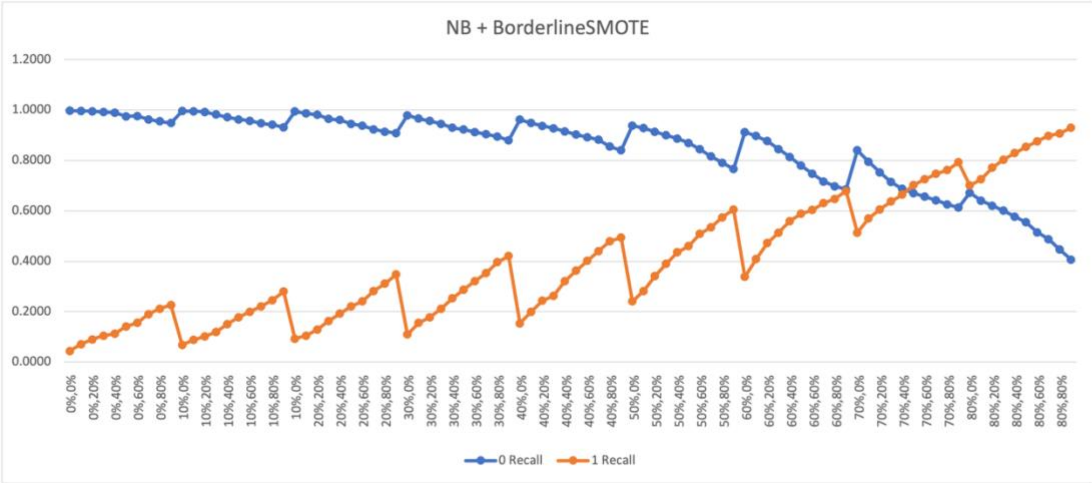


Figure 4.22: Majority and minority recall of Naïve Bayes combined with B-SMOTE.

The Naïve Bayes classifier demonstrated a more significant decline in accuracy and majority recall than the previously discussed classifiers. Within the range of 0.36, the accuracy decreased drastically. From 0.0438 to a maximum of 0.9294, the minority recall, in contrast, increased significantly. Similar to Logistic Regression + hybrid sampling, line graph R1 of Naive Bayes + hybrid sampling resembles a climbing staircase, whereas line graph R0 resembles a declining staircase. Nevertheless, the gradient of the line graph in Naïve Bayes is greater than that of the Logistic Regression graph. When the undersampling ratio is 0%, both R1 of Logistic Regression and Naïve Bayes experience a decrease, while R0 of both classifiers experiences a sudden increase. The Naïve Bayes classifier is the quickest of the five classifiers used in this project.

4.5.4 Random Forest

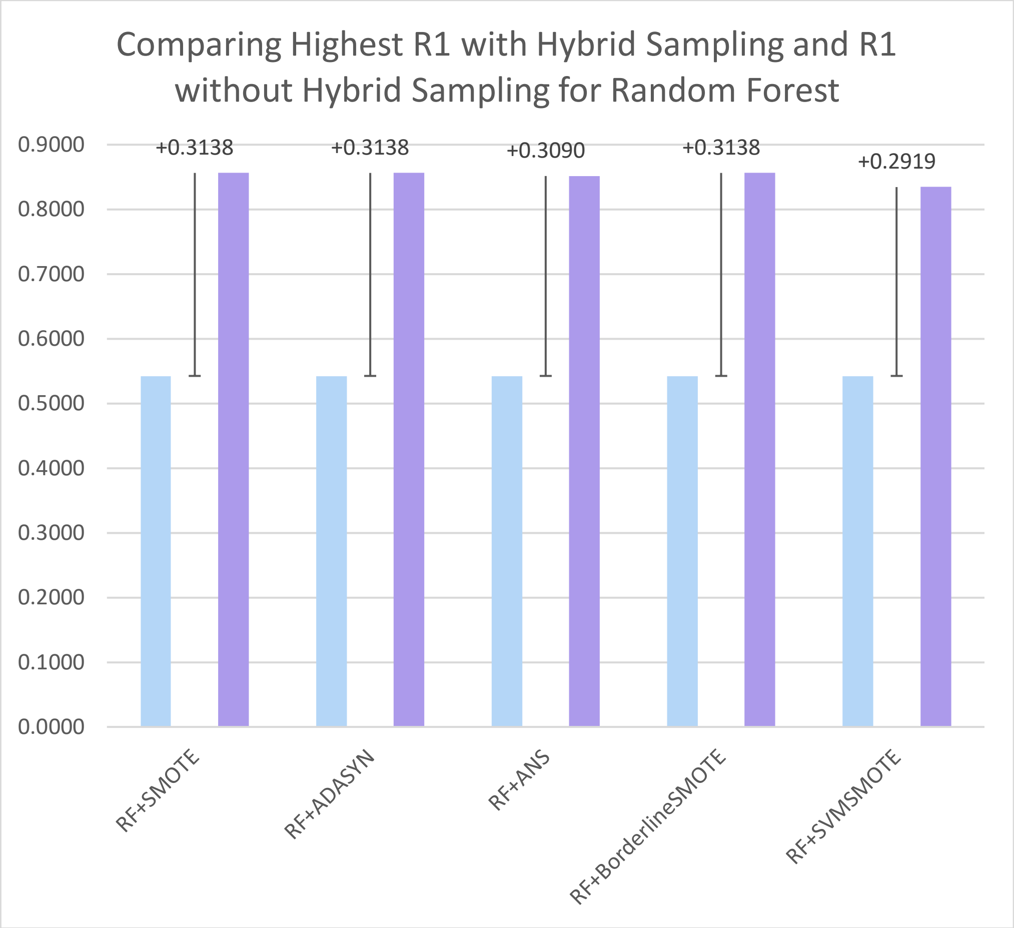


Figure 4.23: Comparing the highest R_1 with hybrid sampling (purple) and R_1 without sampling (blue) for the Random Forest classifier.

Random Forest combined with hybrid sampling generated the highest detection rates of all classifiers. This will be discussed in greater detail in Section 4.3. Overall, the Random Forest plus hybrid sampling combination increased R1 by 0.2919 to 0.3138. In contrast to other classifiers, the improvement R1 generated by SVM-SMOTE is near to that of the other oversamplers; therefore, it is no longer an anomaly.

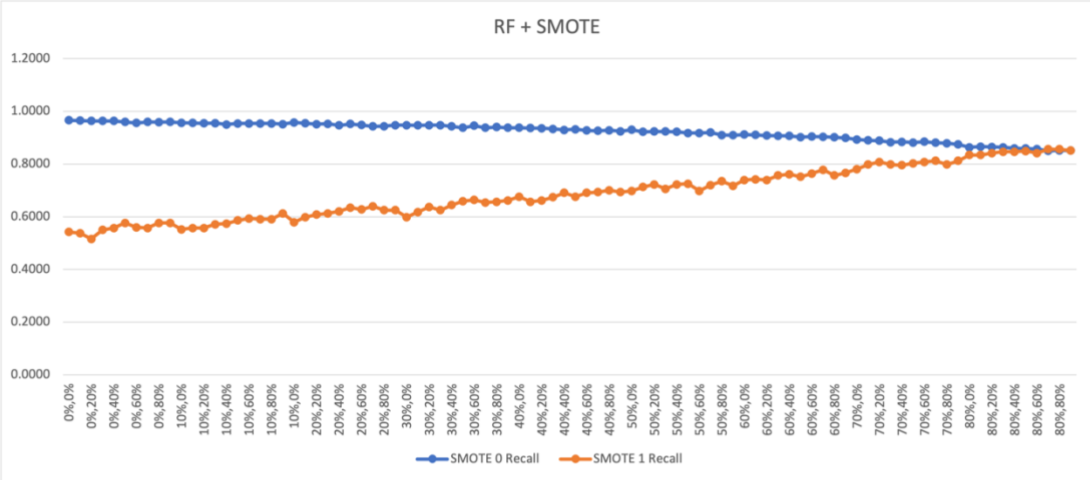


Figure 4.24: Majority and minority recall of Random Forest combined with Standard SMOTE.

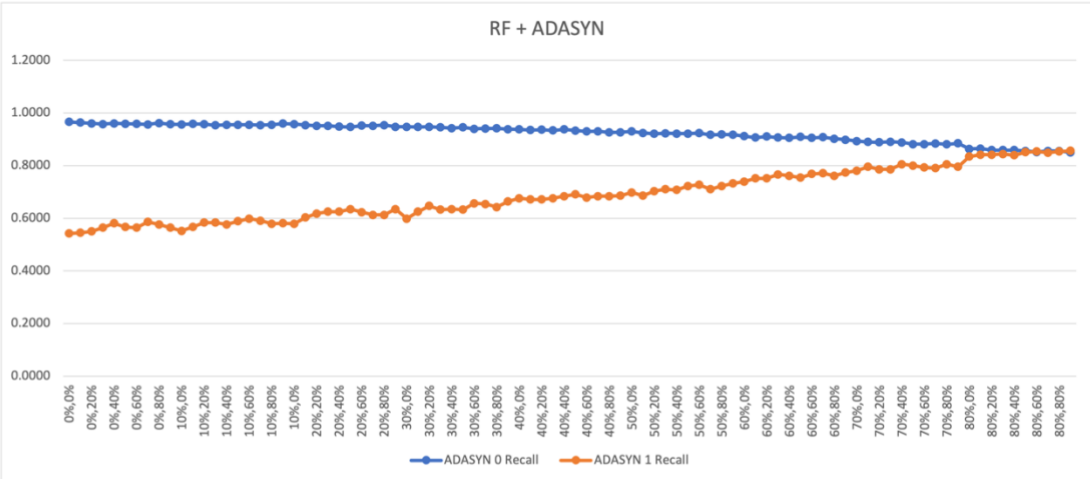


Figure 4.25: Majority and minority recall of Random Forest combined with ADASYN.

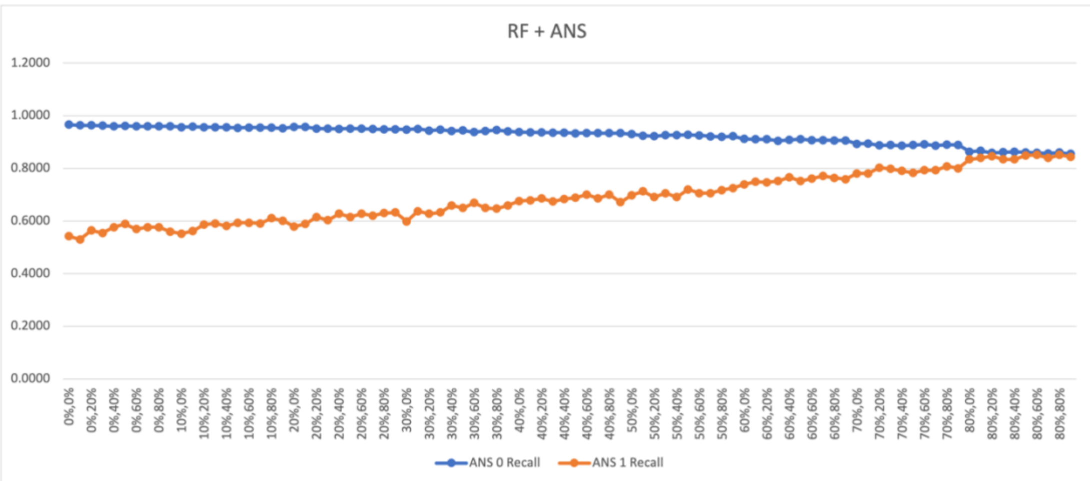


Figure 4.26: Majority and minority recall of Random Forest combined with ANS.

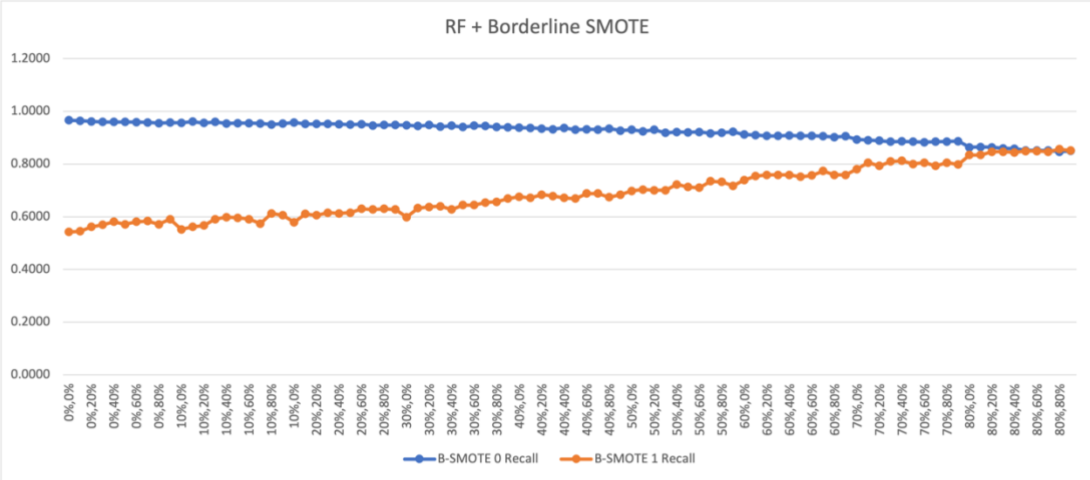


Figure 4.27: Majority and minority recall of Random Forest combined with B-SMOTE.

Similar to the Decision Tree, the line graphs for standard SMOTE, ADASYN, ANS and Borderline SMOTE converge. The accuracy and R0 follow a downward trend, whereas R1 follows an upward trend. However, the accuracy has demonstrated a gradual trend of decline within the range of 0.05%.

4.5.5 SVM

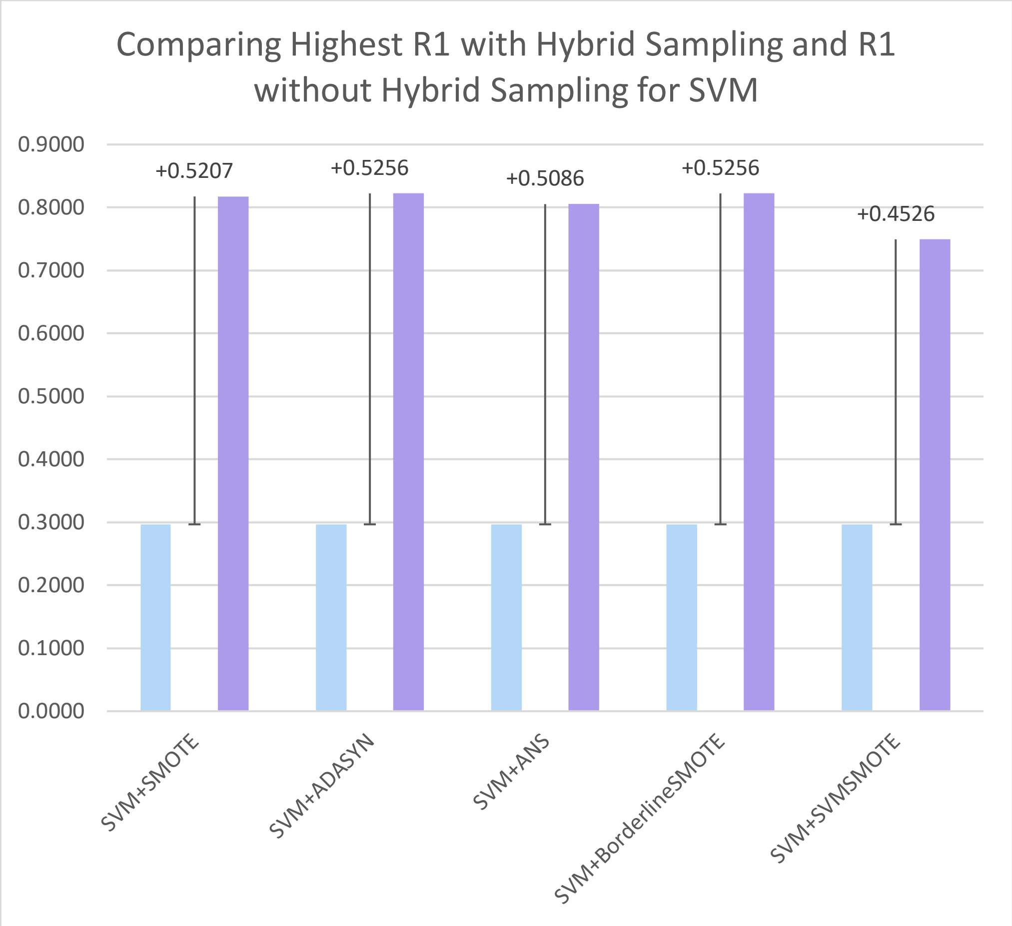


Figure 4.28: Comparing the highest R_1 with hybrid sampling (purple) and R_1 without sampling (blue) for the SVM classifier.

Similar to the earlier combinations, SVM + hybrid sampling significantly increased the detection rate. Except for SVM-SMOTE, SVM + hybrid sampling generally improves R1 by at least 0.50. Two combinations produce the most significant improvement in R1 (0.5256): SVM + RUS + ADASYN and SVM + RUS + B-SMOTE. In contrast to the Random Forest combination, SVM-SMOTE remains an outlier when combined with SVM and RUS, with a slightly reduced improvement rate of 0.4526.

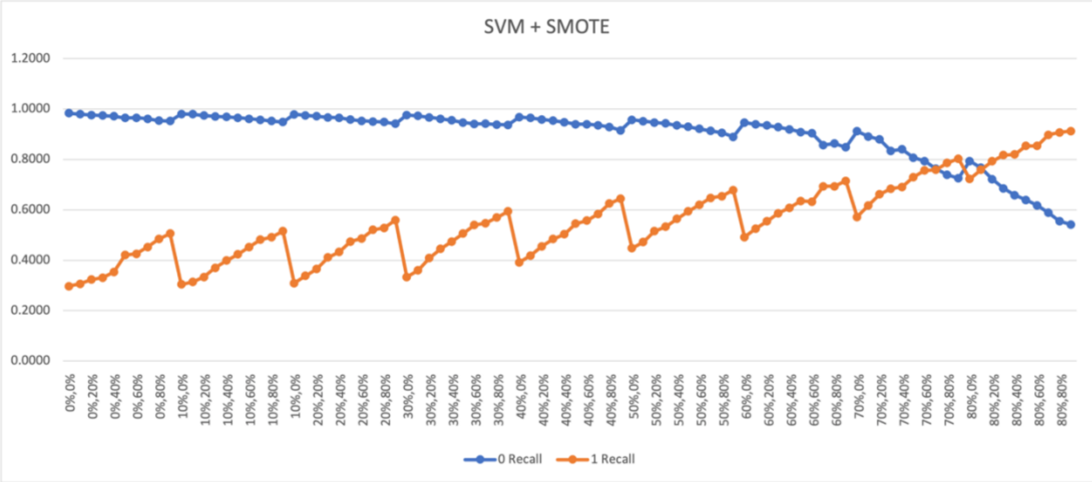


Figure 4.29: Majority and minority recall of SVM combined with Standard SMOTE.

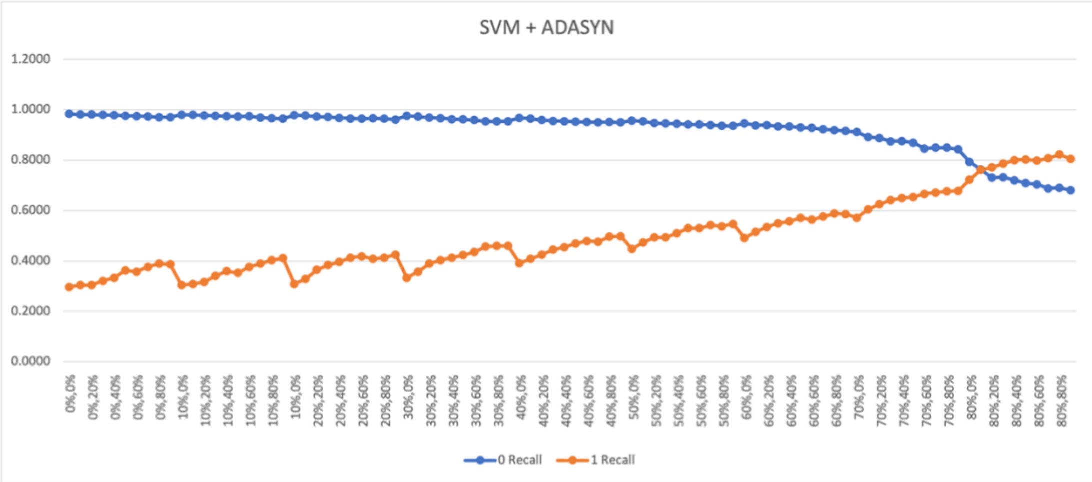


Figure 4.30: Majority and minority recall of SVM combined with ADASYN.

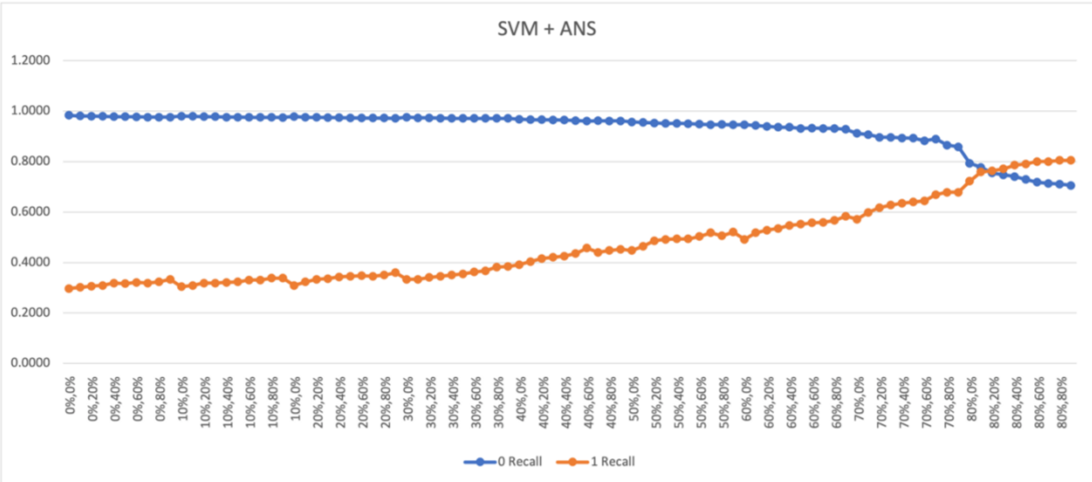


Figure 4.31: Majority and minority recall of SVM combined with ANS.

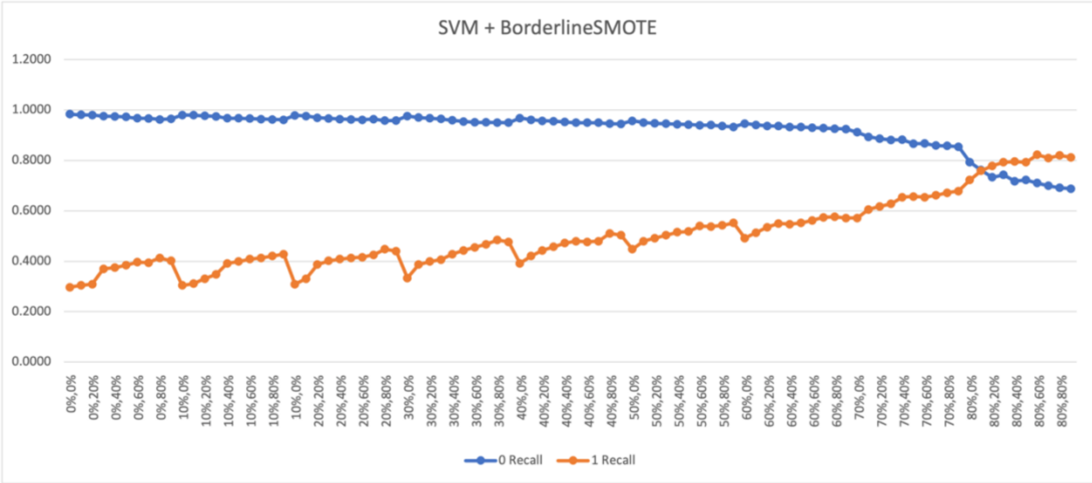


Figure 4.32: Majority and minority recall of SVM combined with B-SMOTE.

In overall, accuracy and R0 have demonstrated declining trends, while minority recall has a rising trend, excluding SVM SMOTE. The line graphs in the preceding section first converge and intersect at one or a few instances before diverging. The R1 line graphs of standard SMOTE, ADASYN, and Borderline SMOTE resemble a staircase out of the four combinations depicted in the graphs above. In contrast, RUS + SVM + SVM SMOTE produced a stair-shaped graph with a steeper gradient as compared to ADASYN and Borderline SMOTE.

4.5.6 Trend with SVM SMOTE

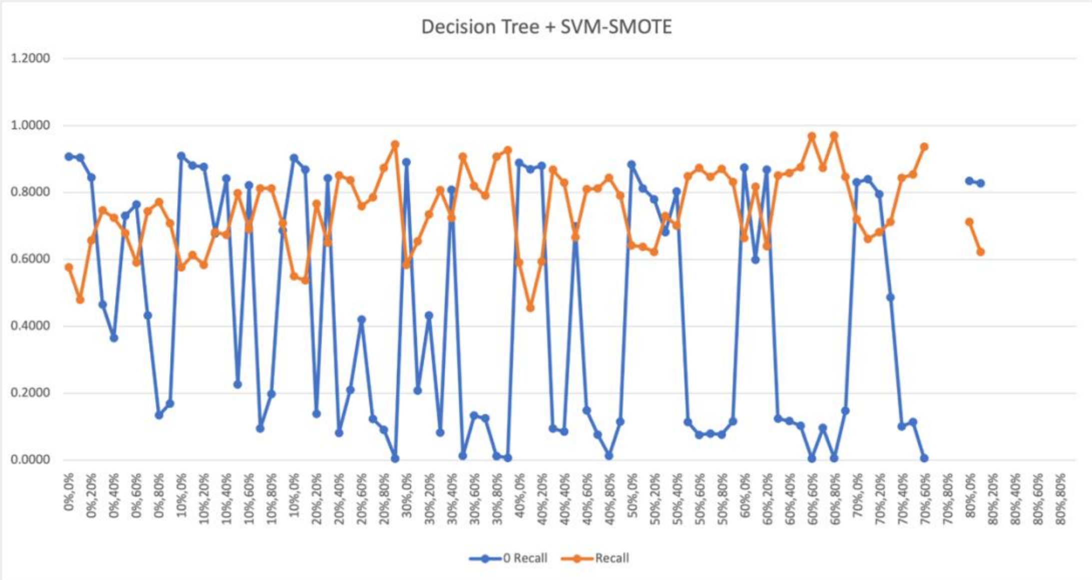


Figure 4.33: Majority and minority recall produced by applying SVM-SMOTE combined with Decision Tree.

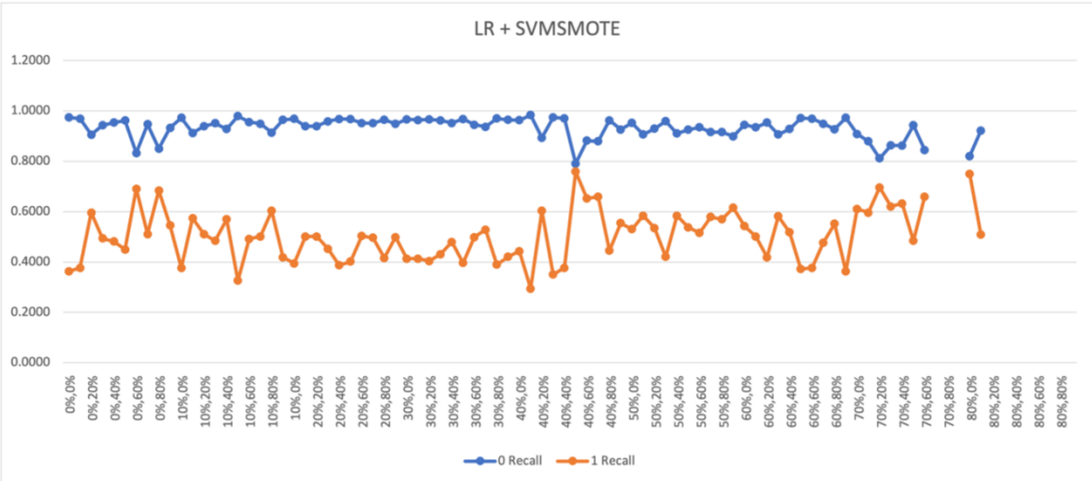


Figure 4.34: Majority and minority recall produced by applying SVM-SMOTE combined with Logistic Regression.

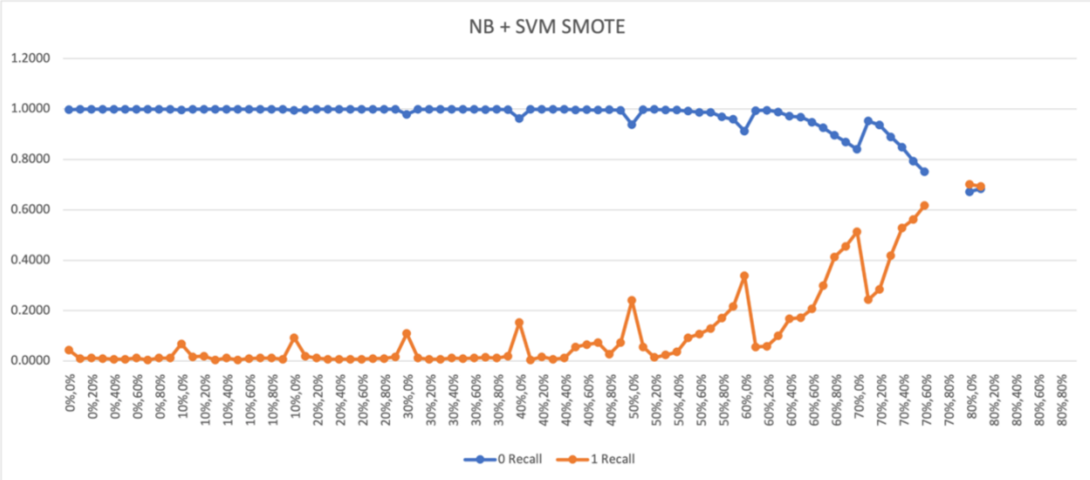


Figure 4.35: Majority and minority recall produced by applying SVM SMOTE combined with Naïve Bayes.

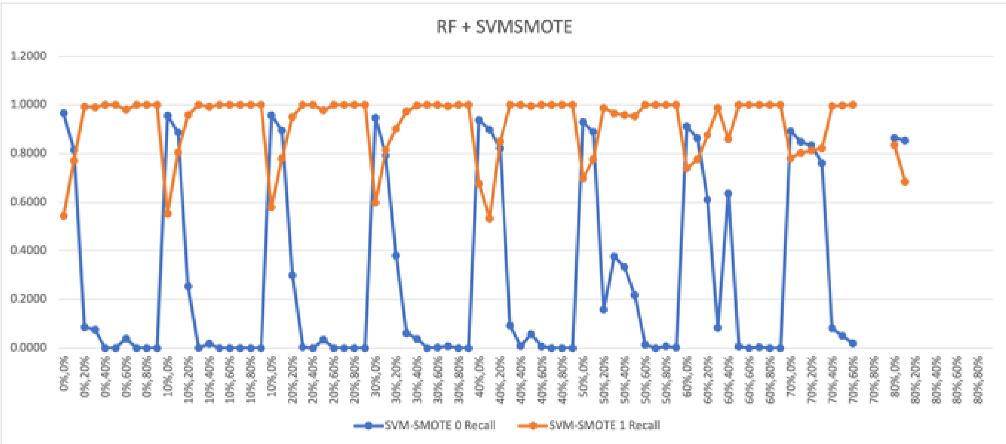


Figure 4.36: Majority and minority recall produced by applying SVM-SMOTE combined with Random Forest.

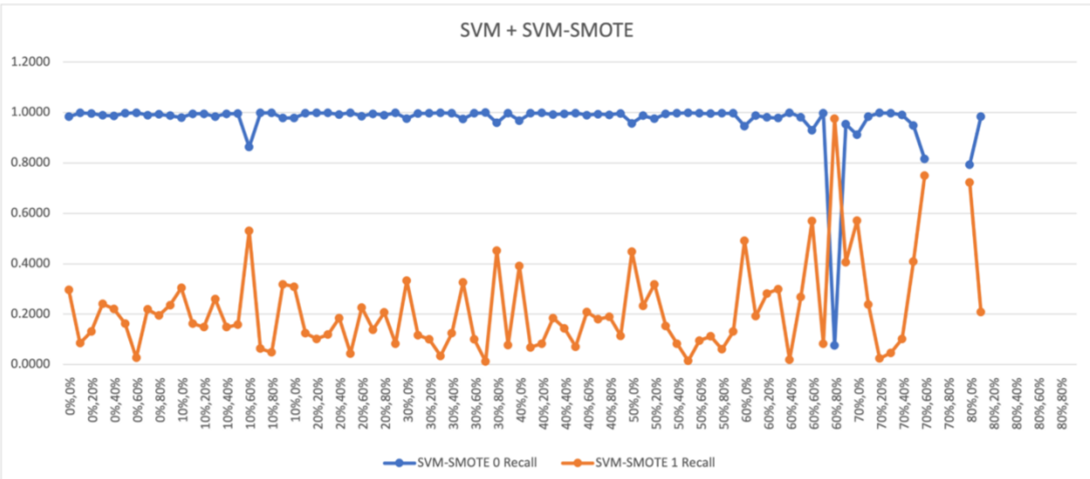


Figure 4.37: Majority and minority recall produced by applying SVM-SMOTE combined with SVM.

SVM-SMOTE is shown to generate unrecognisable random pattern graphs when used in conjunction with all five classifiers. The results of the investigations reveal a few commonalities. First, it biases when integrating with classifiers such as Decision Tree and Random Forest are identified. When combined with either of the two classifiers, the classifier tends to favour the minority class. To verify this assertion, 100% of the minority recall of the Decision Tree + SVM-SMOTE set is greater than 0.80, whereas the majority of the minority recall of the Random Forest + SVM-SMOTE set is greater than 0.80. In contrast, when paired with simpler classifiers such as Logistic Regression and Naïve Bayes, a result with a larger majority bias is observed. In support of this claim, when Logistic Regression is paired, a higher frequency of majority recall values greater than 0.80 is observed, whereas all minority recall values are below 0.80. Similar to Logistic Regression, the majority recall of Naïve Bayes is greater than 0.99 when paired with SVM SMOTE, while all minority recall is below 0.80.

4.6 Comparing The Best Classifier + Hybrid Sampling

Table 4.3: Comparison of The Best Classifier + Hybrid sampling set

(Notes: **UR** indicates **Undersampling ratio**, **OR** indicates **Oversampling ratio**, **R_0** indicates **Majority recall, R_1** indicates **Minority recall** and the bolded rows indicate the best results for the classifier in the data set.)

Classifier + Hybrid Sampling	UR	OR	R0	R1
RF + RUS + SMOTE	80%	180%	0.8521	0.8564
RF + RUS + ADASYN	80%	190%	0.8501	0.8564
RF + RUS + BSMOTE	80%	180%	0.8467	0.8564
RF + RUS + ANS	80%	180%	0.8603	0.8516
RF + RUS + SVM-SMOTE	80%	100%	0.8633	0.8345

Based on the line graphs, one or a few intersection instances are compared. The Best Classifier + Hybrid Sampling were rated by their R1 scores. When a set has a truce for the R1, R0 will be used to break the tie. If both R1 and R0 are tied, accuracy will be factored into the comparison. Lastly, if all three metrics are tied, the F1 value is used to determine the best-performing set. In the following section, the Classifier + Hybrid Sampling sets were ordered by their ranking.

4.6.1 RF + RUS +Standard SMOTE

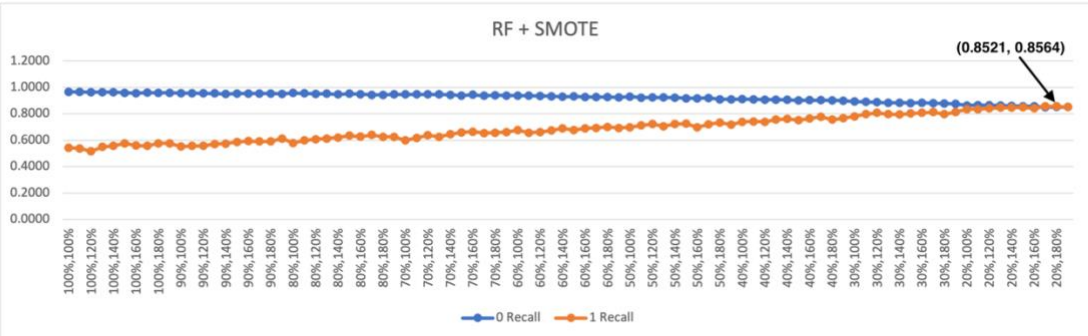


Figure 4.38: Majority and minority recall when Random Forest and Standard SMOTE are applied.

The R0 and R1 of the Random Forest and Standard SMOTE sets have generated a converging graph, as depicted in Figure 4.38. An intersection point (0.8521, 0.8564), where the R0 is 0.8521 and the R1 is 0.8564, is identified as the optimal result for this set. The undersampling rate and oversampling rate at which the classifier produced this result are both 80%.

4.6.2 RF + RUS + ADASYN

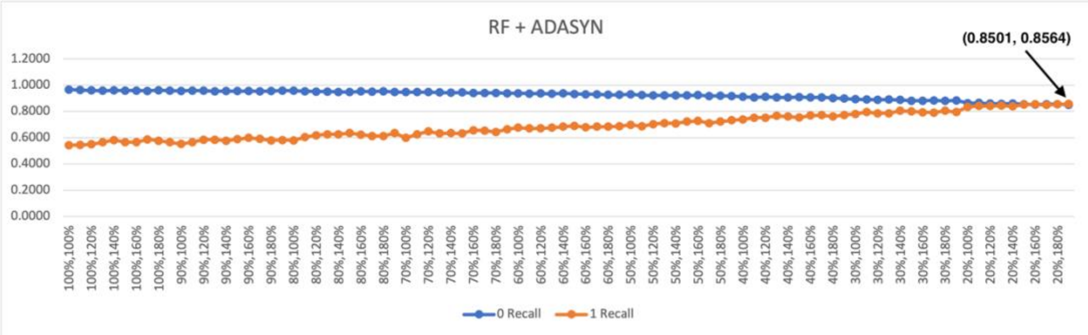


Figure 4.39: Majority and minority recall when Random Forest and ADASYN are applied.

The optimal result was obtained when the line graphs intersected at (0.8501, 0.8564), where the undersampling ratio is 80% and the oversampling ratio is 90%, according to Figure 4.39.

4.6.3 RF + RUS + B-SMOTE

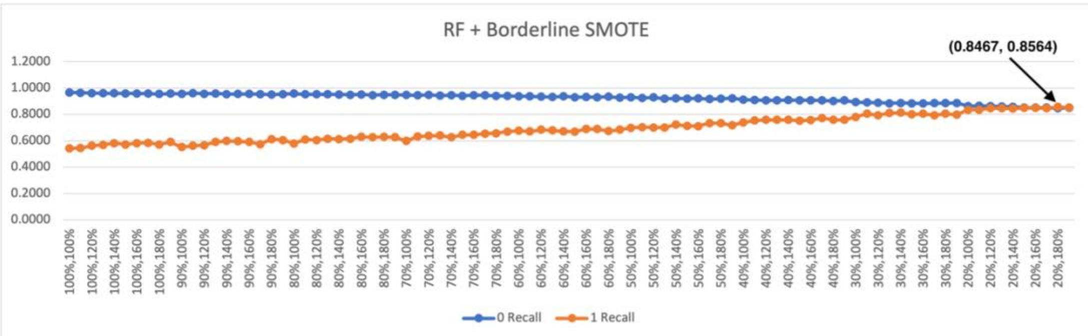


Figure 4.40: Majority and minority recall when Random Forest and B-SMOTE are applied.

In Figure 4.40, the intersection that occurred when both undersampling and oversampling ratios were 80% revealed the optimal detection rate. A majority recall of 0.8467 and a minority recall of 0.8564 are recorded at the line graph's intersection point.

4.6.4 RF + RUS + ANS

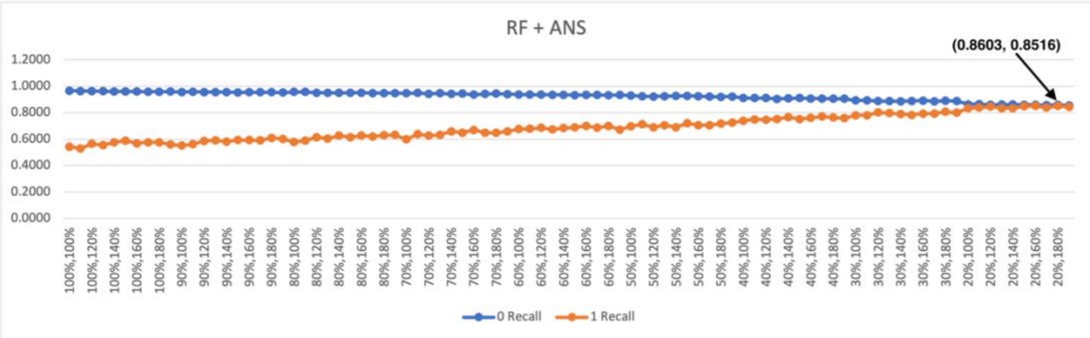


Figure 4.41: Majority and minority recall when Random Forest and ANS are applied.

With a majority recall of 0.8603 and a minority recall of 0.8516, the intersection at (0,8603, 0,8516) in Figure 4.41 exhibited the highest detection rate in its set. The point of intersection occurred when the undersampling and oversampling ratios were both 80%.

4.6.5 RF + RUS + SVM-SMOTE

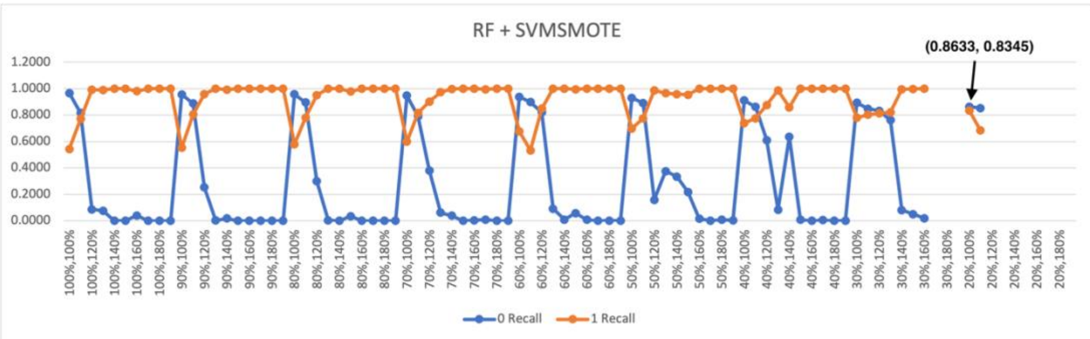


Figure 4.42: Majority and minority recall when Random Forest and SVM-SMOTE are applied.

In contrast to the other graphs, the recall values outputted by the Random Forest + SVM-SMOTE set resulted in a graph with a random pattern rather than a converging graph. According to Figure 4.42, even when the undersampling ratio is low, there are multiple sites of intersection in the graphs, which is not the case with other SMOTE variants (Standard SMOTE, ADASYN, ANS, and B-SMOTE). The optimal output is achieved, however, when the undersampling ratio is 80% and oversampling is not used. There is a majority recall of 0.8633 and a minority recall of 0.8345.

CHAPTER 5
CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

After conducting **2,011 experiments** using hybrid sampling with nine undersampling and ten oversampling ratios, the project has achieved its objectives.

Overall, hybrid sampling has significantly increased the purchase intention detection rate. **The best hybrid sampling technique** is Random Undersampling (80%) and Standard SMOTE (80%) with Random Forest, yielding a Recall of 0.8521 for the majority class and 0.8564 for the minority class.

Random Forest functions well with all hybrid sampling techniques compared to the other classifiers,. Random Forest with the hybrid sampling technique Random Undersampling + Standard SMOTE produces the finest results.

5.2 Recommendations

Applying feature selection to future projects could increase accuracy and clarity while decreasing computational complexity. According to a study by Singh and Jain (2019), the true positive rate (TPR) can be substantially increased by employing feature selection techniques such as filter and wrapper. In the paper, except for the Random Forest classifier, applying a filter or wrapper enhances the TPR of J48, AdaBoost, Naive Bayes, and PART classifiers. Another suggestion would be **to include algorithm fairness** within the project's scope. Since this project demonstrates that certain classifiers are susceptible to bias towards a particular class, addressing algorithm fairness would aid in illuminating the factors influencing the detection rate. Hasanin and Khoshgoftaar (2018) stated that RUS often leads to losing important information as it randomly eliminates patterns of the majority class. Consequently, **additional undersampling techniques can be incorporated into the experiments** by exploring more undersampler options available in the research field. Koziarski (2021) proposed an undersampling technique, Synthetic Majority Undersampling Technique (SMUTE), which has proven a viable alternative to RUS.

REFERENCES

Akbani, R., Kwek, S. and Jakowicz, N., 2004. Applying support vector machines to imbalanced datasets. In: Boulicaut, J., Esposito, F., Giannotti, F., and Pedreschi, D., 15th European conference on machine learning. Pisa, Italy 20-24 September 2004. Berlin: Springer-Verlag.

Akkaya, B. and Çolakoğlu, N., 2019. Comparison of Multi-class Classification Algorithms on Early Diagnosis of Heart Diseases. In: Kocadagli, O., Erkoc, A., Baser, B., Denizli, N.A. and Ekin, T., y-BIS 2019 Conference: ISBIS Young Business and Industrial Statisticians Workshop on Recent Advances in Data Science and Business Analytics, Istanbul, Turkey, 25-28 September 2019, Istanbul: Mimar Sinan Fine Arts University Publications.

Ali, H., Mohd Salleh, M.N., Hussain, K., Ahmad, A., Ullah, A., Muhammad, A., Naseem, R., Khan, M., 2019. A review on data preprocessing methods for class imbalance problem. International Journal of Engineering & Technology 8, 390–397. https://doi.org/10.14419/ijet.v8i3.29508

Baati K, Mohsil M. Real-Time Prediction of Online Shoppers' Purchasing Intention Using Random Forest. Artificial Intelligence Applications and Innovations. 2020 May 6;583:43–51. doi: 10.1007/978-3-030-49161-1_4. PMCID: PMC7256375.

Bell, L., McCloy, R., Butler, L. and Vogt, J., 2020. Frontiers | Motivational and affective factors underlying consumer dropout and transactional success in eCommerce: an overview. Frontiers in Psychology, 11(1546). https://doi.org/10.3389/fpsyg.2020.01546.

Bernhardt, G., 2022. Global eCommerce sales growth report for 2020–2025. Available at: <https://www.shopify.my/blog/global-ecommerce-sales#:~:text=Global%20ecommerce%20sales%20are%20expected> [Accessed 30 July 2022].

Chawla, N.V., Hall, L.O., Bowyer, K.W. and Kegelmeyer, W.P., 2011. SMOTE: synthetic minority oversampling technique. Journal of Artificial Intelligence Research, 16, pp.321- 357. https://doi.org/10.48550/arXiv.1106.1813.

Devi, D., Biswas, S.K., Purkayastha, B., 2020. A Review on Solution to Class Imbalance Problem: Undersampling Approaches. 2020 International Conference on Computational Performance Evaluation (ComPE). https://doi.org/10.1109/compe49325.2020.9200087

Dongre, S., 2017. Rare class problem in data mining: review. International Journal of Advanced Research in Computer Science. 8(7), pp.1102-1105. http://dx.doi.org/10.26483/ijarcs.v8i7.4530

Esmeli, R., Bader-El-Den, M. and Abdullahi, H., 2021. Towards early purchase intention prediction in online session based retailing systems. Electronic Markets. [e-journal] 31, pp.697–715. https://doi.org/10.1007/s12525-020-00448-x

Guo, H., Li, Y., Shang, J. Gu, M., Huang, Y., and Gong B., Learning from class-imbalanced data: review of methods and applications, Expert Systems with Applications, [e-journal] 73, pp.220-239. https://doi.org/10.1016/j.eswa.2016.12.035.

Han, H., Wang, W. and Mao, B., 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Huang, D., Zhang, X., Huang, G., Advances in Intelligent Computing. ICIC 2005. Beijing, China. 23-26 August 2005, Berlin: Springer.

Hasanin, T., Khoshgoftaar, T., 2018. The Effects of Random Undersampling with Simulated Class Imbalance for Big Data. 2018 IEEE International Conference on Information Reuse and Integration (IRI). https://doi.org/10.1109/iri.2018.00018

Hasanin, T., Khoshgoftaar, T.M., Leevy, J., Seliya, N., 2019. Investigating Random Undersampling and Feature Selection on Bioinformatics Big Data. 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService). https://doi.org/10.1109/bigdataservice.2019.00063

He, H., Bai, Y., Garcia E. A. and Li S., 2008. ADASYN: *Adaptive synthetic sampling approach for imbalanced learning.* In: IEEE (Institute of Electrical and Electronics Engineers), 2008 International joint conference on neural networks (IEEE World Congress on Computational Intelligence). Hong Kong, 01-08 June 2008. New York: IEEE.

IRP Commerce., 2022. Ecommerce Market Data and Ecommerce Benchmarks. [online] Available at: https://www.irpcommerce.com/en/gb/EcommerceMarketData.aspx.

Jain, V., Malviya, B. and Arya, S. 2021. An overview of electronic commerce (e-Commerce). *Journal of Contemporary Issues in Business and Government*, 27(3), pp. 665-670, doi:10.47750/cibg.2021.27.03.090.

Japkowicz, N. and Stephen, S., 2002. The class imbalance problem: a systematic study. *Intelligent Data Analysis*, 6(5), pp.429–449.

Johnson, J.M., Khoshgoftaar, T.M., 2020. The Effects of Data Sampling with Deep Learning and Highly Imbalanced Big Data. Information Systems Frontiers 22, 1113–1131. https://doi.org/10.1007/s10796-020-10022-7

Kek, Z.X., Ismail, S., Noorain, I.S. and Muhaime, N.A.D.A., 2021. *Comparisons of data mining classification algorithms for customers' shopping intention in e-commerce.* In: IEEE (Institute of Electrical and Electronics Engineers), 2021 2nd International conference on artificial intelligence and data sciences (AiDAS). Ipoh, Malaysia, 8-9 September 2021, Berlin: Springer.

Kim, R.Y., 2020. The impact of COVID-19 on consumers: preparing for digital sales. IEEE Engineering Management Review, 48(3), pp.212-218, https://doi.org/10.1109/EMR.2020.2990115.

Kotsiantis, S. & Kanellopoulos, D. and Pintelas, P.E., 2005. Handling imbalanced datasets: a review. GESTS International Transactions on Computer Science and Engineering. Available at: <https://www.researchgate.net/publication/228084509_Handling_imbalanced_datasets_A_review> [Accessed 19 August 2022]

Kovács, G., 2019. An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. Applied Soft Computing. 83. https://doi.org/10.1016/j.asoc.2019.105662

Kovács, G., 2019. Smote-variants: a Python implementation of 85 minority oversampling techniques. Neurocomputing. [e-journal] 366, pp. 352-354. https://doi.org/10.1016/j.neucom.2019.06.100.

Koziarski, M., 2020. Radial-Based Undersampling for imbalanced data classification. Pattern Recognition 102, 107262. https://doi.org/10.1016/j.patcog.2020.107262

Koziarski, M., 2021. CSMOUTE: Combined Synthetic Oversampling and Undersampling Technique for Imbalanced Data Classification. 2021 International Joint Conference on Neural Networks (IJCNN). https://doi.org/10.1109/ijcnn52387.2021.9533415

Kumar, P., Bhatnagar, R., Gaur, K. and Bhatnagar, A., 2021. Classification of imbalanced data: review of methods and applications. In: IOPscience, International conference on applied scientific computational intelligence using data science (ASCI 2020). Jaipur, India, 22-23 December 2020, Bristol: IOP.

Kurniawan, I., Abdussomad, A., Akbar, M.F., Saepudin, D.F., Azis, M.S. and Tabrani, M., 2020. Improving The Effectiveness of Classification Using The Data Level Approach and Feature Selection Techniques in Online Shoppers Purchasing Intention Prediction. *Journal of Physics: Conference Series*, 164. http://dx.doi.org/10.1088/1742-6596/1641/1/012083.

Lee, Y.O., Kim, Y.J., 2020. The Effect of Resampling on Data‐imbalanced Conditions for Prediction towards Nuclear Receptor Profiling Using Deep Learning. Molecular Informatics 39, 1900131. https://doi.org/10.1002/minf.201900131

Mahadevan, A., Arock, M., 2020. A class imbalance-aware review rating prediction using hybrid sampling and ensemble learning. Multimedia Tools and Applications 80, 6911–6938. https://doi.org/10.1007/s11042-020-10024-2

Mokryn, O., Bogina, V. and Kuflik, T., 2019. Will this session end with a purchase? Inferring current purchase intent of anonymous visitors, *Electronic Commerce Research and Applications*, [e-journal] 34, https://doi.org/10.1016/j.elerap.2019.100836.

Muda, M.A., Muhammad, Aswari, R.A. and Ahsan, M., 2020. Prediction of online shopper's purchasing intention using binary logistic regression, decision tree, and random forest. http://dx.doi.org/10.13140/RG.2.2.16567.55209.

Nguyen, H.M., Cooper, E.W. and Kamei, K., 2011. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*. 3(1), pp.4-21. https://doi.org/10.1504/IJKESDP.2011.039875.

Obiedat, R., 2020. A comparative study of different data mining algorithms with different oversampling techniques in predicting online shopper behavior. *International Journal of Advanced Trends in Computer Science and Engineering*. [e-journal] 9(3), pp.3575-3583. https://doi.org/10.30534/ijatcse/2020/164932020

Parkhimenka, U., Tatur, M. and Zhvakina, A., 2017. Heuristic approach to online purchase prediction based on internet store visitors classification using data mining methods. In: IEEE (International Electrical and Electronics Engineers), 2017 International conference on information and digital technologies (IDT), Zilina, Slovakia, 5-7 July 2017, New York: IEEE.

Peng, Z., Yan, F. and Li., 2019. Comparison of the different sampling techniques for imbalanced classification problems in machine learning. In: 2019 11th International conference on measuring technology and mechatronics automation (ICMTMA), Qiqihar, China, 28-29 April 2019. New York: IEEE.

Phillips, J. (2016). Ecommerce analytics: Analyze and improve the impact of your digital strategy. Old Tappan, New Jersey: Pearson Education, Inc.

Prachuabsupakij, W., 2015. CLUS: A new hybrid sampling classification for imbalanced data, 2015 12th International joint conference on computer science and software engineering (JCSSE). Prachinburi, Thailand, pp. 281-286, doi: 10.1109/JCSSE.2015.7219810. Songkhla: IEEE.

Prati, R.C., Batista, G.E.A.P.A. and Monard, M.C. Class imbalances versus class overlapping: an analysis of a learning system behavior. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., and Sossa, H. Third mexican international conference on artificial intelligence. Mexico City, Mexico, 26-30 April 2004. Berlin: Springer.

Shamsudin, H., Yusof, U.K., Jayalakshmi, A., Akmal Khalid, M.N., 2020. Combining oversampling and undersampling techniques for imbalanced classification: A comparative study using credit card fraudulent transaction dataset. 2020 IEEE 16th International Conference on Control & Automation (ICCA). https://doi.org/10.1109/icca51439.2020.9264517

Singh, A., Jain, A., 2019. Adaptive Credit Card Fraud Detection Techniques Based on Feature Selection Method. Advances in Intelligent Systems and Computing 167–178. https://doi.org/10.1007/978-981-13-6861-5_15

Siriseriwan, W. and Sinapiromsaran, K., 2017. Adaptive neighbor synthetic minority sampling technique under 1NN outcast handling. Songklanakarin Journal of Science and Technology. 39(5), pp.565-576. http://dx.doi.org/10.14456/sjst-psu.2017.70

Sridhar, S. and Sanagavarapu, S., 2021. Handling data imbalance in predictive maintenance for machines using SMOTE-based oversampling. In: IEEE (International Electrical and Electronics Engineers), 2021 13th International conference on computational intelligence and communication networks (CICN), Lima, Peru, 22-23 September 2021, New York: IEEE.

Sun, Y.M., Wong, A. and Kamel, M.S. 2009. Classification of imbalanced data: a review. International Journal of Pattern Recognition and Artificial Intelligence, 23(04), pp. 687–719. https://doi.org/10.1142/s0218001409007326

Tantithamthavorn, C., Hassan, A.E., Matsumoto, K., 2020. The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models. IEEE Transactions on Software Engineering 46, 1200–1219. https://doi.org/10.1109/tse.2018.2876537

Vuttipittayamongkol, P., Elyan, E. and Petrovski, A. (2021). On the class overlap problem in imbalanced data classification. *Knowledge-Based Systems*, 212. https://doi.org/10.1016/j.knosys.2020.106631

Wang, H., 2008. Combination approach of SMOTE and biased-SVM for imbalanced datasets. In: IEEE (International Electrical and Electronics Engineers), 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1-8 June 2008, New York: IEEE.

Weiss, G. M. and Provost, F. 2003. Learning when training data are costly: The effect of class distribution on tree induction. Journal of Artificial Intelligence Research, [e-journal] 19(2003), pp.315–354. https://doi.org/10.1613/jair.1199

Weiss G. M., 2004. Mining with rarity: a unifying framework. ACM SIGKDD Explorations Newsletter, [e-journal] 6(1), pp. 7-19. https://doi.org/10.1145/1007730.1007734

Wu, G. and Chang, E.Y., 2003. Class-boundary alignment for imbalanced dataset learning. ICML 2003 Workshop on learning from imbalanced data sets, Washington, DC, 21-24 August 2003. Washington: Morgan Kaufmann.

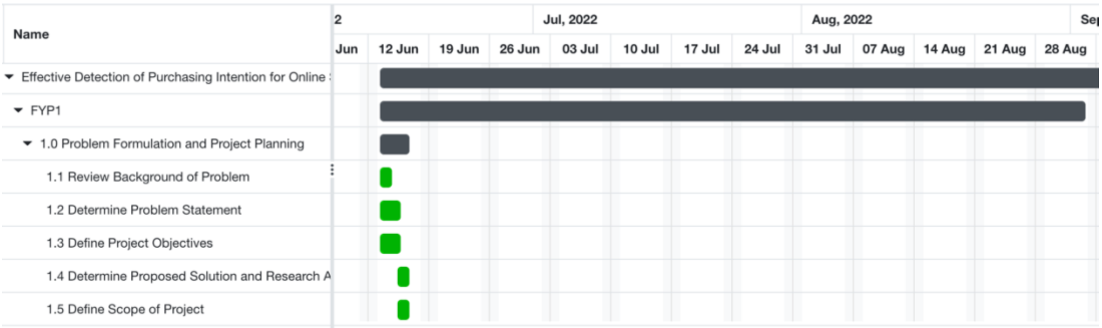
Xiao, J., Wang, Y., Chen, J., Xie, L., Huang, J., 2021. Impact of resampling methods and classification models on the imbalanced credit scoring problems. Information Sciences 569, 508–526. https://doi.org/10.1016/j.ins.2021.05.029

Zuech, R., Hancock, J., Khoshgoftaar, T.M., 2021. Detecting web attacks using random undersampling and ensemble learners. Journal of Big Data 8. https://doi.org/10.1186/s40537-021-00460-8

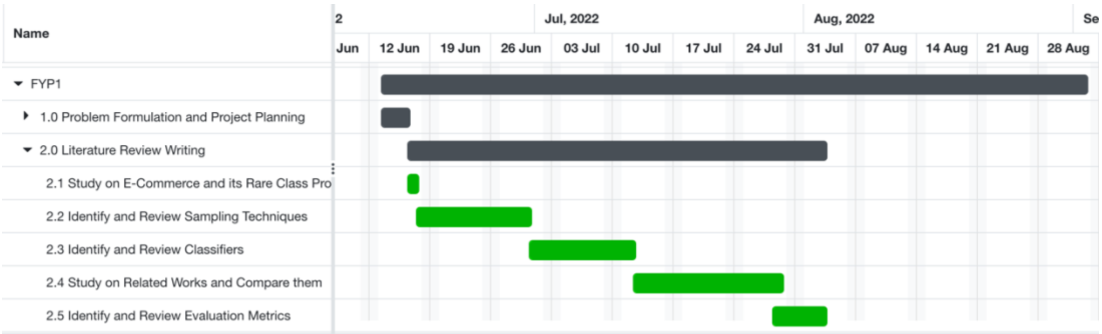
APPENDICES

APPENDIX A: Detailed Gantt Chart for FYP 1

i. Problem Formulation and Project Planning



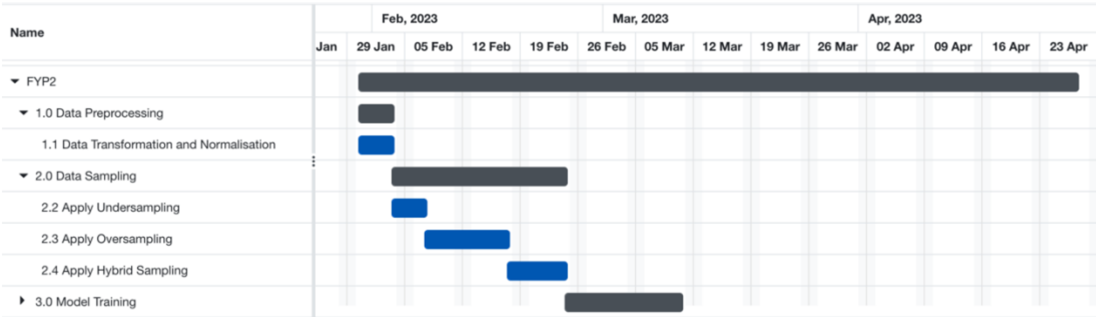
ii. Literature Review Writing



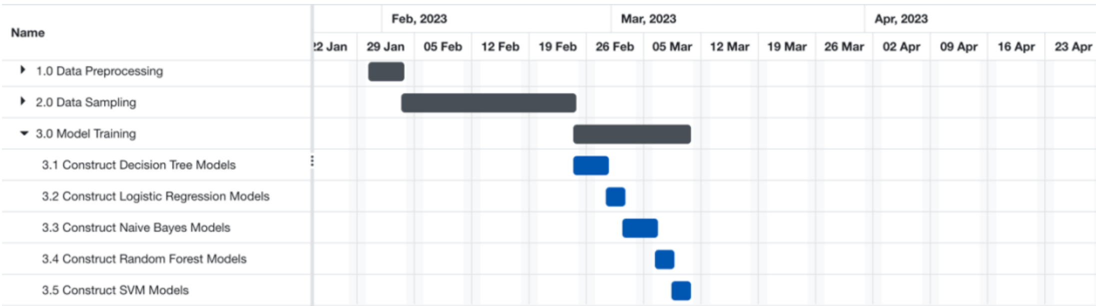
iii. Methodology Writing, Prototyping and Improvisation on FYP 1

APPENDIX B: Detailed Gantt Chart for FYP 2

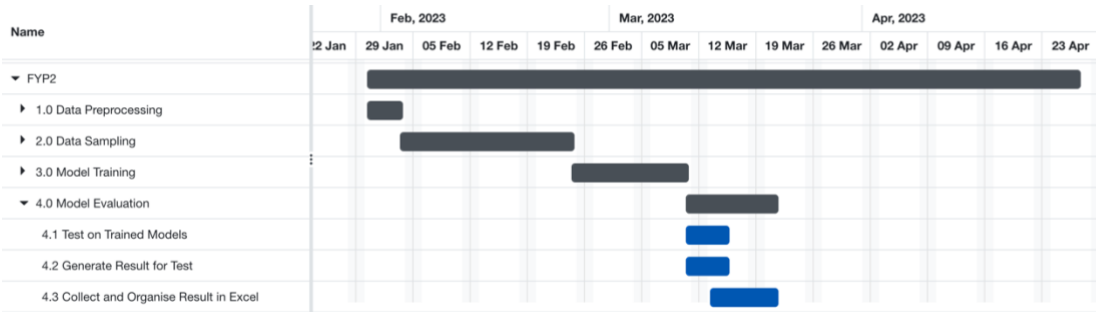
i. Data Pre-processing and Data Sampling



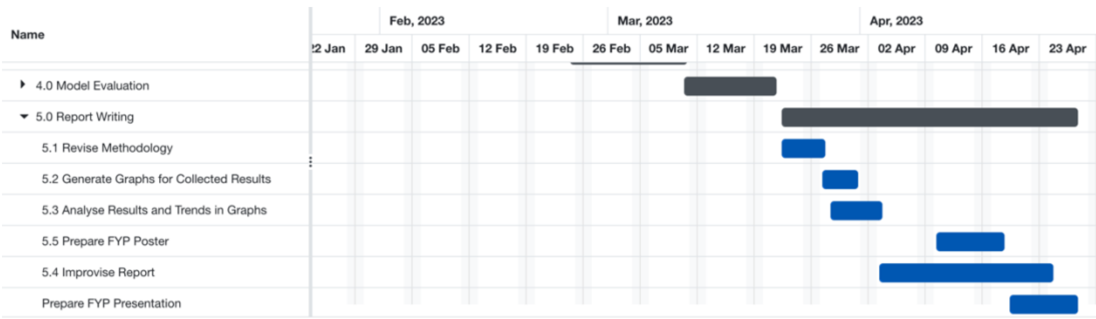
ii. Model Training



iii. Model Evaluation



iv. Report Writing



APPENDIX C: Undersampling Performance Metrics

	Decision Tree				Logistic Regression				Naïve Bayes				Random Forest				SVM			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
100%	0.8524	0.9075	0.5766	0.5656	0.8723	0.9742	0.3625	0.5656	0.8382	0.9971	0.0438	0.0828	0.8958	0.9664	0.5426	0.6344	0.8690	0.9835	0.2968	0.4303
110%	0.8540	0.9095	0.5766	0.5683	0.8739	0.9732	0.3771	0.4992	0.8414	0.9961	0.0681	0.1253	0.8893	0.9567	0.5523	0.6245	0.8678	0.9805	0.3041	0.4340
120%	0.8447	0.9036	0.5499	0.5413	0.8731	0.9689	0.3942	0.5086	0.8443	0.9946	0.0925	0.1652	0.8946	0.9577	0.5791	0.6467	0.8674	0.9791	0.3090	0.4372
130%	0.8402	0.8915	0.5839	0.5492	0.8747	0.9669	0.4136	0.5239	0.8341	0.9791	0.1095	0.5239	0.8897	0.9479	0.5985	0.6440	0.8690	0.9762	0.3333	0.4590
140%	0.8394	0.8891	0.5912	0.5510	0.8767	0.9635	0.4428	0.5449	0.8273	0.9620	0.1533	0.2283	0.8946	0.9382	0.6764	0.6814	0.8723	0.9684	0.3917	0.5055
150%	0.8435	0.8837	0.6423	0.5777	0.8824	0.9528	0.5304	0.6006	0.8224	0.9387	0.2409	0.3113	0.8917	0.9304	0.6983	0.6825	0.8723	0.9572	0.4477	0.5388
160%	0.8394	0.8745	0.6642	0.4477	0.8783	0.9455	0.5426	0.5979	0.8171	0.9129	0.3382	0.3813	0.8832	0.9119	0.7397	0.6786	0.8706	0.9465	0.4915	0.5588
170%	0.8122	0.8307	0.7202	0.5611	0.8593	0.9090	0.6107	0.5913	0.7859	0.8404	0.7859	0.4442	0.8747	0.8934	0.7810	0.6751	0.8552	0.9119	0.5718	0.5683
180%	0.8143	0.8345	0.7129	0.5613	0.8090	0.8209	0.7494	0.5667	0.6764	0.6715	0.7007	0.4192	0.8585	0.8633	0.8345	0.6628	0.7818	0.7937	0.7226	0.5247
190%	0.8524	0.9075	0.5766	0.5656	0.8723	0.9742	0.3625	0.5656	0.8382	0.9971	0.0438	0.0828	0.8958	0.9664	0.5426	0.6344	0.8690	0.9835	0.2968	0.4303

APPENDIX D: Oversampling Performance Metrics

i. SMOTE

	SMOTE																			
	Decision Tree				Logistic Regression				Naïve Bayes				Random Forest				SVM			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
100%	0.8447	0.9075	0.5499	0.5413	0.8723	0.9742	0.3625	0.4861	0.8382	0.9971	0.0438	0.0828	0.8958	0.9664	0.5426	0.6344	0.8690	0.9835	0.2968	0.4303
110%	0.8451	0.9080	0.5499	0.5420	0.8739	0.9723	0.3820	0.5024	0.8414	0.9966	0.0657	0.1213	0.8938	0.9650	0.5377	0.6278	0.8678	0.9800	0.3066	0.4360
120%	0.8487	0.9192	0.5499	0.5479	0.8735	0.9684	0.3990	0.5125	0.8447	0.9961	0.0876	0.1582	0.8893	0.9640	0.5158	0.6083	0.8674	0.9762	0.3236	0.4486
130%	0.8451	0.9090	0.5766	0.5537	0.8747	0.9664	0.4161	0.5253	0.8423	0.9922	0.0925	0.1634	0.8950	0.9640	0.5499	0.6357	0.8670	0.9742	0.3309	0.4533
140%	0.8512	0.9192	0.5864	0.5677	0.8759	0.9630	0.4404	0.5419	0.8394	0.9864	0.1046	0.1784	0.8962	0.9640	0.5572	0.6415	0.8686	0.9718	0.3528	0.4723
150%	0.8435	0.9182	0.5669	0.5469	0.8763	0.9577	0.4696	0.5586	0.8398	0.9820	0.1290	0.2116	0.8958	0.9596	0.5766	0.6484	0.8739	0.9645	0.4209	0.5266
160%	0.8455	0.9192	0.5596	0.5470	0.8779	0.9567	0.4842	0.5694	0.8337	0.9713	0.1460	0.2264	0.8901	0.9562	0.5596	0.6293	0.8755	0.9655	0.4258	0.5327
170%	0.8479	0.9124	0.5888	0.5634	0.8820	0.9533	0.5255	0.5975	0.8256	0.9586	0.1606	0.2349	0.8929	0.9601	0.5572	0.6343	0.8763	0.9611	0.4526	0.5495
180%	0.8573	0.9056	0.6034	0.5849	0.8800	0.9474	0.5426	0.6011	0.8268	0.9528	0.1971	0.2750	0.8954	0.9591	0.5766	0.6475	0.8759	0.9543	0.4842	0.5653
190%	0.8532	0.9260	0.5620	0.5607	0.8816	0.9460	0.5596	0.6117	0.8256	0.9470	0.2190	0.2951	0.8958	0.9596	0.5766	0.6484	0.8783	0.9528	0.5061	0.5810

ii. ADASYN

	ADASYN																			
	Decision Tree				Logistic Regression				Naïve Bayes				Random Forest				SVM			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
100%	0.8524	0.9075	0.5766	0.5656	0.8723	0.9742	0.3625	0.5656	0.8382	0.9971	0.0438	0.0828	0.8958	0.9664	0.5426	0.6344	0.8690	0.9835	0.2968	0.4303
110%	0.8504	0.9168	0.5182	0.5358	0.8739	0.9708	0.3893	0.5358	0.8414	0.9966	0.0657	0.1213	0.8938	0.9635	0.5450	0.6310	0.8682	0.9810	0.3041	0.4348
120%	0.8431	0.9007	0.5547	0.5409	0.8731	0.9679	0.3990	0.5409	0.8439	0.9951	0.0876	0.1575	0.8921	0.9606	0.5499	0.6295	0.8682	0.9810	0.3041	0.4348
130%	0.8491	0.9105	0.5426	0.5452	0.8735	0.9640	0.4209	0.5452	0.8427	0.9912	0.0998	0.1745	0.8921	0.9577	0.5645	0.6356	0.8702	0.9800	0.3212	0.4521
140%	0.8569	0.9182	0.5499	0.5615	0.8751	0.9630	0.4355	0.5615	0.8406	0.9859	0.1144	0.1930	0.8974	0.9606	0.5815	0.6539	0.8715	0.9791	0.3333	0.4636
150%	0.8532	0.9051	0.5937	0.5741	0.8743	0.9557	0.4672	0.5741	0.8382	0.9796	0.1314	0.2130	0.8938	0.9591	0.5669	0.6401	0.8739	0.9762	0.3625	0.4893
160%	0.8491	0.8998	0.5961	0.5684	0.8792	0.9557	0.4964	0.5684	0.8333	0.9698	0.1509	0.2318	0.8929	0.9586	0.5645	0.6374	0.8723	0.9752	0.3577	0.4828
170%	0.8390	0.9002	0.5328	0.5246	0.8812	0.9504	0.5353	0.5246	0.8329	0.9630	0.1825	0.2669	0.8950	0.9567	0.5864	0.6505	0.8743	0.9737	0.3771	0.5000
180%	0.8552	0.9182	0.5401	0.5543	0.8828	0.9479	0.5572	0.5543	0.8281	0.9547	0.1946	0.2740	0.8978	0.9620	0.5766	0.6529	0.8735	0.9703	0.3893	0.5063
190%	0.8443	0.9080	0.5255	0.5294	0.8820	0.9450	0.5669	0.5294	0.8248	0.9450	0.2238	0.2987	0.8917	0.9572	0.5645	0.6347	0.8735	0.9708	0.3869	0.5048

iii. ANS

	ANS																			
	Decision Tree				Logistic Regression				Naïve Bayes				Random Forest				SVM			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
100%	0.8524	0.9075	0.5766	0.5656	0.8723	0.9742	0.3625	0.4861	0.8382	0.9971	0.0438	0.0828	0.8958	0.9664	0.5426	0.6344	0.8690	0.9835	0.2968	0.4303
110%	0.8516	0.9144	0.5377	0.5470	0.8727	0.9718	0.3771	0.4968	0.8423	0.9971	0.0681	0.1258	0.8917	0.9640	0.5304	0.6202	0.8678	0.9810	0.3017	0.4321
120%	0.8577	0.9158	0.5669	0.5704	0.8723	0.9689	0.3893	0.5039	0.8427	0.9956	0.0779	0.1416	0.8974	0.9640	0.5645	0.6471	0.8682	0.9805	0.3066	0.4367
130%	0.8467	0.9056	0.5523	0.5457	0.8739	0.9674	0.4063	0.5178	0.8427	0.9942	0.0852	0.1528	0.8946	0.9625	0.5547	0.6369	0.8678	0.9796	0.3090	0.4379
140%	0.8625	0.9158	0.5961	0.5911	0.8763	0.9674	0.4209	0.5315	0.8398	0.9883	0.0973	0.1684	0.8962	0.9601	0.5766	0.6493	0.8686	0.9786	0.3187	0.4471
150%	0.8435	0.9012	0.5547	0.5416	0.8775	0.9664	0.4331	0.5410	0.8325	0.9781	0.1046	0.1723	0.8998	0.9620	0.5888	0.6621	0.8678	0.9781	0.3163	0.4437
160%	0.8556	0.9075	0.5961	0.5792	0.8796	0.9625	0.4647	0.5626	0.8236	0.9625	0.1290	0.1959	0.8950	0.9601	0.5693	0.6437	0.8682	0.9776	0.3212	0.4482
170%	0.8439	0.9027	0.5499	0.5400	0.8816	0.9606	0.4866	0.5780	0.8240	0.9601	0.1436	0.2138	0.8958	0.9596	0.5766	0.6484	0.8670	0.9766	0.3187	0.4441
180%	0.8577	0.9134	0.5791	0.5756	0.8820	0.9601	0.4915	0.5813	0.8187	0.9504	0.1606	0.2280	0.8958	0.9596	0.5766	0.6484	0.8678	0.9766	0.3236	0.4493
190%	0.8483	0.9158	0.5109	0.5290	0.8828	0.9591	0.5012	0.5877	0.8191	0.9479	0.1752	0.2441	0.8938	0.9606	0.5596	0.6371	0.8690	0.9762	0.3333	0.4590

iv. B-SMOTE

	B-SMOTE																			
	Decision Tree				Logistic Regression				Naïve Bayes				Random Forest				SVM			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
100%	0.8524	0.9075	0.5766	0.5656	0.8723	0.9742	0.3625	0.4861	0.8382	0.9971	0.0438	0.0828	0.8958	0.9664	0.5426	0.6344	0.8690	0.9835	0.2968	0.4303
110%	0.8520	0.9061	0.5815	0.5670	0.8739	0.9703	0.3917	0.5087	0.8423	0.9966	0.0706	0.1298	0.8946	0.9645	0.5450	0.6328	0.8682	0.9810	0.3041	0.4348
120%	0.8564	0.9109	0.5839	0.5755	0.8731	0.9655	0.4112	0.5192	0.8447	0.9956	0.0900	0.1619	0.8950	0.9616	0.5620	0.6408	0.8686	0.9805	0.3090	0.4394
130%	0.8423	0.9075	0.5158	0.5215	0.8727	0.9596	0.4380	0.5341	0.8443	0.9922	0.1046	0.1830	0.8950	0.9601	0.5693	0.6437	0.8747	0.9757	0.3698	0.4959
140%	0.8508	0.9129	0.5401	0.5468	0.8755	0.9567	0.4696	0.5570	0.8435	0.9898	0.1119	0.1925	0.8970	0.9601	0.5815	0.6530	0.8747	0.9747	0.3747	0.4992
150%	0.8459	0.9056	0.5474	0.5422	0.8763	0.9543	0.4866	0.5674	0.8358	0.9747	0.1411	0.2226	0.8950	0.9596	0.5718	0.6447	0.8751	0.9732	0.3844	0.5064
160%	0.8459	0.8978	0.5864	0.5592	0.8788	0.9489	0.5280	0.5921	0.8394	0.9762	0.1557	0.2443	0.8962	0.9591	0.5815	0.6512	0.8723	0.9674	0.3966	0.5086
170%	0.8423	0.9080	0.5134	0.5203	0.8792	0.9445	0.5523	0.6037	0.8337	0.9625	0.1898	0.2756	0.8958	0.9582	0.5839	0.6513	0.8715	0.9669	0.3942	0.5055
180%	0.8398	0.9012	0.5328	0.5258	0.8779	0.9397	0.5693	0.6086	0.8313	0.9552	0.2117	0.2949	0.8917	0.9557	0.5718	0.6377	0.8710	0.9625	0.4136	0.5167
190%	0.8508	0.9095	0.5572	0.5545	0.8792	0.9387	0.5815	0.6160	0.8285	0.9489	0.2263	0.3054	0.8962	0.9572	0.5912	0.6550	0.8706	0.9645	0.4015	0.5085

v. SVM-SMOTE

	SVM-SMOTE																			
	Decision Tree				Logistic Regression				Naïve Bayes				Random Forest				SVM			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
100%	0.8524	0.9075	0.5766	0.5656	0.8723	0.9742	0.3625	0.4861	0.8382	0.9971	0.0438	0.0828	0.8958	0.9664	0.5426	0.6344	0.8690	0.9835	0.2968	0.4303
110%	0.8337	0.9046	0.4793	0.4900	0.8710	0.9698	0.3771	0.4936	0.8341	0.9990	0.0097	0.0192	0.8086	0.8161	0.7713	0.5732	0.8463	0.9985	0.0852	0.1559
120%	0.8135	0.8448	0.6569	0.5400	0.8544	0.9061	0.5961	0.5771	0.8345	0.9990	0.0122	0.0239	0.2368	0.0856	0.9927	0.3024	0.8524	0.9966	0.1314	0.2288
130%	0.5126	0.4657	0.7470	0.3381	0.8682	0.9431	0.4939	0.5554	0.8341	0.9990	0.0097	0.0192	0.2275	0.0749	0.9903	0.2994	0.8646	0.9893	0.2409	0.3722
140%	0.4250	0.3650	0.7251	0.2959	0.8755	0.9543	0.4818	0.5633	0.8337	0.9990	0.0073	0.0144	0.1667	0.0000	1.0000	0.2857	0.8597	0.9873	0.2214	0.3447
150%	0.7218	0.7304	0.6788	0.4486	0.8767	0.9620	0.4501	0.5490	0.8337	0.9990	0.0073	0.0144	0.1671	0.0005	1.0000	0.2858	0.8581	0.9971	0.1630	0.2769
160%	0.7352	0.7640	0.5912	0.4267	0.8090	0.8326	0.6910	0.5467	0.8341	0.9985	0.0122	0.0239	0.1959	0.0389	0.9805	0.2890	0.8374	0.9995	0.0268	0.0520
170%	0.4850	0.4331	0.7445	0.3252	0.8747	0.9474	0.5109	0.5761	0.8333	0.9990	0.0049	0.0096	0.1667	0.0000	1.0000	0.2857	0.8613	0.9898	0.2190	0.3448
180%	0.2405	0.1343	0.7713	0.2529	0.8228	0.8506	0.6837	0.5626	0.8345	0.9990	0.0122	0.0239	0.1667	0.0000	1.0000	0.2857	0.8601	0.9932	0.1946	0.3168
190%	0.2591	0.1693	0.7080	0.2416	0.8686	0.9333	0.5450	0.5803	0.8345	0.9990	0.0122	0.0239	0.1667	0.0000	1.0000	0.2857	0.8625	0.9878	0.2360	0.3640

APPENDIX E: Hybrid Sampling Performance Metrics

i. Decision Tree + RUS 10% + Oversampling

	Decision Tree																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8540	0.9095	0.5766	0.5683	0.8540	0.9095	0.5766	0.5683	0.8540	0.9095	0.5766	0.5683	0.8540	0.9095	0.5766	0.5683	0.8540	0.9095	0.5766	0.5683
10%	0.8496	0.8968	0.6131	0.5760	0.8516	0.9129	0.5450	0.5504	0.8540	0.9051	0.5985	0.5775	0.8548	0.9105	0.5766	0.5697	0.8362	0.8808	0.6131	0.5551
20%	0.8512	0.9071	0.5718	0.5615	0.8520	0.9051	0.5864	0.5691	0.8524	0.9124	0.5523	0.5550	0.8443	0.8973	0.5791	0.5535	0.8281	0.8769	0.5839	0.5310
30%	0.8573	0.9085	0.6010	0.5839	0.8544	0.9109	0.5718	0.5669	0.8479	0.9066	0.5547	0.5487	0.8386	0.8925	0.5693	0.5404	0.6784	0.6779	0.6813	0.4139
40%	0.8455	0.9051	0.5474	0.5415	0.8532	0.9119	0.5596	0.5596	0.8532	0.9071	0.5839	0.5701	0.8573	0.9148	0.5693	0.5707	0.8139	0.8418	0.6740	0.5469
50%	0.8483	0.9109	0.5353	0.5405	0.8548	0.9051	0.6034	0.5808	0.8386	0.9022	0.5207	0.5182	0.8366	0.8968	0.5353	0.5219	0.3216	0.2263	0.7981	0.2817
60%	0.8617	0.9217	0.5620	0.5753	0.8475	0.9007	0.5815	0.5597	0.8475	0.9090	0.5401	0.5415	0.8491	0.9032	0.5791	0.5613	0.7997	0.8214	0.6910	0.5348
70%	0.8475	0.9090	0.5401	0.5415	0.8524	0.9071	0.5791	0.5667	0.8435	0.9105	0.5085	0.5199	0.8479	0.9002	0.5864	0.5624	0.2145	0.0949	0.8127	0.2564
80%	0.8589	0.9144	0.5815	0.5787	0.8402	0.8939	0.5718	0.5440	0.8496	0.9027	0.5839	0.5640	0.8504	0.9041	0.5815	0.5643	0.3001	0.1976	0.8127	0.2790
90%	0.8508	0.9114	0.5474	0.5501	0.8451	0.8968	0.5864	0.5579	0.8508	0.9100	0.5547	0.5534	0.8496	0.9075	0.5596	0.5535	0.6906	0.6871	0.7080	0.4327

ii. Decision Tree + RUS 20% + Oversampling

	Decision Tree																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8447	0.9036	0.5499	0.5413	0.8447	0.9036	0.5499	0.5413	0.8447	0.9036	0.5499	0.5413	0.8447	0.9036	0.5499	0.5413	0.8447	0.9036	0.5499	0.5413
10%	0.8451	0.9041	0.5499	0.5420	0.8406	0.8944	0.5718	0.5446	0.8524	0.9056	0.5864	0.5697	0.8471	0.9100	0.5328	0.5374	0.8139	0.8691	0.5377	0.4906
20%	0.8487	0.9085	0.5499	0.5479	0.8479	0.8973	0.6010	0.5685	0.8483	0.9012	0.5839	0.5621	0.8516	0.8998	0.6107	0.5783	0.2429	0.1382	0.7664	0.2523
30%	0.8451	0.8988	0.5766	0.5537	0.8414	0.8891	0.6034	0.5592	0.8439	0.8978	0.5742	0.5508	0.8410	0.8983	0.5547	0.5377	0.8106	0.8428	0.6496	0.5335
40%	0.8512	0.9041	0.5864	0.5677	0.8394	0.8978	0.5474	0.5319	0.8418	0.8900	0.6010	0.5588	0.8418	0.8998	0.5523	0.5379	0.2097	0.0813	0.8516	0.2643
50%	0.8435	0.8988	0.5669	0.5469	0.8471	0.8983	0.5912	0.5632	0.8496	0.9090	0.5523	0.5503	0.8447	0.8939	0.5985	0.5623	0.3147	0.2102	0.8370	0.2893
60%	0.8455	0.9027	0.5596	0.5470	0.8390	0.8944	0.5620	0.5378	0.8455	0.8998	0.5742	0.5533	0.8410	0.9066	0.5134	0.5184	0.4769	0.4204	0.7591	0.3260
70%	0.8479	0.8998	0.5888	0.5634	0.8374	0.8900	0.5742	0.5407	0.8577	0.9032	0.6302	0.5961	0.8536	0.9017	0.6131	0.5827	0.2340	0.1236	0.7859	0.2548
80%	0.8573	0.9080	0.6034	0.5849	0.8423	0.8944	0.5815	0.5513	0.8552	0.9090	0.5864	0.5745	0.8463	0.9007	0.5742	0.5546	0.2210	0.0905	0.8735	0.2721
90%	0.8532	0.9114	0.5620	0.5607	0.8548	0.9075	0.5912	0.5758	0.8560	0.9100	0.5864	0.5759	0.8491	0.8983	0.6034	0.5714	0.1614	0.0049	0.9440	0.2729

iii. Decision Tree + RUS 30% + Oversampling

	Decision Tree																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8402	0.8915	0.5839	0.5492	0.8402	0.8915	0.5839	0.5492	0.8402	0.8915	0.5839	0.5492	0.8402	0.8915	0.5839	0.5492	0.8402	0.8915	0.5839	0.5492
10%	0.8431	0.8964	0.5766	0.5505	0.8370	0.8866	0.5888	0.5463	0.8459	0.8968	0.5912	0.5612	0.8467	0.9012	0.5766	0.5563	0.2822	0.2078	0.6545	0.2331
20%	0.8540	0.9032	0.6083	0.5814	0.8374	0.8905	0.5718	0.5396	0.8532	0.8993	0.6229	0.5858	0.8418	0.8949	0.5693	0.5455	0.4830	0.4326	0.7348	0.3214
30%	0.8471	0.8978	0.5937	0.5642	0.8232	0.8740	0.5693	0.5177	0.8410	0.8944	0.5742	0.5463	0.8406	0.9007	0.5791	0.5478	0.2036	0.0827	0.8078	0.2527
40%	0.8406	0.8910	0.5888	0.5519	0.8548	0.8949	0.6545	0.6004	0.8504	0.8934	0.6350	0.5859	0.8455	0.8954	0.5937	0.5616	0.7944	0.8083	0.7251	0.5403
50%	0.8427	0.9002	0.5547	0.5403	0.8418	0.8920	0.5912	0.5548	0.8479	0.9007	0.5839	0.5614	0.8475	0.8939	0.5888	0.5628	0.1622	0.0131	0.9075	0.2653
60%	0.8479	0.9027	0.5742	0.5573	0.8406	0.8934	0.5766	0.5467	0.8447	0.8949	0.5937	0.5603	0.8390	0.8934	0.6083	0.5574	0.2478	0.1333	0.8200	0.2665
70%	0.8418	0.8895	0.6034	0.5598	0.8362	0.8920	0.5572	0.5313	0.8467	0.9017	0.5718	0.5542	0.8378	0.8983	0.5474	0.5294	0.2360	0.1251	0.7908	0.2565
80%	0.8483	0.9056	0.5620	0.5526	0.8386	0.8886	0.5888	0.5488	0.8423	0.8954	0.5766	0.5492	0.8390	0.8964	0.5693	0.5410	0.1614	0.0122	0.9075	0.2651
90%	0.8479	0.9056	0.5596	0.5509	0.8443	0.8900	0.6156	0.5685	0.8487	0.8973	0.6058	0.5718	0.8268	0.9012	0.5572	0.5175	0.1602	0.0068	0.9270	0.2690

iv. Decision Tree + RUS 40% + Oversampling

	Decision Tree																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8394	0.8891	0.5912	0.5510	0.8394	0.8891	0.5912	0.5510	0.8394	0.8891	0.5912	0.5510	0.8394	0.8891	0.5912	0.5510	0.8394	0.8891	0.5912	0.5510
10%	0.8386	0.8852	0.6058	0.5558	0.8471	0.8866	0.6496	0.5862	0.8435	0.8881	0.6204	0.5692	0.8455	0.8886	0.6302	0.5762	0.8009	0.8701	0.4550	0.4324
20%	0.8467	0.8944	0.6083	0.5695	0.8479	0.8915	0.6302	0.5801	0.8378	0.8856	0.5985	0.5516	0.8455	0.8959	0.5937	0.5616	0.8325	0.8803	0.5937	0.5416
30%	0.8451	0.8852	0.6448	0.5811	0.8370	0.8769	0.6375	0.5659	0.8378	0.8827	0.6131	0.5575	0.8281	0.8783	0.5766	0.5278	0.2234	0.0944	0.8686	0.2716
40%	0.8418	0.8876	0.6131	0.5638	0.8350	0.8798	0.6107	0.5523	0.8394	0.8895	0.5888	0.5500	0.8504	0.8949	0.6277	0.5831	0.2097	0.0856	0.8297	0.2592
50%	0.8431	0.8876	0.6204	0.5686	0.8398	0.8803	0.6375	0.5702	0.8427	0.8876	0.6180	0.5670	0.8410	0.8891	0.6010	0.5576	0.6938	0.6993	0.6667	0.4206
60%	0.8374	0.8866	0.5912	0.5479	0.8435	0.8895	0.6131	0.5663	0.8374	0.8818	0.6156	0.5579	0.8382	0.8818	0.6204	0.5611	0.2591	0.1489	0.8102	0.2671
70%	0.8459	0.8964	0.5937	0.5622	0.8362	0.8852	0.5912	0.5461	0.8414	0.8808	0.6448	0.5755	0.8362	0.8783	0.6253	0.5599	0.1987	0.0759	0.8127	0.2526
80%	0.8548	0.9085	0.5864	0.5738	0.8325	0.8735	0.6277	0.5554	0.8402	0.8822	0.6302	0.5680	0.8398	0.8818	0.6302	0.5674	0.1517	0.0131	0.8443	0.2491
90%	0.8504	0.9017	0.5937	0.5694	0.8273	0.8696	0.6156	0.5429	0.8317	0.8783	0.5985	0.5424	0.8398	0.8856	0.6107	0.5596	0.2279	0.1153	0.7908	0.2545

v. Decision Tree + RUS 50% + Oversampling

	Decision Tree																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8435	0.8837	0.6423	0.5777	0.8435	0.8837	0.6423	0.5777	0.8435	0.8837	0.6423	0.5777	0.8435	0.8837	0.6423	0.5777	0.8435	0.8837	0.6423	0.5777
10%	0.8374	0.8754	0.6472	0.5702	0.8455	0.8827	0.6594	0.5872	0.8394	0.8866	0.6034	0.5561	0.8410	0.8774	0.6594	0.5803	0.7835	0.8127	0.6375	0.4953
20%	0.8358	0.8769	0.6302	0.5612	0.8354	0.8788	0.6180	0.5558	0.8410	0.8793	0.6496	0.5767	0.8354	0.8745	0.6399	0.5644	0.7539	0.7800	0.6229	0.4576
30%	0.8435	0.8827	0.6472	0.5795	0.8370	0.8696	0.6740	0.5795	0.8106	0.8423	0.6521	0.5344	0.8309	0.8647	0.6618	0.5661	0.6902	0.6822	0.7299	0.4399
40%	0.8350	0.8813	0.6034	0.5493	0.8240	0.8560	0.6642	0.5571	0.8333	0.8740	0.6302	0.5576	0.8394	0.8764	0.6545	0.5760	0.7863	0.8034	0.7007	0.5222
50%	0.8398	0.8793	0.6423	0.5720	0.8350	0.8798	0.6107	0.5523	0.8431	0.8774	0.6715	0.5879	0.8382	0.8783	0.6375	0.5677	0.2364	0.1139	0.8491	0.2704
60%	0.8402	0.8832	0.6253	0.5661	0.8321	0.8759	0.6131	0.5490	0.8350	0.8740	0.6399	0.5638	0.8285	0.8706	0.6180	0.5456	0.2084	0.0754	0.8735	0.2689
70%	0.8366	0.8735	0.6521	0.5708	0.8248	0.8530	0.6837	0.5654	0.8406	0.8735	0.6764	0.5859	0.8337	0.8779	0.6131	0.5514	0.2072	0.0793	0.8467	0.2625
80%	0.8418	0.8861	0.6204	0.5667	0.8374	0.8740	0.6545	0.5729	0.8394	0.8662	0.7056	0.5943	0.8398	0.8783	0.6472	0.5739	0.2088	0.0764	0.8710	0.2685
90%	0.8500	0.8905	0.6472	0.5898	0.8382	0.8647	0.7056	0.5924	0.8354	0.8691	0.6667	0.5744	0.8289	0.8642	0.6521	0.5595	0.2352	0.1158	0.8321	0.2661

vi. Decision Tree + RUS 60% + Oversampling

	Decision Tree																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8394	0.8745	0.6642	0.5796	0.8394	0.8745	0.6642	0.4477	0.8394	0.8745	0.6642	0.5796	0.8394	0.8745	0.6642	0.5796	0.8394	0.8745	0.6642	0.5796
10%	0.8390	0.8691	0.6886	0.5877	0.8366	0.8613	0.7129	0.4818	0.8471	0.8749	0.7080	0.6069	0.8317	0.8676	0.6521	0.5636	0.6358	0.5995	0.8175	0.4280
20%	0.8329	0.8613	0.6910	0.5796	0.8131	0.8389	0.6837	0.4964	0.8386	0.8735	0.6642	0.5784	0.8345	0.8594	0.7105	0.5887	0.8305	0.8686	0.6399	0.5572
30%	0.8341	0.8618	0.6959	0.5831	0.8297	0.8574	0.6910	0.5109	0.8329	0.8618	0.6886	0.5787	0.8212	0.8511	0.6715	0.5559	0.2457	0.1246	0.8516	0.2734
40%	0.8350	0.8633	0.6934	0.5834	0.8309	0.8579	0.6959	0.5231	0.8358	0.8628	0.7007	0.5872	0.8350	0.8550	0.7348	0.5974	0.2405	0.1168	0.8589	0.2737
50%	0.8345	0.8662	0.6764	0.5768	0.8341	0.8676	0.6667	0.5426	0.8244	0.8477	0.7080	0.5734	0.8390	0.8696	0.6861	0.5869	0.2320	0.1032	0.8759	0.2754
60%	0.8386	0.8701	0.6813	0.5846	0.8236	0.8521	0.6813	0.5426	0.8390	0.8618	0.7251	0.6002	0.8305	0.8530	0.7178	0.5853	0.1655	0.0049	0.9684	0.2789
70%	0.8329	0.8642	0.6764	0.5744	0.8337	0.8603	0.7007	0.5572	0.8285	0.8579	0.6813	0.5697	0.8305	0.8594	0.6861	0.5743	0.2259	0.0964	0.8735	0.2733
80%	0.8508	0.8818	0.6959	0.6085	0.8260	0.8594	0.6594	0.5572	0.8354	0.8686	0.6691	0.5753	0.8260	0.8560	0.6764	0.5645	0.1667	0.0058	0.9708	0.2797
90%	0.8333	0.8672	0.6642	0.5868	0.8325	0.8613	0.6886	0.5669	0.8333	0.8657	0.6715	0.5732	0.8273	0.8574	0.6764	0.5662	0.2644	0.1479	0.8467	0.2773

vii. Decision Tree + RUS 70% + Oversampling

Decision Tree	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8122	0.8307	0.7202	0.5611	0.8122	0.8307	0.7202	0.5611	0.8122	0.8307	0.7202	0.5611	0.8122	0.8307	0.7202	0.5611	0.8122	0.8307	0.7202	0.5611
10%	0.8175	0.8433	0.6886	0.5571	0.8289	0.8457	0.7445	0.5919	0.8143	0.8326	0.7226	0.5646	0.8143	0.8380	0.7324	0.5761	0.8106	0.8404	0.6618	0.5381
20%	0.8224	0.8526	0.6715	0.5576	0.8240	0.8438	0.7251	0.5786	0.8127	0.8331	0.7105	0.5583	0.8127	0.8540	0.6983	0.5752	0.7762	0.7951	0.6813	0.5036
30%	0.8256	0.8487	0.7105	0.5759	0.8187	0.8365	0.7299	0.5731	0.8171	0.8370	0.7178	0.5668	0.8171	0.8287	0.7275	0.5631	0.5247	0.4871	0.7129	0.3333
40%	0.8256	0.8506	0.7007	0.5726	0.8301	0.8448	0.7567	0.5975	0.8183	0.8326	0.7470	0.5782	0.8183	0.8248	0.7543	0.5735	0.2251	0.1012	0.8443	0.2664
50%	0.8309	0.8530	0.7202	0.5867	0.8273	0.8472	0.7275	0.5840	0.8139	0.8297	0.7348	0.5682	0.8139	0.8526	0.7007	0.5749	0.2376	0.1144	0.8540	0.2719
60%	0.8370	0.8550	0.7470	0.6043	0.8268	0.8487	0.7178	0.5801	0.8224	0.8448	0.7105	0.5714	0.8224	0.8355	0.7397	0.5774	0.1606	0.0054	0.9367	0.2711
70%	0.8179	0.8345	0.7348	0.5736	0.8049	0.8253	0.7032	0.5458	0.8163	0.8384	0.7056	0.5615	0.8163	0.8258	0.7056	0.5477				
80%	0.8212	0.8418	0.7178	0.5723	0.8252	0.8482	0.7105	0.5754	0.8187	0.8350	0.7372	0.5755	0.8187	0.8491	0.6448	0.5375				
90%	0.8244	0.8462	0.7153	0.5759	0.8110	0.8311	0.7105	0.5562	0.8049	0.8219	0.7202	0.5517	0.8049	0.8404	0.7153	0.5692				

viii. Decision Tree + RUS 80% + Oversampling

	Decision Tree																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8143	0.8345	0.7129	0.5613	0.8110	0.8345	0.7129	0.5562	0.8143	0.8345	0.7129	0.5613	0.8143	0.8345	0.7129	0.5613	0.8143	0.8345	0.7129	0.5613
10%	0.8220	0.8331	0.7664	0.5893	0.8147	0.8224	0.7762	0.5826	0.8212	0.8307	0.7737	0.5905	0.8098	0.8131	0.7932	0.5816	0.7940	0.8282	0.6229	0.5020
20%	0.8179	0.8331	0.7421	0.5760	0.8183	0.8253	0.7835	0.5897	0.8106	0.8136	0.7956	0.5834	0.8094	0.8151	0.7810	0.5773				
30%	0.8147	0.8204	0.7859	0.5857	0.8110	0.8195	0.7689	0.5756	0.8041	0.8097	0.7762	0.5691	0.8147	0.8190	0.7932	0.5879				
40%	0.8175	0.8336	0.7372	0.5739	0.8102	0.8136	0.7932	0.5821	0.7924	0.8044	0.7324	0.5404	0.8187	0.8243	0.7908	0.5925				
50%	0.8135	0.8209	0.7762	0.5811	0.8021	0.8015	0.8054	0.5757	0.8216	0.8311	0.7737	0.5911	0.8240	0.8248	0.8200	0.6083				
60%	0.8151	0.8273	0.7543	0.5762	0.8086	0.8127	0.7883	0.5786	0.8118	0.8161	0.7908	0.5835	0.8131	0.8165	0.7956	0.5865				
70%	0.8191	0.8302	0.7640	0.5847	0.8017	0.8083	0.7689	0.5638	0.7936	0.8019	0.7518	0.5484	0.8131	0.8204	0.7762	0.5805				
80%	0.8232	0.8326	0.7762	0.5940	0.8049	0.8097	0.7810	0.5717	0.8118	0.8214	0.7640	0.5751	0.8155	0.8297	0.7445	0.5736				
90%	0.8005	0.8136	0.7348	0.5511	0.8159	0.8224	0.7835	0.5865	0.8204	0.8277	0.7835	0.5925	0.8054	0.8151	0.7567	0.5644				

ix. Logistic Regression + RUS 10% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8739	0.3771	0.9732	0.4992	0.8739	0.9732	0.3771	0.4992	0.8739	0.9732	0.3771	0.4992	0.8739	0.9732	0.3771	0.4992	0.8739	0.9732	0.3771	0.4992
10%	0.8739	0.3966	0.9693	0.5118	0.8743	0.9698	0.3966	0.5126	0.8727	0.9693	0.3893	0.5047	0.8743	0.9689	0.4015	0.5156	0.8564	0.9129	0.5742	0.5714
20%	0.8735	0.4136	0.9655	0.5215	0.8759	0.9659	0.4258	0.5335	0.8735	0.9669	0.4063	0.5170	0.8739	0.9630	0.4282	0.5309	0.8678	0.9392	0.5109	0.5630
30%	0.8743	0.4380	0.9616	0.5373	0.8755	0.9616	0.4453	0.5438	0.8755	0.9664	0.4209	0.5299	0.8759	0.9601	0.4550	0.5500	0.8735	0.9513	0.4842	0.5606
40%	0.8747	0.4647	0.9567	0.5528	0.8755	0.9577	0.4647	0.5544	0.8767	0.9650	0.4355	0.5408	0.8759	0.9523	0.4939	0.5702	0.8682	0.9280	0.5693	0.5902
50%	0.8792	0.4988	0.9552	0.5791	0.8800	0.9552	0.5036	0.5831	0.8796	0.9620	0.4672	0.5639	0.8763	0.9470	0.5231	0.5850	0.8706	0.9796	0.3260	0.4566
60%	0.8816	0.5304	0.9518	0.5989	0.8824	0.9523	0.5328	0.6016	0.8796	0.9591	0.4818	0.5714	0.8775	0.9426	0.5523	0.6005	0.8788	0.9562	0.4915	0.5747
70%	0.8820	0.5499	0.9484	0.6083	0.8828	0.9460	0.5669	0.6172	0.8820	0.9601	0.4915	0.5813	0.8779	0.9406	0.5645	0.6065	0.8747	0.9494	0.5012	0.5714
80%	0.8816	0.5645	0.9450	0.6138	0.8816	0.9431	0.5742	0.6178	0.8828	0.9582	0.5061	0.5901	0.8783	0.9377	0.5815	0.6144	0.8625	0.9144	0.6034	0.5940
90%	0.8824	0.5693	0.9450	0.6174	0.8804	0.9392	0.5864	0.6203	0.8840	0.9562	0.5231	0.6006	0.8755	0.9314	0.5961	0.6148	0.8735	0.9645	0.4185	0.5244

x. Logistic Regression + RUS 20% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8731	0.3942	0.9689	0.5086	0.8731	0.9689	0.3942	0.5086	0.8731	0.9689	0.3942	0.5086	0.8731	0.9689	0.3942	0.5086	0.8731	0.9689	0.3942	0.5086
10%	0.8747	0.4136	0.9669	0.5239	0.8743	0.9659	0.4161	0.5245	0.8755	0.9674	0.4161	0.5270	0.8751	0.9659	0.4209	0.5291	0.8662	0.9392	0.5012	0.5553
20%	0.8755	0.4380	0.9630	0.5397	0.8751	0.9620	0.4404	0.5403	0.8763	0.9650	0.4331	0.5386	0.8763	0.9596	0.4599	0.5534	0.8666	0.9397	0.5012	0.5560
30%	0.8763	0.4647	0.9586	0.5560	0.8783	0.9582	0.4793	0.5677	0.8783	0.9630	0.4550	0.5549	0.8796	0.9547	0.5036	0.5823	0.8743	0.9586	0.4526	0.5455
40%	0.8804	0.5134	0.9538	0.5886	0.8816	0.9518	0.5304	0.5989	0.8812	0.9606	0.4842	0.5760	0.8800	0.9484	0.5377	0.5989	0.8715	0.9684	0.3869	0.5008
50%	0.8836	0.5499	0.9504	0.6116	0.8836	0.9504	0.5499	0.6116	0.8804	0.9577	0.4939	0.5792	0.8804	0.9440	0.5620	0.6103	0.8731	0.9674	0.4015	0.5132
60%	0.8848	0.5645	0.9489	0.6203	0.8840	0.9470	0.5693	0.6207	0.8840	0.9586	0.5109	0.5949	0.8804	0.9406	0.5791	0.6174	0.8771	0.9518	0.5036	0.5774
70%	0.8832	0.5791	0.9440	0.6230	0.8816	0.9421	0.5791	0.6198	0.8840	0.9562	0.5231	0.6006	0.8804	0.9367	0.5985	0.6252	0.8755	0.9513	0.4964	0.5706
80%	0.8844	0.5839	0.9445	0.6275	0.8783	0.9353	0.5937	0.6193	0.8856	0.9562	0.5328	0.6083	0.8767	0.9304	0.6083	0.6219	0.8739	0.9655	0.4161	0.5237
90%	0.8808	0.5937	0.9382	0.6240	0.8771	0.9285	0.6204	0.6273	0.8861	0.9528	0.5523	0.6177	0.8735	0.9265	0.6083	0.6158	0.8739	0.9489	0.4988	0.5687

xi. Logistic Regression + RUS 30% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8747	0.4136	0.9669	0.5492	0.8747	0.9669	0.4136	0.5239	0.8747	0.9669	0.4136	0.5239	0.8747	0.9669	0.4136	0.5239	0.8747	0.9669	0.4136	0.5239
10%	0.8763	0.4404	0.9635	0.5505	0.8767	0.9640	0.4404	0.5435	0.8755	0.9640	0.4331	0.5370	0.8771	0.9630	0.4428	0.5457	0.8723	0.9640	0.4136	0.5191
20%	0.8788	0.4793	0.9586	0.5814	0.8779	0.9562	0.4866	0.5706	0.8800	0.9620	0.4696	0.5660	0.8755	0.9562	0.4647	0.5544	0.8727	0.9664	0.4039	0.5139
30%	0.8808	0.5255	0.9518	0.5642	0.8816	0.9523	0.5280	0.5978	0.8808	0.9601	0.4842	0.5751	0.8800	0.9499	0.5109	0.5866	0.8739	0.9625	0.4307	0.5323
40%	0.8844	0.5620	0.9489	0.5519	0.8824	0.9474	0.5572	0.6123	0.8828	0.9582	0.5061	0.5901	0.8816	0.9460	0.5401	0.6033	0.8731	0.9518	0.4793	0.5573
50%	0.8836	0.5718	0.9460	0.5403	0.8816	0.9426	0.5766	0.6188	0.8828	0.9547	0.5231	0.5981	0.8828	0.9406	0.5596	0.6142	0.8731	0.9684	0.3966	0.5102
60%	0.8820	0.5912	0.9401	0.5573	0.8783	0.9353	0.5937	0.6193	0.8848	0.9547	0.5353	0.6077	0.8832	0.9324	0.5669	0.6180	0.8706	0.9450	0.4988	0.5624
70%	0.8779	0.5888	0.9358	0.5598	0.8719	0.9251	0.6058	0.6118	0.8861	0.9533	0.5499	0.6166	0.8788	0.9270	0.5791	0.6142	0.8690	0.9372	0.5280	0.5733
80%	0.8788	0.5961	0.9353	0.5526	0.8662	0.9158	0.6180	0.6062	0.8861	0.9509	0.5620	0.6218	0.8783	0.9182	0.5888	0.6173	0.8735	0.9703	0.3893	0.5063
90%	0.8739	0.6156	0.9255	0.5509	0.8670	0.9129	0.6375	0.6150	0.8836	0.9460	0.5718	0.6209	0.8710	0.9071	0.6010	0.6084	0.8739	0.9645	0.4209	0.5266

xii. Logistic Regression + RUS 40% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8767	0.4428	0.9635	0.5449	0.8767	0.9635	0.4428	0.5449	0.8767	0.9635	0.4428	0.5449	0.8767	0.9635	0.4428	0.5449	0.8767	0.9635	0.4428	0.5449
10%	0.8812	0.4988	0.9591	0.5832	0.8788	0.9562	0.4915	0.5747	0.8808	0.9611	0.4793	0.5727	0.8800	0.9557	0.5012	0.5819	0.8686	0.9835	0.2944	0.4276
20%	0.8820	0.5255	0.9513	0.5975	0.8836	0.9523	0.5401	0.6074	0.8828	0.9562	0.5158	0.5947	0.8840	0.9499	0.5547	0.6146	0.8455	0.8939	0.6034	0.5656
30%	0.8852	0.5620	0.9489	0.6201	0.8824	0.9455	0.5669	0.6164	0.8840	0.9543	0.5328	0.6050	0.8824	0.9455	0.5669	0.6164	0.8702	0.9742	0.3504	0.4737
40%	0.8844	0.5742	0.9474	0.6235	0.8796	0.9397	0.5791	0.6158	0.8848	0.9528	0.5450	0.6120	0.8767	0.9353	0.5839	0.6122	0.8719	0.9708	0.3771	0.4952
50%	0.8844	0.5937	0.9382	0.6313	0.8751	0.9299	0.6010	0.6160	0.8844	0.9504	0.5547	0.6154	0.8682	0.9226	0.5961	0.6012	0.7855	0.7908	0.7591	0.5412
60%	0.8775	0.5961	0.9294	0.6187	0.8686	0.9202	0.6107	0.6077	0.8848	0.9474	0.5718	0.6233	0.8678	0.9178	0.6180	0.6091	0.8439	0.8822	0.6521	0.5820
70%	0.8743	0.6107	0.9212	0.6182	0.8642	0.9105	0.6326	0.6082	0.8812	0.9416	0.5791	0.6190	0.8637	0.9080	0.6423	0.6111	0.8427	0.8793	0.6594	0.5828
80%	0.8719	0.6350	0.9153	0.6229	0.8605	0.9032	0.6472	0.6073	0.8775	0.9348	0.5912	0.6168	0.8573	0.8993	0.6472	0.6018	0.8767	0.9630	0.4453	0.5463
90%	0.8637	0.6521	0.9056	0.6147	0.8516	0.8920	0.6496	0.5933	0.8779	0.9319	0.6083	0.6242	0.8532	0.8876	0.6813	0.6074	0.8646	0.9265	0.5547	0.5772

xiii. Logistic Regression + RUS 50% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8824	0.5304	0.9528	0.6006	0.8824	0.9528	0.5304	0.6006	0.8824	0.9528	0.5304	0.6006	0.8824	0.9528	0.5304	0.6006	0.8824	0.9528	0.5304	0.6006
10%	0.8836	0.5596	0.9484	0.6158	0.8812	0.9455	0.5596	0.6109	0.8844	0.9489	0.5620	0.6185	0.8828	0.9465	0.5645	0.6162	0.8528	0.9066	0.5839	0.5694
20%	0.8800	0.5766	0.9406	0.6156	0.8808	0.9406	0.5815	0.6192	0.8820	0.9455	0.5645	0.6146	0.8808	0.9406	0.5815	0.6192	0.8642	0.9299	0.5353	0.5677
30%	0.8771	0.5912	0.9343	0.6160	0.8674	0.9212	0.5985	0.6007	0.8812	0.9411	0.5815	0.6200	0.8698	0.9246	0.5961	0.6042	0.8698	0.9596	0.4209	0.5187
40%	0.8690	0.6058	0.9217	0.6066	0.8658	0.9144	0.6229	0.6074	0.8804	0.9367	0.5985	0.6252	0.8633	0.9119	0.6204	0.6021	0.8564	0.9109	0.5839	0.5755
50%	0.8670	0.6350	0.9134	0.6141	0.8621	0.9066	0.6399	0.6074	0.8751	0.9285	0.6083	0.6188	0.8621	0.9061	0.6423	0.6083	0.8609	0.9255	0.5377	0.5631
60%	0.8637	0.6569	0.9051	0.6164	0.8552	0.8973	0.6448	0.5975	0.8751	0.9236	0.6326	0.6280	0.8577	0.8929	0.6813	0.6147	0.8650	0.9348	0.5158	0.5601
70%	0.8605	0.6715	0.8983	0.6161	0.8455	0.8779	0.6837	0.5960	0.8743	0.9207	0.6423	0.6301	0.8479	0.8803	0.6861	0.6006	0.8597	0.9158	0.5791	0.5791
80%	0.8581	0.6740	0.8949	0.6128	0.8345	0.8628	0.6934	0.5828	0.8710	0.9158	0.6472	0.6259	0.8366	0.8608	0.7153	0.5933	0.8585	0.9163	0.5693	0.5728
90%	0.8443	0.7129	0.8706	0.6041	0.8317	0.8521	0.7299	0.5911	0.8682	0.9080	0.6691	0.6286	0.8345	0.8540	0.7372	0.5976	0.8520	0.8993	0.6156	0.5809

xiv. Logistic Regression + RUS 60% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8783	0.5426	0.9455	0.5979	0.8783	0.9455	0.5426	0.5979	0.8783	0.9455	0.5426	0.5979	0.8783	0.9455	0.5426	0.5979	0.8783	0.9455	0.5426	0.5979
10%	0.8755	0.5815	0.9343	0.6089	0.8739	0.9328	0.5791	0.6048	0.8779	0.9387	0.5742	0.6106	0.8751	0.9358	0.5718	0.6041	0.8629	0.9353	0.5012	0.5493
20%	0.8706	0.5937	0.9260	0.6047	0.8666	0.9202	0.5985	0.5993	0.8727	0.9309	0.5815	0.6035	0.8690	0.9231	0.5985	0.6037	0.8646	0.9538	0.4185	0.5074
30%	0.8662	0.6083	0.9178	0.6024	0.8605	0.9071	0.6277	0.6000	0.8690	0.9192	0.6180	0.6113	0.8573	0.9036	0.6253	0.5935	0.8524	0.9066	0.5815	0.5677
40%	0.8569	0.6253	0.9032	0.5928	0.8512	0.8920	0.6472	0.5918	0.8646	0.9119	0.6277	0.6071	0.8516	0.8934	0.6423	0.5906	0.8601	0.9285	0.5182	0.5525
50%	0.8479	0.6521	0.8871	0.5884	0.8394	0.8740	0.6667	0.5805	0.8601	0.9022	0.6496	0.6075	0.8418	0.8764	0.6691	0.5851	0.8719	0.9718	0.3723	0.4920
60%	0.8524	0.6837	0.8861	0.6069	0.8341	0.8647	0.6813	0.5779	0.8532	0.8920	0.6594	0.5996	0.8350	0.8613	0.7032	0.5868	0.8710	0.9698	0.3771	0.4936
70%	0.8414	0.7105	0.8676	0.5990	0.8244	0.8453	0.7202	0.5776	0.8524	0.8856	0.6861	0.6078	0.8240	0.8443	0.7226	0.5778	0.8706	0.9494	0.4769	0.5513
80%	0.8394	0.7202	0.8633	0.5992	0.8204	0.8375	0.7348	0.5769	0.8475	0.8783	0.6934	0.6025	0.8244	0.8448	0.7226	0.5784	0.8650	0.9275	0.5523	0.5769
90%	0.8260	0.7202	0.8472	0.5798	0.8049	0.8146	0.7567	0.5639	0.8402	0.8681	0.7007	0.5938	0.8102	0.8209	0.7567	0.5706	0.8715	0.9732	0.3625	0.4846

xv. Logistic Regression + RUS 70% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8593	0.6107	0.9090	0.5913	0.8593	0.9090	0.6107	0.5913	0.8593	0.9090	0.6107	0.5913	0.8593	0.9090	0.6107	0.5913	0.8593	0.9090	0.6107	0.5913
10%	0.8520	0.6448	0.8934	0.5922	0.8487	0.8886	0.6496	0.5888	0.8560	0.9007	0.6326	0.5943	0.8496	0.8900	0.6472	0.5891	0.8325	0.8798	0.5961	0.5426
20%	0.8443	0.6642	0.8803	0.5871	0.8313	0.8637	0.6691	0.5694	0.8467	0.8813	0.6740	0.5944	0.8345	0.8672	0.6715	0.5750	0.7928	0.8122	0.6959	0.5282
30%	0.8309	0.6934	0.8584	0.5775	0.8252	0.8472	0.7153	0.5770	0.8374	0.8647	0.7007	0.5896	0.8244	0.8482	0.7056	0.5726	0.8228	0.8633	0.6204	0.5385
40%	0.8260	0.7348	0.8443	0.5847	0.8106	0.8243	0.7421	0.5664	0.8313	0.8550	0.7129	0.5848	0.8155	0.8302	0.7421	0.5728	0.8244	0.8628	0.6326	0.5456
50%	0.8167	0.7567	0.8287	0.5791	0.7993	0.8039	0.7762	0.5631	0.8191	0.8370	0.7299	0.5736	0.7976	0.8010	0.7810	0.5627	0.8666	0.9431	0.4842	0.5475
60%	0.8102	0.7737	0.8175	0.5761	0.7867	0.7873	0.7835	0.5504	0.8151	0.8282	0.7494	0.5746	0.7908	0.7908	0.7908	0.5575	0.8143	0.8453	0.6594	0.5420
70%	0.7916	0.7859	0.7927	0.5569	0.7794	0.7762	0.7956	0.5459	0.8066	0.8170	0.7543	0.5652	0.7810	0.7771	0.8005	0.5492				
80%	0.7822	0.8005	0.7786	0.5506	0.7656	0.7538	0.8248	0.5398	0.7985	0.8049	0.7664	0.5590	0.7717	0.7625	0.8175	0.5441				
90%	0.7762	0.8151	0.7684	0.5483	0.7526	0.7372	0.8297	0.5279	0.7903	0.7898	0.7932	0.5577	0.7587	0.7450	0.8273	0.5333				

xvi. Logistic Regression + RUS 80% + Oversampling

	Logistic Regression																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8090	0.7494	0.8209	0.5667	0.8090	0.8209	0.7494	0.5667	0.8090	0.8209	0.7494	0.5667	0.8090	0.8209	0.7494	0.5667	0.8090	0.8209	0.7494	0.5667
10%	0.7928	0.7713	0.7971	0.5537	0.7899	0.7912	0.7835	0.5542	0.7964	0.8010	0.7737	0.5589	0.7899	0.7927	0.7762	0.5519	0.8532	0.9221	0.5085	0.5359
20%	0.7737	0.7932	0.7698	0.5388	0.7616	0.7523	0.8078	0.5304	0.7774	0.7747	0.7908	0.5421	0.7612	0.7538	0.7981	0.5269				
30%	0.7603	0.8127	0.7499	0.5306	0.7437	0.7260	0.8321	0.5198	0.7660	0.7572	0.8102	0.5358	0.7466	0.7309	0.8248	0.5203				
40%	0.7490	0.8370	0.7314	0.5264	0.7230	0.7002	0.8370	0.5018	0.7514	0.7367	0.8248	0.5252	0.7283	0.7056	0.8418	0.5081				
50%	0.7307	0.8540	0.7061	0.5139	0.7149	0.6876	0.8516	0.4989	0.7401	0.7202	0.8394	0.5184	0.7084	0.6769	0.8662	0.4976				
60%	0.7202	0.8613	0.6920	0.5064	0.7011	0.6672	0.8710	0.4928	0.7311	0.7080	0.8467	0.5121	0.6951	0.6579	0.8808	0.4905				
70%	0.7028	0.8808	0.6672	0.4969	0.6946	0.6574	0.8808	0.4902	0.7121	0.6832	0.8564	0.4979	0.6865	0.6477	0.8808	0.4836				
80%	0.6902	0.8905	0.6501	0.4893	0.6764	0.6316	0.9002	0.4811	0.7084	0.6759	0.8710	0.4990	0.6732	0.6277	0.9002	0.4787				
90%	0.6869	0.8954	0.6453	0.4881	0.6667	0.6170	0.9148	0.4778	0.6942	0.6550	0.8905	0.4926	0.6521	0.6010	0.9075	0.4651				

xvii. Naïve Bayes + RUS 10% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8414	0.9961	0.0681	0.1253	0.8414	0.9961	0.0681	0.1253	0.8414	0.9961	0.0681	0.1253	0.8414	0.9961	0.0681	0.1253	0.8414	0.9961	0.0681	0.1253
10%	0.8443	0.9951	0.0900	0.1616	0.8439	0.9951	0.0876	0.1575	0.8443	0.9951	0.0900	0.1616	0.8439	0.9951	0.0876	0.1575	0.8354	0.9990	0.0170	0.0333
20%	0.8435	0.9922	0.0998	0.1752	0.8439	0.9922	0.1022	0.1791	0.8439	0.9946	0.0900	0.1612	0.8443	0.9927	0.1022	0.1795	0.8358	0.9990	0.0195	0.0380
30%	0.8398	0.9849	0.1144	0.1922	0.8406	0.9854	0.1168	0.1963	0.8329	0.9800	0.0973	0.1626	0.8386	0.9825	0.1192	0.1976	0.8333	0.9990	0.0049	0.0096
40%	0.8305	0.9669	0.1484	0.2259	0.8345	0.9732	0.1411	0.2214	0.8313	0.9727	0.1241	0.1969	0.8350	0.9718	0.1509	0.2335	0.8345	0.9990	0.0122	0.0239
50%	0.8313	0.9659	0.1582	0.2381	0.8297	0.9645	0.1557	0.2336	0.8236	0.9596	0.1436	0.2134	0.8321	0.9630	0.1776	0.2607	0.8337	0.9995	0.0049	0.0097
60%	0.8281	0.9562	0.1873	0.2664	0.8297	0.9567	0.1946	0.2759	0.8204	0.9523	0.1606	0.2296	0.8309	0.9572	0.1995	0.2823	0.8341	0.9990	0.0097	0.0192
70%	0.8228	0.9440	0.2165	0.2894	0.8297	0.9523	0.2165	0.2977	0.8167	0.9455	0.1727	0.2391	0.8264	0.9474	0.2214	0.2984	0.8345	0.9990	0.0122	0.0239
80%	0.8216	0.9382	0.2384	0.3082	0.8195	0.9353	0.2409	0.3079	0.8179	0.9431	0.1922	0.2603	0.8256	0.9416	0.2457	0.3196	0.8345	0.9990	0.0122	0.0239
90%	0.8208	0.9304	0.2725	0.3363	0.8171	0.9270	0.2676	0.3279	0.8212	0.9382	0.2360	0.3055	0.8232	0.9319	0.2798	0.3453	0.8337	0.9990	0.0073	0.0144

xviii. Naïve Bayes + RUS 20% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8443	0.9946	0.0925	0.1652	0.8443	0.9946	0.0925	0.1652	0.8443	0.9946	0.0925	0.1652	0.8443	0.9946	0.0925	0.1652	0.8443	0.9946	0.0925	0.1652
10%	0.8402	0.9878	0.1022	0.1757	0.8390	0.9859	0.1046	0.1781	0.8410	0.9903	0.0949	0.1660	0.8394	0.9864	0.1046	0.1784	0.8345	0.9976	0.0195	0.0377
20%	0.8337	0.9757	0.1241	0.1992	0.8358	0.9781	0.1241	0.2012	0.8341	0.9781	0.1144	0.1869	0.8398	0.9820	0.1290	0.2116	0.8345	0.9990	0.0122	0.0239
30%	0.8301	0.9669	0.1460	0.2226	0.8301	0.9650	0.1557	0.2340	0.8289	0.9669	0.1387	0.2127	0.8313	0.9650	0.1630	0.2436	0.8337	0.9990	0.0073	0.0144
40%	0.8293	0.9582	0.1849	0.2653	0.8309	0.9596	0.1873	0.2697	0.8212	0.9528	0.1630	0.2330	0.8325	0.9606	0.1922	0.2767	0.8337	0.9990	0.0073	0.0144
50%	0.8260	0.9494	0.2092	0.2862	0.8277	0.9489	0.2214	0.2998	0.8175	0.9450	0.1800	0.2475	0.8248	0.9455	0.2214	0.2964	0.8337	0.9990	0.0073	0.0144
60%	0.8224	0.9377	0.2457	0.3156	0.8216	0.9372	0.2433	0.3125	0.8204	0.9436	0.2044	0.2750	0.8216	0.9377	0.2409	0.3103	0.8337	0.9990	0.0073	0.0144
70%	0.8216	0.9328	0.2652	0.3313	0.8163	0.9246	0.2749	0.3328	0.8240	0.9387	0.2506	0.3219	0.8167	0.9236	0.2822	0.3392	0.8341	0.9990	0.0097	0.0192
80%	0.8171	0.9231	0.2871	0.3435	0.8187	0.9217	0.3041	0.3587	0.8252	0.9372	0.2652	0.3359	0.8139	0.9144	0.3114	0.3580	0.8341	0.9990	0.0097	0.0192
90%	0.8155	0.9148	0.3187	0.3654	0.8135	0.9085	0.3382	0.3767	0.8212	0.9285	0.2847	0.3467	0.8151	0.9085	0.3479	0.3854	0.8354	0.9990	0.0170	0.0333

xix. Naïve Bayes + RUS 30% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8341	0.9791	0.1095	0.1804	0.8341	0.9791	0.1095	0.5239	0.8341	0.9791	0.1095	0.1804	0.8341	0.9791	0.1095	0.1804	0.8341	0.9791	0.1095	0.1804
10%	0.8305	0.9669	0.1484	0.2259	0.8301	0.9664	0.1484	0.5435	0.8285	0.9645	0.1484	0.2239	0.8309	0.9659	0.1557	0.2349	0.8341	0.9985	0.0122	0.0239
20%	0.8248	0.9557	0.1703	0.2448	0.8289	0.9577	0.1849	0.5706	0.8220	0.9528	0.1679	0.2392	0.8268	0.9567	0.1776	0.2548	0.8337	0.9990	0.0073	0.0144
30%	0.8228	0.9450	0.2117	0.2848	0.8232	0.9445	0.2165	0.5978	0.8191	0.9460	0.1849	0.2542	0.8232	0.9455	0.2117	0.2852	0.8341	0.9995	0.0073	0.0145
40%	0.8208	0.9358	0.2457	0.3137	0.8200	0.9343	0.2482	0.6123	0.8208	0.9411	0.2190	0.2894	0.8167	0.9294	0.2530	0.3152	0.8345	0.9990	0.0122	0.0239
50%	0.8175	0.9275	0.2676	0.3284	0.8163	0.9236	0.2798	0.6188	0.8232	0.9377	0.2506	0.3209	0.8167	0.9226	0.2871	0.3430	0.8341	0.9990	0.0097	0.0192
60%	0.8175	0.9173	0.3187	0.3680	0.8139	0.9129	0.3187	0.6193	0.8216	0.9299	0.2798	0.3433	0.8139	0.9124	0.3212	0.3651	0.8341	0.9985	0.0122	0.0239
70%	0.8139	0.9080	0.3431	0.3806	0.8114	0.9017	0.3601	0.6118	0.8200	0.9207	0.3163	0.3693	0.8122	0.9041	0.3528	0.3851	0.8341	0.9981	0.0146	0.0285
80%	0.8159	0.9007	0.3917	0.4149	0.8114	0.8929	0.4039	0.6062	0.8155	0.9119	0.3333	0.3759	0.8118	0.8949	0.3966	0.4127	0.8345	0.9990	0.0122	0.0239
90%	0.8110	0.8900	0.4161	0.4233	0.8086	0.8837	0.4331	0.6150	0.8082	0.8973	0.3625	0.3865	0.8033	0.8798	0.4209	0.4164	0.8341	0.9971	0.0195	0.0376

xx. Naïve Bayes + RUS 40% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8273	0.9620	0.1533	0.2283	0.8273	0.9620	0.1533	0.2283	0.8273	0.9620	0.1533	0.2283	0.8273	0.9620	0.1533	0.2283	0.8273	0.9620	0.1533	0.2283
10%	0.8240	0.9504	0.1922	0.2669	0.8232	0.9484	0.1971	0.2709	0.8224	0.9489	0.1898	0.2626	0.8240	0.9489	0.1995	0.2742	0.8337	0.9995	0.0049	0.0097
20%	0.8216	0.9392	0.2336	0.3038	0.8212	0.9372	0.2409	0.3099	0.8204	0.9406	0.2190	0.2889	0.8216	0.9372	0.2433	0.3125	0.8350	0.9985	0.0170	0.0333
30%	0.8179	0.9294	0.2603	0.3228	0.8183	0.9285	0.2676	0.3293	0.8228	0.9358	0.2579	0.3267	0.8167	0.9275	0.2628	0.3234	0.8337	0.9990	0.0073	0.0144
40%	0.8191	0.9231	0.2993	0.3555	0.8159	0.9168	0.3114	0.3606	0.8220	0.9270	0.2968	0.3572	0.8163	0.9153	0.3212	0.3682	0.8345	0.9990	0.0122	0.0239
50%	0.8159	0.9085	0.3528	0.3898	0.8135	0.9032	0.3650	0.3947	0.8208	0.9212	0.3187	0.3722	0.8127	0.9027	0.3625	0.3921	0.8398	0.9966	0.0560	0.1043
60%	0.8143	0.8954	0.4088	0.4232	0.8110	0.8925	0.4039	0.4160	0.8143	0.9066	0.3528	0.3877	0.8106	0.8925	0.4015	0.4141	0.8423	0.9976	0.0657	0.1219
70%	0.8139	0.8881	0.4428	0.4423	0.8017	0.8740	0.4404	0.4254	0.8082	0.8920	0.3893	0.4035	0.8086	0.8822	0.4404	0.4341	0.8423	0.9961	0.0730	0.1336
80%	0.8066	0.8730	0.4745	0.4498	0.7960	0.8647	0.4526	0.4251	0.8001	0.8735	0.4331	0.4193	0.7928	0.8555	0.4793	0.4354	0.8358	0.9976	0.0268	0.0515
90%	0.7924	0.8526	0.4915	0.4410	0.7835	0.8414	0.4939	0.4319	0.7932	0.8579	0.4696	0.4308	0.7826	0.8404	0.4939	0.4310	0.8414	0.9951	0.0730	0.1330

xxi. Naïve Bayes + RUS 50% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8224	0.9387	0.2409	0.3113	0.8224	0.9387	0.2409	0.3113	0.8224	0.9387	0.2409	0.3113	0.8224	0.9387	0.2409	0.3113	0.8224	0.9387	0.2409	0.3113
10%	0.8220	0.9304	0.2798	0.3438	0.8191	0.9280	0.2749	0.3363	0.8216	0.9304	0.2798	0.3433	0.8204	0.9280	0.2822	0.3437	0.8406	0.9976	0.0560	0.1048
20%	0.8171	0.9134	0.3358	0.3796	0.8167	0.9134	0.3333	0.3774	0.8204	0.9231	0.3163	0.3698	0.8183	0.9139	0.3406	0.3846	0.8345	0.9985	0.0146	0.0286
30%	0.8151	0.9017	0.3820	0.4078	0.8163	0.8993	0.4015	0.4215	0.8167	0.9090	0.3504	0.3892	0.8147	0.8998	0.3893	0.4118	0.8345	0.9966	0.0243	0.0467
40%	0.8090	0.8827	0.4404	0.4346	0.8131	0.8881	0.4380	0.4385	0.8122	0.8871	0.4112	0.4220	0.8118	0.8871	0.4355	0.4355	0.8362	0.9961	0.0365	0.0691
50%	0.8054	0.8710	0.4769	0.4495	0.8021	0.8681	0.4720	0.4429	0.7997	0.8730	0.4380	0.4215	0.8009	0.8691	0.4599	0.4350	0.8418	0.9917	0.0925	0.1631
60%	0.7843	0.8394	0.5085	0.4400	0.7859	0.8399	0.5158	0.4454	0.7895	0.8506	0.4915	0.4377	0.7883	0.8443	0.5085	0.4447	0.8406	0.9873	0.1071	0.1830
70%	0.7709	0.8122	0.5645	0.4509	0.7644	0.8088	0.5426	0.4343	0.7745	0.8292	0.5036	0.4268	0.7693	0.8161	0.5353	0.4361	0.8443	0.9873	0.1290	0.2163
80%	0.7563	0.7888	0.5937	0.4481	0.7543	0.7864	0.5937	0.4461	0.7697	0.8146	0.5547	0.4453	0.7547	0.7908	0.5742	0.4383	0.8358	0.9689	0.1703	0.2569
90%	0.7401	0.7698	0.5912	0.4312	0.7344	0.7582	0.6156	0.4358	0.7522	0.7859	0.5766	0.4369	0.7393	0.7659	0.6058	0.4365	0.8354	0.9591	0.2165	0.3048

xxii. Naïve Bayes + RUS 60% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8171	0.9129	0.3382	0.3813	0.8171	0.9129	0.3382	0.3813	0.8171	0.9129	0.3382	0.3813	0.8171	0.9129	0.3382	0.3813	0.8171	0.9129	0.3382	0.3813
10%	0.8167	0.8983	0.4088	0.4264	0.8155	0.8978	0.4039	0.4219	0.8147	0.8983	0.3966	0.4163	0.8159	0.8973	0.4088	0.4253	0.8378	0.9942	0.0560	0.1031
20%	0.8118	0.8803	0.4696	0.4541	0.8110	0.8808	0.4623	0.4492	0.8074	0.8740	0.4745	0.4509	0.8094	0.8769	0.4720	0.4522	0.8390	0.9951	0.0584	0.1079
30%	0.7895	0.8477	0.4988	0.4413	0.7932	0.8496	0.5109	0.4516	0.7920	0.8487	0.5085	0.4490	0.7895	0.8448	0.5134	0.4485	0.8406	0.9888	0.0998	0.1726
40%	0.7701	0.8146	0.5474	0.4425	0.7644	0.8044	0.5645	0.4440	0.7741	0.8238	0.5255	0.4368	0.7713	0.8136	0.5596	0.4492	0.8374	0.9713	0.1679	0.2560
50%	0.7478	0.7791	0.5912	0.4386	0.7461	0.7766	0.5937	0.4381	0.7530	0.7903	0.5669	0.4335	0.7474	0.7791	0.5888	0.4372	0.8354	0.9679	0.1727	0.2591
60%	0.7299	0.7547	0.6058	0.4278	0.7242	0.7460	0.6156	0.4266	0.7303	0.7547	0.6083	0.4292	0.7238	0.7479	0.6034	0.4214	0.8244	0.9479	0.2068	0.2819
70%	0.7088	0.7246	0.6302	0.4191	0.6995	0.7105	0.6448	0.4170	0.7234	0.7411	0.6350	0.4336	0.7024	0.7168	0.6302	0.4137	0.8212	0.9255	0.2993	0.3581
80%	0.6975	0.7071	0.6496	0.4172	0.6869	0.6934	0.6545	0.4107	0.7141	0.7251	0.6594	0.4346	0.6890	0.6973	0.6472	0.4095	0.8155	0.8959	0.4136	0.4277
90%	0.6849	0.6856	0.6813	0.4188	0.6736	0.6749	0.6667	0.4050	0.6975	0.6993	0.6886	0.4314	0.6841	0.6852	0.6788	0.4174	0.8001	0.8691	0.4550	0.4314

xxiii. Naïve Bayes + RUS 70% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.7859	0.8404	0.5134	0.4442	0.7859	0.8404	0.7859	0.4442	0.7859	0.8404	0.5134	0.4442	0.7859	0.8404	0.5134	0.4442	0.7859	0.8404	0.5134	0.4442
10%	0.7616	0.7995	0.5718	0.4442	0.7567	0.7942	0.7567	0.4382	0.7612	0.7995	0.5693	0.4428	0.7575	0.7951	0.5693	0.4390	0.8345	0.9528	0.2433	0.3289
20%	0.7295	0.7552	0.6010	0.4255	0.7238	0.7479	0.7238	0.4214	0.7328	0.7586	0.6034	0.4294	0.7287	0.7533	0.6058	0.4267	0.8277	0.9363	0.2847	0.3551
30%	0.7019	0.7178	0.6229	0.4106	0.7028	0.7178	0.7028	0.4131	0.7182	0.7328	0.6448	0.4327	0.7015	0.7144	0.6375	0.4159	0.8110	0.8895	0.4185	0.4247
40%	0.6906	0.6944	0.6715	0.4198	0.6849	0.6891	0.6849	0.4127	0.6963	0.6998	0.6788	0.4269	0.6837	0.6876	0.6642	0.4118	0.7952	0.8487	0.5280	0.4622
50%	0.6764	0.6754	0.6813	0.4124	0.6727	0.6672	0.6727	0.4165	0.6861	0.6827	0.7032	0.4275	0.6756	0.6701	0.7032	0.4194	0.7551	0.7937	0.5620	0.4334
60%	0.6650	0.6530	0.7251	0.4191	0.6626	0.6521	0.6626	0.4141	0.6732	0.6652	0.7129	0.4210	0.6675	0.6560	0.7251	0.4209	0.7291	0.7513	0.6180	0.4320
70%	0.6594	0.6443	0.7348	0.4183	0.6594	0.6438	0.6594	0.4191	0.6711	0.6530	0.7616	0.4356	0.6594	0.6418	0.7470	0.4223				
80%	0.6529	0.6297	0.7689	0.4247	0.6500	0.6297	0.6500	0.4173	0.6565	0.6355	0.7616	0.4250	0.6484	0.6258	0.7616	0.4193				
90%	0.6456	0.6185	0.7810	0.4235	0.6407	0.6151	0.6407	0.4163	0.6464	0.6112	0.8224	0.4367	0.6431	0.6131	0.7932	0.4256				

xxiv. Naïve Bayes + RUS 80% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.6764	0.6715	0.7007	0.4192	0.6764	0.6715	0.7007	0.4192	0.6764	0.6715	0.7007	0.4192	0.6764	0.6715	0.7007	0.4192	0.6764	0.6715	0.7007	0.4192
10%	0.6573	0.6438	0.7251	0.4136	0.6545	0.6399	0.7275	0.4124	0.6573	0.6443	0.7226	0.4128	0.6545	0.6404	0.7251	0.4116	0.6857	0.6842	0.6934	0.4238
20%	0.6460	0.6234	0.7591	0.4168	0.6440	0.6204	0.7616	0.4162	0.6480	0.6234	0.7713	0.4221	0.6456	0.6204	0.7713	0.4204				
30%	0.6383	0.6068	0.7956	0.4230	0.6358	0.6054	0.7883	0.4191	0.6367	0.6019	0.8102	0.4264	0.6342	0.6005	0.8029	0.4225				
40%	0.6241	0.5839	0.8248	0.4224	0.6204	0.5791	0.8273	0.4208	0.6196	0.5762	0.8370	0.4231	0.6192	0.5771	0.8297	0.4207				
50%	0.6115	0.5635	0.8516	0.4222	0.6111	0.5640	0.8467	0.4205	0.6002	0.5474	0.8637	0.4186	0.6046	0.5547	0.8540	0.4186				
60%	0.5957	0.5416	0.8662	0.4166	0.5929	0.5382	0.8662	0.4149	0.5702	0.5080	0.8808	0.4058	0.5750	0.5148	0.8759	0.4072				
70%	0.5604	0.4934	0.8954	0.4044	0.5734	0.5114	0.8832	0.4083	0.5446	0.4754	0.8905	0.3946	0.5560	0.4876	0.8978	0.4026				
80%	0.5296	0.4555	0.9002	0.3895	0.5438	0.4725	0.9002	0.3968	0.5049	0.4248	0.9051	0.3786	0.5231	0.4462	0.9075	0.3881				
90%	0.5109	0.4287	0.9221	0.3859	0.5223	0.4433	0.9173	0.3903	0.4781	0.3878	0.9294	0.3725	0.4939	0.4068	0.9294	0.3797				

xxv.Random Forest + RUS 10% + Oversampling

	Naïve Bayes																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8893	0.9567	0.5523	0.6245	0.8893	0.9567	0.5523	0.6245	0.8893	0.9567	0.5523	0.6245	0.8893	0.9567	0.5523	0.6245	0.8893	0.9567	0.5523	0.6245
10%	0.8897	0.9562	0.5572	0.6274	0.8933	0.9586	0.5669	0.6392	0.8925	0.9586	0.5620	0.6355	0.8946	0.9611	0.5620	0.6399	0.8739	0.8876	0.8054	0.6804
20%	0.8885	0.9547	0.5572	0.6248	0.8958	0.9582	0.5839	0.6513	0.8950	0.9567	0.5864	0.6505	0.8913	0.9562	0.5669	0.6349	0.3706	0.2530	0.9586	0.3368
30%	0.8917	0.9557	0.5718	0.6377	0.8917	0.9533	0.5839	0.6426	0.8958	0.9567	0.5912	0.6541	0.8982	0.9596	0.5912	0.6594	0.1679	0.0015	1.0000	0.2860
40%	0.8873	0.9499	0.5742	0.6293	0.8921	0.9552	0.5766	0.6405	0.8938	0.9562	0.5815	0.6459	0.8946	0.9538	0.5985	0.6543	0.1800	0.0175	0.9927	0.2875
50%	0.8929	0.9543	0.5864	0.6461	0.8946	0.9557	0.5888	0.6505	0.8942	0.9543	0.5937	0.6515	0.8958	0.9557	0.5961	0.6560	0.1667	0.0000	1.0000	0.2857
60%	0.8942	0.9543	0.5937	0.6515	0.8962	0.9557	0.5985	0.6578	0.8954	0.9557	0.5937	0.6542	0.8950	0.9557	0.5912	0.6523	0.1667	0.0000	1.0000	0.2857
70%	0.8938	0.9543	0.5912	0.6497	0.8938	0.9543	0.5912	0.6497	0.8950	0.9557	0.5912	0.6523	0.8909	0.9543	0.5742	0.6370	0.1667	0.0000	1.0000	0.2857
80%	0.8933	0.9538	0.5912	0.6489	0.8929	0.9557	0.5791	0.6432	0.8974	0.9547	0.6107	0.6649	0.8938	0.9499	0.6131	0.6580	0.1667	0.0000	1.0000	0.2857
90%	0.8950	0.9513	0.6131	0.6606	0.8966	0.9596	0.5815	0.6521	0.8938	0.9523	0.6010	0.6534	0.8954	0.9533	0.6058	0.6587	0.1667	0.0000	1.0000	0.2857

xxvi. Random Forest + RUS 20% + Oversampling

	Random Forest																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8946	0.9577	0.5791	0.6467	0.8946	0.9577	0.5791	0.6467	0.8946	0.9577	0.5791	0.6467	0.8946	0.9577	0.5791	0.6467	0.8946	0.9577	0.5791	0.6467
10%	0.8954	0.9547	0.5985	0.6560	0.8954	0.9538	0.6034	0.6578	0.8958	0.9572	0.5888	0.6532	0.8954	0.9523	0.6107	0.6605	0.8763	0.8954	0.7810	0.6779
20%	0.8946	0.9518	0.6083	0.6579	0.8958	0.9513	0.6180	0.6641	0.8954	0.9513	0.6156	0.6623	0.8946	0.9523	0.6058	0.6570	0.4079	0.2993	0.9513	0.3488
30%	0.8962	0.9528	0.6131	0.6632	0.8970	0.9513	0.6253	0.6693	0.8929	0.9509	0.6034	0.6526	0.8966	0.9528	0.6156	0.6649	0.1695	0.0034	1.0000	0.2864
40%	0.8925	0.9470	0.6204	0.6581	0.8946	0.9484	0.6253	0.6641	0.8962	0.9499	0.6277	0.6684	0.8950	0.9513	0.6131	0.6606	0.1667	0.0000	1.0000	0.2857
50%	0.8994	0.9523	0.6350	0.6779	0.8950	0.9470	0.6350	0.6684	0.8954	0.9513	0.6156	0.6623	0.8938	0.9494	0.6156	0.6589	0.1918	0.0345	0.9781	0.2875
60%	0.8950	0.9484	0.6277	0.6658	0.8974	0.9523	0.6229	0.6693	0.8970	0.9509	0.6277	0.6701	0.8978	0.9513	0.6302	0.6727	0.1667	0.0000	1.0000	0.2857
70%	0.8925	0.9431	0.6399	0.6650	0.8950	0.9513	0.6131	0.6606	0.8946	0.9494	0.6204	0.6623	0.8933	0.9465	0.6277	0.6624	0.1667	0.0000	1.0000	0.2857
80%	0.8909	0.9440	0.6253	0.6564	0.8974	0.9543	0.6131	0.6658	0.8954	0.9484	0.6302	0.6675	0.8954	0.9484	0.6302	0.6675	0.1667	0.0000	1.0000	0.2857
90%	0.8938	0.9474	0.6253	0.6624	0.8958	0.9479	0.6350	0.6701	0.8962	0.9489	0.6326	0.6701	0.8950	0.9484	0.6277	0.6658	0.1667	0.0000	1.0000	0.2857

xxvii. Random Forest + RUS 30% + Oversampling

	Random Forest																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8897	0.9479	0.5985	0.6440	0.8897	0.9479	0.5985	0.6440	0.8897	0.9479	0.5985	0.6440	0.8897	0.9479	0.5985	0.6440	0.8897	0.9479	0.5985	0.6440
10%	0.8921	0.9470	0.6180	0.6563	0.8933	0.9470	0.6253	0.6615	0.8978	0.9499	0.6375	0.6753	0.8925	0.9445	0.6326	0.6624	0.7956	0.7917	0.8151	0.5707
20%	0.8958	0.9474	0.6375	0.6709	0.8974	0.9474	0.6472	0.6777	0.8909	0.9436	0.6277	0.6573	0.8966	0.9484	0.6375	0.6727	0.4676	0.3805	0.9027	0.3611
30%	0.8942	0.9479	0.6253	0.6632	0.8933	0.9455	0.6326	0.6641	0.8946	0.9470	0.6326	0.6667	0.8917	0.9421	0.6399	0.6633	0.2129	0.0608	0.9732	0.2919
40%	0.8933	0.9431	0.6448	0.6683	0.8913	0.9426	0.6350	0.6608	0.8954	0.9426	0.6594	0.6775	0.8929	0.9460	0.6277	0.6615	0.1979	0.0380	0.9976	0.2931
50%	0.8917	0.9382	0.6594	0.6700	0.8933	0.9455	0.6326	0.6641	0.8954	0.9445	0.6496	0.6742	0.8913	0.9406	0.6448	0.6642	0.1667	0.0000	1.0000	0.2857
60%	0.8986	0.9455	0.6642	0.6859	0.8929	0.9401	0.6569	0.6716	0.8929	0.9377	0.6691	0.6757	0.8958	0.9460	0.6448	0.6734	0.1691	0.0029	1.0000	0.2863
70%	0.8905	0.9377	0.6545	0.6658	0.8933	0.9411	0.6545	0.6717	0.8938	0.9426	0.6496	0.6709	0.8962	0.9445	0.6545	0.6776	0.1727	0.0083	0.9951	0.2862
80%	0.8933	0.9406	0.6569	0.6725	0.8917	0.9416	0.6423	0.6642	0.8958	0.9455	0.6472	0.6743	0.8933	0.9406	0.6569	0.6725	0.1667	0.0000	1.0000	0.2857
90%	0.8921	0.9382	0.6618	0.6716	0.8929	0.9387	0.6642	0.6741	0.8942	0.9411	0.6594	0.6750	0.8946	0.9397	0.6691	0.6790	0.1667	0.0000	1.0000	0.2857

xxviii. Random Forest + RUS 40% + Oversampling

	Random Forest																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8946	0.9382	0.6764	0.6814	0.8946	0.9382	0.6764	0.6814	0.8946	0.9382	0.6764	0.6814	0.8946	0.9382	0.6764	0.6814	0.8946	0.9382	0.6764	0.6814
10%	0.8901	0.9367	0.6569	0.6658	0.8913	0.9353	0.6715	0.6732	0.8942	0.9372	0.6788	0.6813	0.8929	0.9372	0.6715	0.6765	0.8374	0.8983	0.5328	0.5221
20%	0.8905	0.9363	0.6618	0.6683	0.8925	0.9367	0.6715	0.6756	0.8950	0.9367	0.6861	0.6853	0.8929	0.9348	0.6837	0.6804	0.8260	0.8214	0.8491	0.6193
30%	0.8901	0.9333	0.6740	0.6715	0.8913	0.9343	0.6764	0.6748	0.8925	0.9363	0.6740	0.6764	0.8901	0.9324	0.6788	0.6731	0.2429	0.0915	1.0000	0.3057
40%	0.8897	0.9294	0.6910	0.6762	0.8958	0.9382	0.6837	0.6862	0.8938	0.9358	0.6837	0.6820	0.8929	0.9372	0.6715	0.6765	0.1740	0.0088	1.0000	0.2875
50%	0.8893	0.9319	0.6764	0.6707	0.8929	0.9333	0.6910	0.6827	0.8925	0.9333	0.6886	0.6811	0.8869	0.9304	0.6691	0.6634	0.2137	0.0574	0.9951	0.2967
60%	0.8881	0.9275	0.6910	0.6730	0.8885	0.9304	0.6788	0.6699	0.8958	0.9348	0.7007	0.6915	0.8913	0.9319	0.6886	0.6787	0.1723	0.0068	1.0000	0.2871
70%	0.8877	0.9265	0.6934	0.6730	0.8897	0.9309	0.6837	0.6739	0.8929	0.9343	0.6861	0.6812	0.8905	0.9309	0.6886	0.6770	0.1667	0.0000	1.0000	0.2857
80%	0.8901	0.9280	0.7007	0.6800	0.8861	0.9265	0.6837	0.6667	0.8950	0.9338	0.7007	0.6898	0.8913	0.9348	0.6740	0.6740	0.1671	0.0005	1.0000	0.2858
90%	0.8861	0.9246	0.6934	0.6698	0.8869	0.9270	0.6861	0.6690	0.8905	0.9343	0.6715	0.6715	0.8865	0.9270	0.6837	0.6675	0.1671	0.0005	1.0000	0.2858

xxix. Random Forest + RUS 50% + Oversampling

	Random Forest																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8917	0.9304	0.6983	0.6825	0.8917	0.9304	0.6983	0.6825	0.8917	0.9304	0.6983	0.6825	0.8917	0.9304	0.6983	0.6825	0.8917	0.9304	0.6983	0.6825
10%	0.8877	0.9226	0.7129	0.6790	0.8844	0.9241	0.6861	0.6643	0.8893	0.9246	0.7129	0.6822	0.8877	0.9246	0.7032	0.6760	0.8710	0.8900	0.7762	0.6674
20%	0.8901	0.9236	0.7226	0.6867	0.8848	0.9212	0.7032	0.6705	0.8840	0.9226	0.6910	0.6651	0.8917	0.9299	0.7007	0.6833	0.2968	0.1586	0.9878	0.3189
30%	0.8873	0.9236	0.7056	0.6760	0.8877	0.9231	0.7105	0.6783	0.8893	0.9260	0.7056	0.6800	0.8824	0.9187	0.7007	0.6651	0.4740	0.3757	0.9659	0.3797
40%	0.8893	0.9226	0.7226	0.6851	0.8861	0.9217	0.7080	0.6744	0.8877	0.9270	0.6910	0.6722	0.8885	0.9217	0.7226	0.6835	0.4376	0.3333	0.9586	0.3623
50%	0.8856	0.9178	0.7251	0.6788	0.8885	0.9217	0.7226	0.6835	0.8929	0.9275	0.7202	0.6916	0.8861	0.9207	0.7129	0.6759	0.3402	0.2175	0.9538	0.3252
60%	0.8816	0.9182	0.6983	0.6628	0.8909	0.9236	0.7275	0.6897	0.8889	0.9255	0.7056	0.6792	0.8869	0.9221	0.7105	0.6767	0.1788	0.0146	1.0000	0.2887
70%	0.8865	0.9197	0.7202	0.6789	0.8828	0.9173	0.7105	0.6690	0.8861	0.9221	0.7056	0.6736	0.8861	0.9163	0.7348	0.6825	0.1671	0.0005	1.0000	0.2858
80%	0.8808	0.9100	0.7348	0.6726	0.8861	0.9187	0.7226	0.6789	0.8861	0.9197	0.7178	0.6774	0.8881	0.9192	0.7324	0.6856	0.1727	0.0073	1.0000	0.2872
90%	0.8775	0.9095	0.7178	0.6614	0.8873	0.9182	0.7324	0.6841	0.8897	0.9226	0.7251	0.6866	0.8885	0.9226	0.7178	0.6821	0.1691	0.0029	1.0000	0.2863

xxx. Random Forest + RUS 60% + Oversampling

	Random Forest																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8832	0.9119	0.7397	0.6786	0.8832	0.9119	0.7397	0.6786	0.8832	0.9119	0.7397	0.6786	0.8832	0.9119	0.7397	0.6786	0.8832	0.9119	0.7397	0.6786
10%	0.8824	0.9105	0.7421	0.6778	0.8812	0.9071	0.7518	0.6784	0.8840	0.9109	0.7494	0.6829	0.8840	0.9100	0.7543	0.6843	0.8483	0.8628	0.7762	0.6304
20%	0.8800	0.9080	0.7397	0.6726	0.8844	0.9109	0.7518	0.6844	0.8836	0.9109	0.7470	0.6815	0.8820	0.9066	0.7591	0.6820	0.6549	0.6107	0.8759	0.4583
30%	0.8824	0.9075	0.7567	0.6820	0.8836	0.9071	0.7664	0.6870	0.8796	0.9051	0.7518	0.6754	0.8824	0.9071	0.7591	0.6827	0.2336	0.0827	0.9878	0.3005
40%	0.8824	0.9066	0.7616	0.6834	0.8816	0.9056	0.7616	0.6819	0.8848	0.9085	0.7664	0.6893	0.8840	0.9090	0.7591	0.6857	0.6732	0.6360	0.8589	0.4669
50%	0.8771	0.9022	0.7518	0.6710	0.8836	0.9095	0.7543	0.6836	0.8848	0.9114	0.7518	0.6851	0.8812	0.9071	0.7518	0.6784	0.1723	0.0068	1.0000	0.2871
60%	0.8808	0.9041	0.7640	0.6811	0.8828	0.9056	0.7689	0.6862	0.8828	0.9071	0.7616	0.6842	0.8824	0.9075	0.7567	0.6820	0.1667	0.0000	1.0000	0.2857
70%	0.8828	0.9036	0.7786	0.6889	0.8852	0.9080	0.7713	0.6914	0.8848	0.9075	0.7713	0.6906	0.8836	0.9056	0.7737	0.6891	0.1703	0.0044	1.0000	0.2866
80%	0.8775	0.9017	0.7567	0.6732	0.8792	0.9027	0.7616	0.6775	0.8824	0.9061	0.7640	0.6841	0.8788	0.9027	0.7591	0.6761	0.1667	0.0000	1.0000	0.2857
90%	0.8771	0.8993	0.7664	0.6752	0.8779	0.8988	0.7737	0.6788	0.8816	0.9061	0.7591	0.6812	0.8812	0.9056	0.7591	0.4174	0.1671	0.0005	1.0000	0.2858

xxxi. Random Forest + RUS 70% + Oversampling

	Random Forest																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8747	0.8934	0.7810	0.6751	0.8747	0.8934	0.7810	0.6751	0.8747	0.8934	0.7810	0.6751	0.8747	0.8934	0.7810	0.6751	0.8747	0.8934	0.7810	0.6751
10%	0.8747	0.8900	0.7981	0.6798	0.8743	0.8900	0.7956	0.6784	0.8755	0.8944	0.7810	0.6765	0.8763	0.8905	0.8054	0.6846	0.8402	0.8477	0.8029	0.6262
20%	0.8751	0.8886	0.8078	0.6831	0.8719	0.8891	0.7859	0.6715	0.8735	0.8876	0.8029	0.6790	0.8731	0.8891	0.7932	0.6756	0.8297	0.8331	0.8127	0.6140
30%	0.8690	0.8832	0.7981	0.6701	0.8731	0.8905	0.7859	0.6736	0.8735	0.8886	0.7981	0.6777	0.8723	0.8847	0.8102	0.6789	0.7717	0.7616	0.8224	0.5456
40%	0.8694	0.8842	0.7956	0.6701	0.8743	0.8881	0.8054	0.6811	0.8702	0.8861	0.7908	0.6701	0.8739	0.8861	0.8127	0.6823	0.2332	0.0808	0.9951	0.3020
50%	0.8686	0.8818	0.8029	0.6707	0.8682	0.8818	0.8005	0.6694	0.8715	0.8891	0.7835	0.6701	0.8706	0.8847	0.8005	0.6735	0.2080	0.0501	0.9976	0.2957
60%	0.8719	0.8847	0.8078	0.6776	0.8670	0.8818	0.7932	0.6653	0.8751	0.8915	0.7932	0.6792	0.8702	0.8832	0.8054	0.6741	0.1825	0.0190	1.0000	0.2896
70%	0.8694	0.8808	0.8127	0.6747	0.8686	0.8842	0.7908	0.6674	0.8715	0.8871	0.7932	0.6729	0.8698	0.8852	0.7932	0.6701				
80%	0.8658	0.8793	0.7981	0.6646	0.8682	0.8808	0.8054	0.6707	0.8763	0.8900	0.8078	0.6852	0.8723	0.8856	0.8054	0.6776				
90%	0.8646	0.8749	0.8127	0.6667	0.8698	0.8847	0.7956	0.6708	0.8747	0.8895	0.8005	0.6805	0.8719	0.8866	0.7981	0.6749				

xxxii. Random Forest + RUS 80% + Oversampling

	Random Forest																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8585	0.8633	0.8345	0.6628	0.8585	0.8633	0.8345	0.6628	0.8585	0.8633	0.8345	0.6628	0.8585	0.8633	0.8345	0.6628	0.8585	0.8633	0.8345	0.6628
10%	0.8609	0.8662	0.8345	0.6667	0.8609	0.8647	0.8418	0.6686	0.8621	0.8667	0.8394	0.6699	0.8597	0.8647	0.8345	0.6647	0.8252	0.8535	0.6837	0.5660
20%	0.8613	0.8652	0.8418	0.6692	0.8564	0.8594	0.8418	0.6616	0.8573	0.8594	0.8467	0.6641	0.8609	0.8637	0.8467	0.6699				
30%	0.8605	0.8633	0.8467	0.6692	0.8564	0.8589	0.8443	0.6622	0.8573	0.8618	0.8345	0.6609	0.8577	0.8599	0.8467	0.6648				
40%	0.8577	0.8599	0.8467	0.6648	0.8560	0.8594	0.8394	0.6603	0.8581	0.8628	0.8345	0.6622	0.8560	0.8584	0.8443	0.6616				
50%	0.8573	0.8589	0.8491	0.6648	0.8552	0.8560	0.8516	0.6623	0.8589	0.8608	0.8491	0.6673	0.8516	0.8521	0.8491	0.6560				
60%	0.8548	0.8574	0.8418	0.6590	0.8524	0.8521	0.8540	0.6585	0.8581	0.8594	0.8516	0.6667	0.8516	0.8521	0.8491	0.6560				
70%	0.8512	0.8501	0.8564	0.6573	0.8548	0.8560	0.8491	0.6610	0.8544	0.8574	0.8394	0.6578	0.8508	0.8516	0.8467	0.6541				
80%	0.8528	0.8521	0.8564	0.6598	0.8556	0.8560	0.8540	0.6635	0.8589	0.8603	0.8516	0.6679	0.8483	0.8467	0.8564	0.6531				
90%	0.8516	0.8516	0.8516	0.6567	0.8512	0.8501	0.8564	0.6573	0.8532	0.8550	0.8443	0.6572	0.8504	0.8501	0.8516	0.6548				

xxxiii.SVM + RUS 10% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8678	0.9805	0.3041	0.4340	0.8678	0.9805	0.3041	0.4340	0.8678	0.9805	0.3041	0.4340	0.8678	0.9805	0.3041	0.4340	0.8678	0.9805	0.3041	0.4340
10%	0.8686	0.9796	0.3139	0.4433	0.8678	0.9796	0.3090	0.4379	0.8678	0.9796	0.3090	0.4379	0.8686	0.9800	0.3114	0.4414	0.8569	0.9956	0.1630	0.2752
20%	0.8674	0.9742	0.3333	0.4559	0.8674	0.9776	0.3163	0.4429	0.8682	0.9781	0.3187	0.4463	0.8694	0.9771	0.3309	0.4579	0.8540	0.9951	0.1484	0.2531
30%	0.8706	0.9708	0.3698	0.4880	0.8702	0.9762	0.3406	0.4667	0.8686	0.9786	0.3187	0.4471	0.8702	0.9747	0.3479	0.4719	0.8637	0.9844	0.2603	0.3891
40%	0.8739	0.9689	0.3990	0.5133	0.8719	0.9742	0.3601	0.4837	0.8674	0.9766	0.3212	0.4467	0.8715	0.9674	0.3917	0.5039	0.8544	0.9956	0.1484	0.2536
50%	0.8751	0.9655	0.4234	0.5305	0.8702	0.9737	0.3528	0.4754	0.8674	0.9762	0.3236	0.4486	0.8727	0.9674	0.3990	0.5109	0.8564	0.9961	0.1582	0.2686
60%	0.8767	0.9616	0.4526	0.5503	0.8747	0.9742	0.3771	0.5008	0.8682	0.9757	0.3309	0.4556	0.8731	0.9659	0.4088	0.5177	0.8078	0.8633	0.5304	0.4791
70%	0.8779	0.9572	0.4818	0.5681	0.8727	0.9693	0.3893	0.5047	0.8682	0.9757	0.3309	0.4556	0.8723	0.9640	0.4136	0.5191	0.8427	0.9985	0.0633	0.1182
80%	0.8759	0.9528	0.4915	0.5690	0.8723	0.9659	0.4039	0.5131	0.8694	0.9757	0.3382	0.4633	0.8723	0.9625	0.4209	0.5234	0.8406	0.9990	0.0487	0.0924
90%	0.8767	0.9489	0.5158	0.5824	0.8727	0.9650	0.4112	0.5184	0.8690	0.9752	0.3382	0.4626	0.8723	0.9611	0.4282	0.5277	0.8682	0.9781	0.3187	0.4463

xxxiv.SVM + RUS 20% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8674	0.9791	0.3090	0.4372	0.8674	0.9791	0.3090	0.4372	0.8674	0.9791	0.3090	0.4372	0.8674	0.9791	0.3090	0.4372	0.8674	0.9791	0.3090	0.4372
10%	0.8686	0.9747	0.3382	0.4618	0.8690	0.9771	0.3285	0.4553	0.8670	0.9757	0.3236	0.4478	0.8686	0.9762	0.3309	0.4564	0.8516	0.9971	0.1241	0.2179
20%	0.8706	0.9718	0.3650	0.4847	0.8719	0.9732	0.3650	0.4870	0.8686	0.9757	0.3333	0.4582	0.8727	0.9698	0.3869	0.5032	0.8491	0.9985	0.1022	0.1842
30%	0.8743	0.9669	0.4112	0.5216	0.8739	0.9718	0.3844	0.5040	0.8686	0.9752	0.3358	0.4600	0.8727	0.9669	0.4015	0.5124	0.8520	0.9985	0.1192	0.2117
40%	0.8767	0.9655	0.4331	0.5394	0.8731	0.9684	0.3966	0.5102	0.8690	0.9742	0.3431	0.4661	0.8715	0.9640	0.4088	0.5145	0.8581	0.9927	0.1849	0.3028
50%	0.8775	0.9582	0.4745	0.5636	0.8735	0.9655	0.4136	0.5215	0.8690	0.9737	0.3455	0.4679	0.8715	0.9630	0.4136	0.5175	0.8398	0.9990	0.0438	0.0835
60%	0.8755	0.9533	0.4866	0.5658	0.8735	0.9645	0.4185	0.5244	0.8686	0.9727	0.3479	0.4689	0.8706	0.9616	0.4161	0.5174	0.8593	0.9859	0.2263	0.3490
70%	0.8792	0.9509	0.5207	0.5895	0.8735	0.9664	0.4088	0.5185	0.8682	0.9727	0.3455	0.4663	0.8743	0.9640	0.4258	0.5303	0.8524	0.9951	0.1387	0.2385
80%	0.8788	0.9489	0.5280	0.5921	0.8727	0.9645	0.4136	0.5199	0.8694	0.9732	0.3504	0.4721	0.8731	0.9582	0.4477	0.5404	0.8593	0.9898	0.2068	0.3288
90%	0.8783	0.9421	0.5596	0.6053	0.8723	0.9616	0.4258	0.5263	0.8698	0.9718	0.3601	0.4797	0.8723	0.9586	0.4404	0.5347	0.8459	0.9985	0.0827	0.1518

xxxv.SVM + RUS 30% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8690	0.9762	0.3333	0.4590	0.8690	0.9762	0.3333	0.4590	0.8690	0.9762	0.3333	0.4590	0.8690	0.9762	0.3333	0.4590	0.8690	0.9762	0.3333	0.4590
10%	0.8706	0.9727	0.3601	0.4813	0.8706	0.9732	0.3577	0.4796	0.8670	0.9737	0.3333	0.4551	0.8731	0.9703	0.3869	0.5040	0.8500	0.9966	0.1168	0.2060
20%	0.8739	0.9669	0.4088	0.5193	0.8723	0.9689	0.3893	0.5039	0.8682	0.9737	0.3406	0.4628	0.8727	0.9674	0.3990	0.5109	0.8479	0.9976	0.0998	0.1794
30%	0.8755	0.9616	0.4453	0.5438	0.8731	0.9669	0.4039	0.5147	0.8678	0.9723	0.3455	0.4656	0.8715	0.9645	0.4063	0.5131	0.8382	0.9990	0.0341	0.0656
40%	0.8751	0.9552	0.4745	0.5587	0.8710	0.9625	0.4136	0.5167	0.8682	0.9718	0.3504	0.4698	0.8710	0.9596	0.4282	0.5254	0.8516	0.9971	0.1241	0.2179
50%	0.8731	0.9465	0.5061	0.5706	0.8723	0.9620	0.4234	0.5249	0.8690	0.9718	0.3552	0.4748	0.8694	0.9547	0.4428	0.5306	0.8670	0.9752	0.3260	0.4497
60%	0.8743	0.9411	0.5401	0.5889	0.8723	0.9596	0.4355	0.5319	0.8702	0.9718	0.3625	0.4822	0.8686	0.9513	0.4550	0.5358	0.8479	0.9976	0.0998	0.1794
70%	0.8763	0.9421	0.5474	0.5960	0.8710	0.9538	0.4574	0.5418	0.8710	0.9718	0.3674	0.4871	0.8706	0.9513	0.4672	0.5462	0.8354	1.0000	0.0122	0.0240
80%	0.8771	0.9387	0.5693	0.6070	0.8719	0.9543	0.4599	0.5447	0.8735	0.9718	0.3820	0.5016	0.8731	0.9509	0.4842	0.5598	0.8751	0.9596	0.4526	0.5471
90%	0.8800	0.9372	0.5937	0.6224	0.8723	0.9547	0.4599	0.5455	0.8739	0.9718	0.3844	0.5040	0.8710	0.9499	0.4769	0.5521	0.8443	0.9976	0.0779	0.1429

xxxvi.SVM + RUS 40% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8723	0.9684	0.3917	0.5055	0.8723	0.9684	0.3917	0.5055	0.8723	0.9684	0.3917	0.5055	0.8723	0.9684	0.3917	0.5055	0.8723	0.9684	0.3917	0.5055
10%	0.8735	0.9645	0.4185	0.5244	0.8723	0.9650	0.4088	0.5161	0.8727	0.9664	0.4039	0.5139	0.8715	0.9616	0.4209	0.5219	0.8427	0.9976	0.0681	0.1261
20%	0.8747	0.9586	0.4550	0.5476	0.8706	0.9596	0.4258	0.5232	0.8743	0.9659	0.4161	0.5245	0.8719	0.9577	0.4428	0.5353	0.8459	0.9985	0.0827	0.1518
30%	0.8759	0.9543	0.4842	0.5653	0.8706	0.9557	0.4453	0.5343	0.8747	0.9655	0.4209	0.5282	0.8731	0.9562	0.4574	0.5457	0.8573	0.9917	0.1849	0.3016
40%	0.8739	0.9479	0.5036	0.5710	0.8715	0.9547	0.4550	0.5412	0.8747	0.9645	0.4258	0.5311	0.8727	0.9528	0.4720	0.5527	0.8532	0.9951	0.1436	0.2458
50%	0.8739	0.9397	0.5450	0.5903	0.8723	0.9528	0.4696	0.5506	0.8751	0.9630	0.4355	0.5375	0.8715	0.9499	0.4793	0.5541	0.8435	0.9981	0.0706	0.1306
60%	0.8759	0.9397	0.5572	0.5995	0.8727	0.9513	0.4793	0.5565	0.8767	0.9606	0.4574	0.5529	0.8710	0.9499	0.4769	0.5521	0.8597	0.9898	0.2092	0.3320
70%	0.8771	0.9358	0.5839	0.6130	0.8719	0.9509	0.4769	0.5537	0.8751	0.9620	0.4404	0.5403	0.8715	0.9499	0.4793	0.5541	0.8581	0.9937	0.1800	0.2972
80%	0.8783	0.9290	0.6253	0.6314	0.8755	0.9513	0.4964	0.5706	0.8759	0.9616	0.4477	0.5460	0.8735	0.9460	0.5109	0.5738	0.8577	0.9912	0.1898	0.3077
90%	0.8702	0.9153	0.6448	0.6235	0.8747	0.9499	0.4988	0.5702	0.8763	0.9611	0.4526	0.5495	0.8715	0.9450	0.5036	0.5663	0.8496	0.9966	0.1144	0.2022

xxxvii.SVM + RUS 50% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8723	0.9572	0.4477	0.5388	0.8723	0.9572	0.4477	0.5388	0.8723	0.9572	0.4477	0.5388	0.8723	0.9572	0.4477	0.5388	0.8723	0.9572	0.4477	0.5388
10%	0.8719	0.9518	0.4720	0.5511	0.8743	0.9543	0.4745	0.5571	0.8735	0.9552	0.4647	0.5504	0.8723	0.9509	0.4793	0.5557	0.8621	0.9878	0.2336	0.3609
20%	0.8747	0.9465	0.5158	0.5784	0.8719	0.9474	0.4939	0.5623	0.8755	0.9533	0.4866	0.5658	0.8710	0.9470	0.4915	0.5596	0.8662	0.9757	0.3187	0.4426
30%	0.8747	0.9431	0.5328	0.5863	0.8706	0.9460	0.4939	0.5600	0.8747	0.9513	0.4915	0.5666	0.8723	0.9460	0.5036	0.5679	0.8552	0.9956	0.1533	0.2609
40%	0.8735	0.9353	0.5645	0.5979	0.8727	0.9450	0.5109	0.5722	0.8755	0.9518	0.4939	0.5694	0.8723	0.9436	0.5158	0.5737	0.8451	0.9976	0.0827	0.1511
50%	0.8735	0.9294	0.5937	0.6100	0.8735	0.9421	0.5304	0.5829	0.8743	0.9504	0.4939	0.5670	0.8719	0.9426	0.5182	0.5741	0.8350	0.9990	0.0146	0.0286
60%	0.8715	0.9217	0.6204	0.6167	0.8735	0.9421	0.5304	0.5829	0.8747	0.9489	0.5036	0.5726	0.8735	0.9401	0.5401	0.5873	0.8467	0.9971	0.0949	0.1711
70%	0.8690	0.9134	0.6472	0.6222	0.8735	0.9397	0.5426	0.5884	0.8751	0.9465	0.5182	0.5804	0.8735	0.9406	0.5377	0.5862	0.8487	0.9961	0.1119	0.1978
80%	0.8642	0.9061	0.6545	0.6163	0.8702	0.9367	0.5377	0.5801	0.8739	0.9474	0.5061	0.5722	0.8710	0.9367	0.5426	0.5838	0.8414	0.9976	0.0608	0.1134
90%	0.8548	0.8900	0.6788	0.6092	0.8715	0.9363	0.5474	0.5867	0.8751	0.9460	0.5207	0.5815	0.8698	0.9333	0.5523	0.5858	0.8528	0.9971	0.1314	0.2293

xxxviii.SVM + RUS 60% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8706	0.9465	0.4915	0.5588	0.8706	0.9465	0.4915	0.5588	0.8706	0.9465	0.4915	0.5588	0.8706	0.9465	0.4915	0.5588	0.8706	0.9465	0.4915	0.5588
10%	0.8702	0.9392	0.5255	0.5745	0.8682	0.9387	0.5158	0.5661	0.8723	0.9431	0.5182	0.5749	0.8698	0.9411	0.5134	0.5680	0.8560	0.9888	0.1922	0.3080
20%	0.8719	0.9353	0.5547	0.5907	0.8723	0.9397	0.5353	0.5828	0.8710	0.9397	0.5280	0.5771	0.8702	0.9372	0.5353	0.5789	0.8646	0.9810	0.2822	0.4099
30%	0.8710	0.9280	0.5864	0.6025	0.8698	0.9338	0.5499	0.5847	0.8698	0.9367	0.5353	0.5782	0.8719	0.9363	0.5499	0.5885	0.8650	0.9781	0.2993	0.4249
40%	0.8678	0.9197	0.6083	0.6053	0.8715	0.9343	0.5572	0.5910	0.8715	0.9363	0.5474	0.5867	0.8686	0.9328	0.5474	0.5814	0.8358	0.9990	0.0195	0.0380
50%	0.8629	0.9085	0.6350	0.6070	0.8698	0.9294	0.5718	0.5942	0.8682	0.9314	0.5523	0.5828	0.8690	0.9324	0.5523	0.5843	0.8625	0.9815	0.2676	0.3936
60%	0.8589	0.9041	0.6326	0.5991	0.8674	0.9280	0.5645	0.5866	0.8706	0.9333	0.5572	0.5894	0.8690	0.9304	0.5620	0.5885	0.8702	0.9304	0.5693	0.5939
70%	0.8301	0.8574	0.6934	0.5763	0.8650	0.9226	0.5766	0.5874	0.8694	0.9314	0.5596	0.5882	0.8690	0.9280	0.5742	0.5937	0.8451	0.9976	0.0827	0.1511
80%	0.8350	0.8633	0.6934	0.5834	0.8646	0.9197	0.5888	0.5917	0.8710	0.9319	0.5669	0.5944	0.8682	0.9265	0.5766	0.5932	0.2255	0.0754	0.9757	0.2957
90%	0.8260	0.8482	0.7153	0.5782	0.8613	0.9163	0.5864	0.5850	0.8710	0.9285	0.5839	0.6015	0.8658	0.9246	0.5718	0.5868	0.8625	0.9538	0.4063	0.4963

xxxix.SVM + RUS 70% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.8552	0.9119	0.5718	0.5683	0.8552	0.9119	0.5718	0.5683	0.8552	0.9119	0.5718	0.5683	0.8552	0.9119	0.5718	0.5683	0.8552	0.9119	0.5718	0.5683
10%	0.8455	0.8910	0.6180	0.5714	0.8443	0.8920	0.6058	0.5646	0.8560	0.9075	0.5985	0.5809	0.8459	0.8939	0.6058	0.5672	0.8601	0.9844	0.2384	0.3623
20%	0.8431	0.8793	0.6618	0.5843	0.8447	0.8886	0.6253	0.5730	0.8500	0.8964	0.6180	0.5786	0.8418	0.8866	0.6180	0.5657	0.8362	0.9985	0.0243	0.0472
30%	0.8090	0.8341	0.6837	0.5440	0.8358	0.8745	0.6423	0.5659	0.8516	0.8964	0.6277	0.5850	0.8394	0.8818	0.6277	0.5658	0.8394	0.9981	0.0462	0.0876
40%	0.8155	0.8404	0.6910	0.5552	0.8386	0.8764	0.6496	0.5730	0.8508	0.8939	0.6350	0.5865	0.8443	0.8822	0.6545	0.5835	0.8423	0.9903	0.1022	0.1776
50%	0.7944	0.8073	0.7299	0.5420	0.8333	0.8691	0.6545	0.5669	0.8512	0.8934	0.6399	0.5890	0.8317	0.8667	0.6569	0.5654	0.8593	0.9494	0.4088	0.4919
60%	0.7875	0.7937	0.7567	0.5428	0.8167	0.8467	0.6667	0.5480	0.8431	0.8827	0.6448	0.5780	0.8317	0.8672	0.6545	0.5645	0.8049	0.8161	0.7494	0.5615
70%	0.7624	0.7630	0.7591	0.5157	0.8200	0.8496	0.6715	0.5542	0.8532	0.8900	0.6691	0.6031	0.8264	0.8594	0.6618	0.5597				
80%	0.7470	0.7392	0.7859	0.5087	0.8212	0.8501	0.6764	0.5577	0.8341	0.8652	0.6788	0.5770	0.8268	0.8579	0.6715	0.5638				
90%	0.7384	0.7255	0.8029	0.5057	0.8159	0.8433	0.6788	0.5514	0.8289	0.8589	0.6788	0.5694	0.8252	0.8545	0.6788	0.5642				

xl.SVM + RUS 80% + Oversampling

	SVM																			
	SMOTE				ADASYN				ANS				B-SMOTE				SVM-SMOTE			
OR	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1	A	R0	R1	F1
0%	0.7818	0.7937	0.7226	0.5247	0.7818	0.7937	0.7226	0.5247	0.7818	0.7937	0.7226	0.5247	0.7818	0.7937	0.7226	0.5247	0.7818	0.7937	0.7226	0.5247
10%	0.7656	0.7669	0.7591	0.5191	0.7628	0.7630	0.7616	0.5169	0.7741	0.7771	0.7591	0.5284	0.7616	0.7620	0.7591	0.5149	0.8548	0.9839	0.2092	0.3245
20%	0.7340	0.7221	0.7932	0.4985	0.7372	0.7304	0.7713	0.4945	0.7567	0.7552	0.7640	0.5114	0.7409	0.7333	0.7786	0.5004				
30%	0.7076	0.6856	0.8175	0.4824	0.7413	0.7324	0.7859	0.5031	0.7518	0.7479	0.7713	0.5088	0.7510	0.7426	0.7932	0.5150				
40%	0.6853	0.6584	0.8200	0.4648	0.7336	0.7202	0.8005	0.5004	0.7482	0.7406	0.7859	0.5099	0.7307	0.7178	0.7956	0.4962				
50%	0.6752	0.6394	0.8540	0.4671	0.7247	0.7090	0.8029	0.4929	0.7393	0.7290	0.7908	0.5027	0.7344	0.7226	0.7932	0.4989				
60%	0.6573	0.6180	0.8540	0.4538	0.7202	0.7046	0.7981	0.4874	0.7324	0.7187	0.8005	0.4992	0.7291	0.7105	0.8224	0.5030				
70%	0.6403	0.5888	0.8978	0.4542	0.7080	0.6881	0.8078	0.4798	0.7283	0.7139	0.8005	0.4955	0.7178	0.6993	0.8102	0.4890				
80%	0.6135	0.5547	0.9075	0.4391	0.7129	0.6910	0.8224	0.4884	0.7271	0.7114	0.8054	0.4959	0.7129	0.6915	0.8200	0.4877				
90%	0.6034	0.5416	0.9124	0.4340	0.7011	0.6803	0.8054	0.4732	0.7222	0.7056	0.8054	0.4915	0.7088	0.6881	0.8127	0.4820				