

UNIVERSITY OF HERTFORDSHIRE

Department of Computer Science

#### MSc Data Sciences and Analytics

SVM Based Oversampling Technique for Imbalanced Dataset

Name: Arshad Shaik

Supervisor: Seyedali Pourmoafi

Advisor: Dr. Na Helian

# ACKNOWLEDGEMENT

I'd like to take this opportunity to thank my supervisor, Professor Seyedali Pourmoafi, and the project advisor, Dr. Na Helian, for their invaluable advice and support that helped me finish this project.

Given that this is relatively a new field for me, I am grateful for their patience and understanding.

In addition, I'd like to thank my friends for their unwavering faith in me.

Finally, I'd like to express my gratitude to my family for their unwavering support, as this thesis would not have been possible without them.

# Abstract

The problem with imbalanced learning includes uneven distribution of data across different classes where the majority of the data samples belongs to one class and the remaining minority portion of samples to the other. To overcome this problem a synthetic oversampling method is proposed where artificial data points as generated on the minority side to match the numbers of majority data points. However, the existing oversampling algorithms have a few limitations of noise interference, data points overlapping, and neighbour selection blindness. This research aims to create an oversampling technique that is based on SVM as it can learn composite distributions. The SVM is used to draw a decision boundary between different classes and generate artificial data points along the minority side which will lead to shifting the bias towards the minority class. Furthermore, the proposed oversampling algorithm is evaluated using a confusion matrix, F1 score, recall, and accuracy. Additionally, the results are compared with existing oversampling techniques such as Adaptive Synthetic Sampling Approach (ADASYN) and Synthetic Minority Oversampling Technique (SMOTE). Moreover, these experiments are performed on two artificially generated datasets where the overlap, shape, size, orientation, and separation of the majority and minority samples are fully controlled. This will help us understand how different parameters affect the existing as well as proposed oversampling techniques. The proposed algorithm provides a balanced performance which is in middle of SMOTE and ADASYN, however ADASYN provides better minority performance when compared to other techniques.

Contents

[ACKNOWLEDGEMENT 2](#_Toc114419449)

[Final Project Declaration 3](#_Toc114419450)

[Abstract 4](#_Toc114419451)

[**1.** Introduction: 9](#_Toc114419452)

[1.1 Background Research 9](#_Toc114419453)

[**1.1.1** Description of Imbalance Data 9](#_Toc114419454)

[1.2 Research Aim and Objective: 10](#_Toc114419455)

[1.3 Aim and Objectives: 10](#_Toc114419456)

[2 Literature Review: Increase the size of this section. 11](#_Toc114419457)

[2.1 Modelling: 13](#_Toc114419458)

[2.2 Evaluation : 14](#_Toc114419459)

[3 Methodology: 15](#_Toc114419460)

[3.1 Flow Chart: 15](#_Toc114419461)

[3.2 Generating Synthetic Imbalance Dataset: Move this to implementation 16](#_Toc114419462)

[3.3 Exploratory Data Analysis (EDA) / Data Pre-processing: 16](#_Toc114419463)

[3.4 Dataset Split: 16](#_Toc114419464)

[3.5 Classification models: 16](#_Toc114419465)

[3.5.1 Naïve Bayes: 16](#_Toc114419466)

[3.5.2 Multi-Layer Perceptron (MLP): 17](#_Toc114419467)

[3.5.3 Support Vector Machines (SVM): 18](#_Toc114419468)

[3.6 Evaluation: 19](#_Toc114419469)

[3.7 Oversampling Algorithms: 20](#_Toc114419470)

[3.7.1 SMOTE: 20](#_Toc114419471)

[3.7.2 ADASYN: 21](#_Toc114419472)

[3.8 Proposed Algorithm: 22](#_Toc114419473)

[3.8.1. SVM-based oversampling 22](#_Toc114419474)

[4 Implementation: 23](#_Toc114419475)

[4.1 Tools and Techniques: 23](#_Toc114419476)

[4.2 Artificial Data Generation: 24](#_Toc114419477)

[4.3 Data Processing and Exploratory Data Analysis (EDS): 26](#_Toc114419478)

[4.4 Oversampling: 28](#_Toc114419479)

[4.5 Model Training and Evaluation: 34](#_Toc114419480)

[5 Experiments And Results: 36](#_Toc114419481)

[5.1 Phase I – Multiple degrees of overlapping: 36](#_Toc114419482)

[5.2 Phase II – Modifying Gamma: 43](#_Toc114419483)

[5.3 Experiment Phase III – Modifying Decision Boundary: 50](#_Toc114419484)

[5 Conclusion and Future Work: 54](#_Toc114419485)

[7 Referencing: 55](#_Toc114419486)

[8 Appendices: 56](#_Toc114419487)

List Of Figures:

[Figure 1 Imbalance Data Example 9](file:///C:\Users\imsha\Desktop\University%20docx\docs\19050630-Arshad-Shaik(FPR).docx#_Toc114419499)

[Figure 2 A Flow chart for the research process 15](#_Toc114419500)

[Figure 3 Data Distribution 16](file:///C:\Users\imsha\Desktop\University%20docx\docs\19050630-Arshad-Shaik(FPR).docx#_Toc114419501)

[Figure 4 MLP Network 17](#_Toc114419502)

[Figure 5 SMOTE Oversampling 21](#_Toc114419503)

[Figure 6 ADASYN Oversampling 22](#_Toc114419504)

[Figure 7 Proposed Algorithm Oversampling (Gamma = 0.1, C = 0.1) 22](file:///C:\Users\imsha\Desktop\University%20docx\docs\19050630-Arshad-Shaik(FPR).docx#_Toc114419505)

[Figure 8 Circular Scatter Plot 25](#_Toc114419506)

[Figure 9 Thick Flat Line Scatter Plot 25](#_Toc114419507)

[Figure 10 SMOTE Circular Scatter Plot 28](#_Toc114419508)

[Figure 11 Thick Flat Line Scatter Plot 29](#_Toc114419509)

[Figure 12 ADASYN Circular Oversampling 30](#_Toc114419510)

[Figure 13 ADASYN Thick Flat Line Oversampling 30](#_Toc114419511)

[Figure 14 Proposed Algorithm Circular Oversampling 33](#_Toc114419512)

[Figure 15 Proposed Algorithm Thick Flat Line Oversampling 34](#_Toc114419513)

[Figure 16 Decision Boundary (Circular) 43](#_Toc114419514)

[Figure 17 Modified Gamma Scatter Plot (Circular) 44](#_Toc114419515)

[Figure 18 Modified Gamma Scatter Plot (Thick Flat Line) 47](#_Toc114419516)

[Figure 19 Decision Boundary (Circular) 51](#_Toc114419517)

[Figure 20 Decision boundary Flat Line 52](#_Toc114419518)

List Of Tables:

[Table 1 List of Research Papers 12](#_Toc114419344)

[Table 2 Confusion Matrix Naïve Bayes (Circular) 38](#_Toc114419345)

[Table 3 Recall Score Naïve Bayes (Circular) 38](#_Toc114419346)

[Table 4 F1 Score Naïve Bayes (Circular) 39](#_Toc114419347)

[Table 5 Confusion Matrix SVM (Circular) 39](#_Toc114419348)

[Table 6 Recall Scores SVM (Circular) 39](#_Toc114419349)

[Table 7 F1 Scores SVM (Circular) 39](#_Toc114419350)

[Table 8 Confusion Matrix MLP (Circular) 40](#_Toc114419351)

[Table 9 Recall Scores MLP (Circular) 40](#_Toc114419352)

[Table 10 F1 Scores MLP (Circular) 40](#_Toc114419353)

[Table 11 Confusion Matrix Naïve Bayes (Thick Flat Line) 41](#_Toc114419354)

[Table 12 Recall Scores Naïve Bayes (Thick Flat Line) 41](#_Toc114419355)

[Table 13 F1 Scores Naïve Bayes (Thick Flat Line) 41](#_Toc114419356)

[Table 14 Confusion Matrix SVM (Thick Flat Line) 42](#_Toc114419357)

[Table 15 Recall Scores SVM (Thick Flat Line) 42](#_Toc114419358)

[Table 16 F1 Scores SVM (Thick Flat Line) 42](#_Toc114419359)

[Table 17 Confusion Matrix MLP (Thick Flat Line) 43](#_Toc114419360)

[Table 18 Recall Scores MLP (Thick Flat Line) 43](#_Toc114419361)

[Table 19 F1 Scores MLP (Thick Flat Line) 43](#_Toc114419362)

[Table 20 Modified Gamma Scatter Plot 47](#_Toc114419363)

[Table 21 Decision Boundary (Thick Flat Line) 47](#_Toc114419364)

[Table 22 Confusion Matrix Decision Boundary 52](#_Toc114419365)

[Table 23 Recall Scores Decision Boundary 52](#_Toc114419366)

[Table 24 F1 Scores Decision Boundary (Circular) 53](#_Toc114419367)

[Table 25 Confusion Matrix Decision Boundary 53](#_Toc114419368)

[Table 26 Recall Score Decision Boundary (Thick Flat Line) 54](#_Toc114419369)

[Table 27 Recall Score Decision Boundary (Thick Flat Line) 54](#_Toc114419370)

# Introduction:

## 1.1 Background Research

Classification is one of the most common and widely used machine learning techniques. To perform prediction on Classification models, it studies the labeled input data or predictor and tries to predict the target value where the target value is categorical.

While building a classification model the most repetitive instance faced is when a target label class contains a class with higher class labels while the other class is significantly lower-class labels, This phenomenon leads to a case called imbalanced class dataset that is very common on a real-world dataset where the data is biased towards a class that leads to incorrect/ biased/inappropriate predictions for the machine learning models.

Further in the report, we will discuss what is an imbalanced dataset in detail, the problems faced during predictions on these datasets, and how effectively deal with this kind of data using different approaches.

### 1.1.1 Description of Imbalance Data

When a dataset that contains a categorical target class is distributed in a dissimilar fashion, i.e one majority class that contains a greater number of observations and a minority class that contains a lower number of observations that case is referred to as an imbalanced dataset.

Example:

Let us consider a bank “ABC” that provides a credit card to its consumers. Additionally, this back is concerned about some fraudulent transactions going on upon analyzing their data the bank found that for every 2000 transactions there are around 30 fraudulent transactions. In other words for every 100 transactions, there are nearly 2% fraudulent transactions or 98% of exchanges are fraud-free. In this case, the fraud-free(98%) class is known as the **majority class,** while the other smaller portion which contains data on “Fraud” is known as the **minority class.**

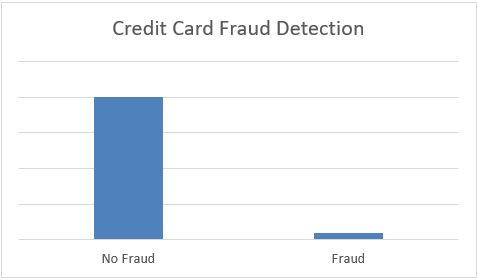


Figure Imbalance Data Example

More such examples of imbalanced datasets are

· Disease diagnosis

· Customer churn prediction

· Spam Email Detection

To solve this problem, several pre-processing techniques can be used such as oversampling or under sampling, In the oversampling method the minority samples are increased to match the amount of the majority samples, however in an under-sampling method the majority samples is reduced to have same number of the minority samples in a dataset. To perform oversampling there are multiple algorithms such as Random oversampling, Adaptive Synthetic Sampling Approach (ADASYN), and Synthetic Minority Oversampling Technique (SMOTE). To justify this algorithm Random oversampling copies a random set of minority classes and expands to match the majority class this algorithm will lead to overfitting of the model resulting in decreased efficiency, Furthermore SMOTE and ADASYN tends to generate synthetic data points in which SMOTE chooses samples in a feature space that are closer making a line in the space between the samples and generating synthetic data points along the line. Respectively ADASYN to determine the number of synthetic data points it uses density distribution for every minority data point. while for under-sampling, there is one proposed algorithm Random Under-sampling which randomly reduces the majority samples to match the number of minority instances.

## 1.2 Research Aim and Objective:

My dissertation research question is as follows:

**“***What are the advantages and disadvantages of the proposed oversampling method in comparison to the existing oversampling methods?***”**

***“****Can we increase the accuracy of the proposed algorithm by changing the Gamma and Decision boundary value?”*

## 1.3 Aim and Objectives:

The main aim is to create a new oversampling algorithm based on SVM and match up the results of the proposed oversampling algorithm with existing oversampling algorithms on the artificially generated dataset.

Objectives:

* Go through previous research papers.
* Work on creating a new oversampling algorithm (SVM Oversampling).
* Use synthetic data sets to test the performance of the SVM based algorithm.
* Evaluate the oversampling algorithms using multiple measuring metrics such as confusion matrix, F1 score, Recall, and Accuracy.
* Compare the results with existing oversampling techniques, namely, SMOTE and ADASYN oversampling.
* Document and conclude the findings.

# 2 Literature Review: Increase the size of this section.

## 2.1 Oversampling:

The following papers provide different aspects research performed in the area of oversampling.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PAPER | YEAR | DATASET | PRE-PROCESSING (Oversampling) | MODELLING | EVALUATION |
| Classification of Imbalanced Data by Oversampling  in Kernel Space of Support Vector Machines | 2018 | Multiple Benchmark datasets from UCI and KEEL | SMOTE, ADASYN, Borderline, PI-SMOTE, WK-SMOTE | SVM | Accuracy, F1 score, G mean. |
| Kernel-Based SMOTE for SVM Classification of  Imbalanced Datasets | 2015 | Benchmark datasets: 51 classification datasets from the data repository of KEEL | SMOTE, K-SMOTE | SVM kernel(RBF) | G-Mean, Accuracy, F1 |
| AWSMOTE: An SVM-Based Adaptive Weighted SMOTE for  Class-Imbalance Learning | 2021 | real-world datasets from UCI, and KEEL | NO-RS, SMOTE, BLSMOTE, ADASYN, DBSMOTE, AWSMOTE | SVM | F-Measure, Accuracy, AUC, Precision, G-mean. |
| Improved Oversampling Algorithm Based on the Samples’ Selection Strategy for Classifying | 2019 | Datasets are imported from UCI, and KEEL Library | SMOTE, Random SMOTE, AKN Random SMOTE | SVM | Precision, Recall, F-measure, F1, G-mean  AUC, ROC |
| Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning | 2005 | Synthetic Circular Dataset, Pima(UCI), Satimage(UCI), Haberman(UCI) | SMOTE | SVM,  ADA Booster, | Confusion Matrix, Accuracy, FP rate, TP rate, Precision. |
| SMOTE: Synthetic Minority Over-sampling Technique | 2002 | Pima(UCI),  Phoneme,  Adult, E-state,  Satimage,  Forest Cover,  Oil,  Mammography,  Can. | Producing synthetic data in feature space on the line connecting each minority class. | Naïve Bayes | AUC, ROC Convex Hull |
| ADASYN:  Adaptive  Synthetic  Sampling  Approach for  Imbalanced  Learning | 2008 | Vehicle, Pima,  Vowel  recognition,  Ionosphere,  Abalone,  Winning times | Generating Synthetic samples depending on the density distribution | Decision Tree (DT) | Confusion matrix, ROC curve |

Table List of Research Papers

This research on Classification of Imbalanced Data by Oversampling in Kernel Space of Support Vector Machines (J. Mathew *et al*, 2018) was performed by adding weights to the SMOTE kernel and checking for the performance by comparing them with regular SMOTE while observing if the weighted SMOTE overcomes the limitations of SMOTE. The proposed algorithm outperforms when compared to baseline methods. Furthermore, to deal with a multi-class imbalanced dataset that has a progressive order a framework was developed known as a hierarchical framework.

In this report on Kernel-Based SMOTE for SVM Classification of Imbalanced Datasets (J. Mathew *et al*, 2015) research was performed using a kernel where Kernel-based SMOTE generates Synthetic data points directly in the feature space of the SVM classifier to overcome the limitations of SMOTE that gives a distorted performance for Support vector machines which works in kernel-based induced feature space. Kernel-based SMOTE proposed to generate data points by augmenting the original Gram Matrices depending on the neighboring point's information in the feature space. The proposed oversampling algorithm shows improved performance on 51 benchmark datasets when compared to traditional oversampling methods.

This paper on AWSMOTE: An SVM-Based Adaptive Weighted SMOTE for Class Imbalance Learning (J.B. Wang *et al*, 2021) was performed to apply an SVM Based adaptive weights to SMOTE where the weights of the newly generated points were calculated using SVM and the points with lower weight scores were dropped. The proposed algorithm divides the minority class into two Support vectors and nonsupport vectors as per SVM assigning weight to those vectors. The additional weights on the vectors are determined by the accuracy of the model, Higher the accuracy higher the weight on the support vectors. Initially to make the minority points that are support vectors more important specific weights are added with the non-support vector remaining the same this will result in more points near the decision boundary. This research is performed on multiple datasets and works significantly well when compared to existing oversampling techniques.

This paper on Improved Oversampling Algorithm Based on the Samples’ Selection Strategy for Classifying (W. Xie. *et al*, 2019) was performed to overcome the drawbacks of blind interpolation, and fuzzy class boundaries processed by the SMOTE by applying random-SMOTE where the algorithm generates support vectors from the parent samples and generates a new minority class from those points. The dataset is classified using SVM classification while the evaluations are done by using F1, Accuracy, G-mean, etc.

This paper on borderline SMOTE (H. Han *et al*., 2005) shines a light on how most of the classification models work by attempting to learn the borderline of each class(the points near and on the borderline) thus two oversampling algorithms were introduced borderline SMOTE1 and borderline SMOTE2. The algorithm (borderline SMOTE1) works similarly to SMOTE by creating K nearest neighbors for the minority class. Additionally, the algorithm is focused to create the oversampling points to strengthen the borderline of the minority class. Additionally, borderline SMOTE2 works by generating data points for its positive nearest neighbors and from the negative nearest neighbors while the difference between the negative and positive point is multiplied with a random number between 0 and 0.5 generating new data points closer to the minority class.

In this paper (N. Chawla *et al.*, 2002) a study on SMOTE was performed with a combination of minority oversampling techniques and under-sampling techniques on the majority class to raise the performance of a classification model. It is demonstrated that a combined strategy of over and under-sampling the majority and minority classes may lead to better performance for the classification model when compared to adjusting the loss ratio in the class priors for Naïve Bayes. C4.5, Naïve Bayes, and a Ripper are used to carry out the experiments on Area under the ROC (Receiver Operating Character Curve) along with ROC convex hull to evaluate the effectiveness of the given method.

This paper (Haibo He, 2008) on ADASYN uses a basic concept of weight distribution in different minority classes depending on the difficulty in learning. To elaborate, the algorithm works by creating more synthetic points that are more difficult to learn by doing so it is making those example points easier to learn and it helps in changing the decision boundary making it more inclined towards difficult data points which helps in reducing the bias that is caused by imbalance dataset. The proposed algorithm was evaluated using 5 datasets from the machine learning library of UCI. To evaluate the performance of the propped algorithm the datasets were converted into a two-class problem. The proposed value was assigned the value of k as 5 and with the desired ratio of 1:1. However, to compare the results SMOTE oversampling was used with the ka value being 5. The decision tree was chosen as the classification model to be trained on the data, and the evaluation of the models was done using a confusion matrix and the ROC curve plot. It is observed that the proposed algorithm (ADASYN) has outperformed the existing algorithm (SMOTE) in terms of the confusion matrix value, On the other hand, this study was only performed on the two-class problem and was not tested on multiple class problems.

## 2.2 Modelling:

(S and Dr. S V, 2014) This paper on support vector machines (SVM) provides a detailed explanation of the use of support Vectors Machine for binary task classification. The support vector machines use hyperplanes to separate defining support vector machines as a discriminative classifiers. The algorithm works on generating a hyperparameter that separates one class from another. This paper provides highly sensible incites on how the synthetic dataset is used to perform oversampling and deal with the binary classification problem. As SVM is very versatile and does not work very well with SMOTE. I am going to use this to test if the proposed model overcomes the limitation when trained with SVM

(Webb, G.I. (2011)) This paper on Naïve Bayes provides an detailed working of the classification models. It is a simple learning algorithm which used Bayes rule. This model is computationally efficient due to this it most widely used classification model. I am using this algorithm to check how the proposed oversampling algorithm works with the classification model.

## 2.3 Evaluation :

In papers [1] (J. Mathew *et al*, 2018) and [2] (J. Mathew *et al*, 2015) the performance was evaluated based on Accuracy, F1 score, and G-mean. SVM was used as the classification model to train and test the dataset. Based on the research the proposed algorithm provides better results

In paper [3] (J.B. Wang *et al*, 2021) the performance was evaluated based on F-Measure, Accuracy, AUC, Precision, and G-mean. SVM was used as the classification model to train and assess the dataset.

In paper [4] (W. Xie. *et al*, 2019) the performance was evaluated based on Precision, Recall, F-measure, F1, G-mean, AUC, and ROC. SVM was used as the classification model to train and assess the dataset.

In paper [5] (H. Han *et al*., 2005) the performance was evaluated based on Confusion Matrix, Accuracy, FP rate, TP rate, and Precision. SVM and ADA Booster were used as the classification model to train and assess the dataset.

In paper [6] (N. Chawla *et al.*, 2002) the performance was evaluated based on AUC, ROC Convex Hull. Naïve Bayes was used as the classification model to train and assess the dataset

In paper [7] (Haibo He, 2008) the performance was evaluated based on Confusion Matrix, Accuracy, FP rate, TP rate, and Precision. SVM and ADA Booster were used as the classification model to train and assess the dataset

# 3 Methodology:

## 3.1 Flow Chart:

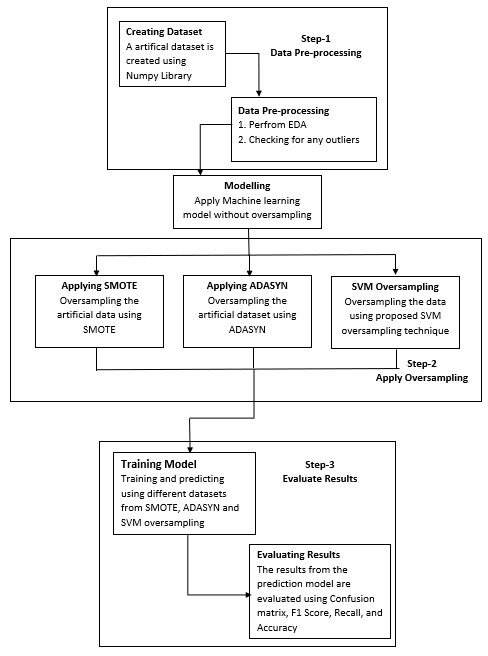


Figure A Flow chart for the research process

## 3.2 Generating Synthetic Imbalance Dataset: Move this to implementation

The research is performed using multiple different types of data few of the examples are shown in the next section. The shape of the dataset is controlled using the Covariance Matrix. Furthermore, two more variables are passed into the function separation and overlapping where separation provides a gap between the minority and majority, while overlapping provides the amount of percentage where the majority class overlaps with the minority class.

In the given four datasets the majority class contains 2000 data points and the minority contains 250 creating a ratio of 10:1, the separation and overlap are kept at 10 and 60 percent respectively.

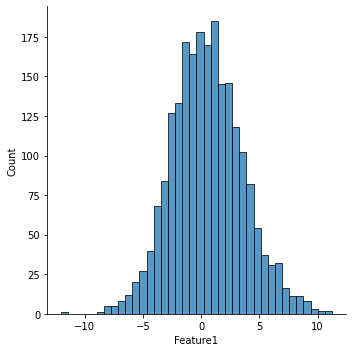
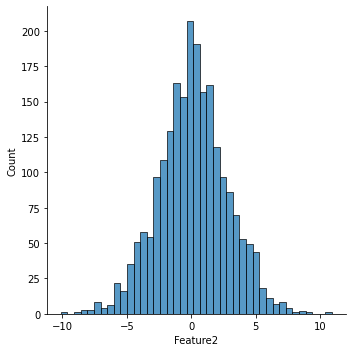
Additionally, the Distribution for all the datasets is maintained as normal /gaussian distribution as most of the real-world data are bell-shaped.

Figure Data Distribution

## 3.3 Exploratory Data Analysis (EDA) / Data Pre-processing:

As the dataset is generated artificially the dataset does not contain any empty/ unwanted data points. Furthermore, the only pre-processing performed on the dataset is to find any outlier in features or columns and drop those cases, and the model it testing with drop different

## 3.4 Dataset Split:

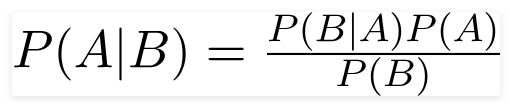
To evaluate the performance of the machine learning algorithms a train-test split procedure is used where most commonly the data is split in 80-20 configurations where 80 percent is for train purposes and 20 percent is for testing the model.

A machine learning library known as scikit-learn is used to implement the train-test split by calling a function “train\_test\_split()” and passing the dependent and target value separately as parameters.

## 3.5 Classification models:

Various classification models can be used to check the accuracy of the newly proposed oversampling algorithm, However, for this report, Naïve Bayes, Support Vector Machine (SVM), and Multi-layer perceptron (MLP) are mainly used to test the accuracy of the new oversampling algorithm, but in the future, we can apply more classification models to check how the proposed algorithm works with them.

##### 3.5.1 Naïve Bayes:



This algorithm works very well with spam filtering, sentiment analysis, etc. The advantage of this model is that it has low time complexity making it faster and easy to implement as this model does not require any hyperparameter tuning. however, the biggest setback of this model is that it requires the predictors to be independent, while in most real life datasets predictors are dependent.

There are three types of models Naïve Bayes Multinomial Naïve Bayes, Bernoulli Naïve Bayes, and Gaussian Naïve Bayes, for the experimentation of our project I am using Gaussian Naïve Bayes as the distribution of the dataset is in gaussian distribution and the predictor is independent by nature.

##### 3.5.2 Multi-Layer Perceptron (MLP):

**Multi-layer Perceptron (MLP)** is one of the supervised learning algorithms. The algorithm's capability of predicting comes from a multi-layered or hierarchical structure of the network. The features at different resolutions or scales can be selected by the data structure. The main building block of MLP is artificial neurons.

**Neurons** are simple computing units that take a weighted input signal and generate an output by using activation functions such as Sigmoid, Relu, etc.

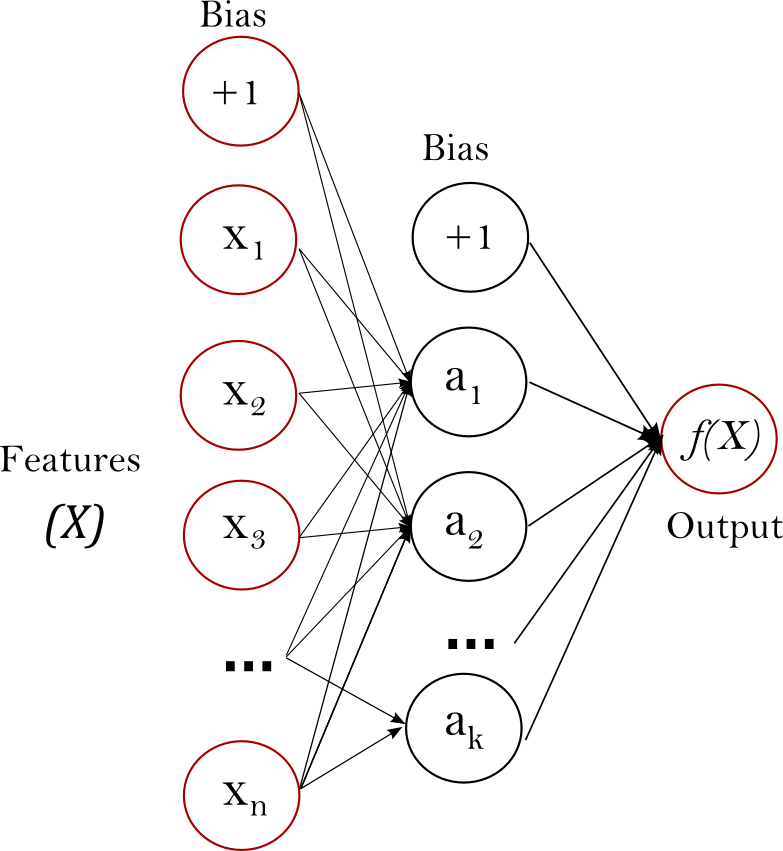
[](https://scikit-learn.org/stable/_images/multilayerperceptron_network.png)

Figure MLP Network

A group of neurons combines to form a network where a network contains three essential layers Input layer, Hidden or Inner Layer, and an Output Layer.

**Input or Visible Layer:**

The layer from the extreme left of the above figure is an input layer, it takes input from the dataset directly making it the exposed part of the network. The input layer simply passes the input data to the next layer.

**Hidden or Inner Layer:**

The middle layer after the input layer is known hidden layer as this part of the network is not directly exposed externally, this layer takes the values from the prior layer and transforms them using weighted linear aggregation along with the activation function.

**Output Layer:**The last layer of the network is known as the output layer, it corresponds to providing the final output value or vector of values of the network, and the selection of activation function at output layers plays an important role in the type of problem for modeling i.e regression or classification. For example. The regression problem may have a single neuron as an output layer with no activation function. A classification problem may have a single output neural with the activation function being Sigmoid.

##### 3.5.3 Support Vector Machines (SVM):

A support vector machine works by creating a hyperplane or collection of hyperplanes in multi-dimensions feature space so that the model can categorize the data points this process even works in non-linearly separated data, by expanding the data into multiple dimensions. This algorithm can be used for both classification and regression models. The important terms that influence a hyperplane are Kernel, Support vectors, Hyperplane, Hyperparameters (Cost and Gamma), and Margin.

1. **Kernel:**

The functions that mathematically transform the data are known as Kernel functions. The most common kernel functions are Linear, Polynomial, Radial basis Functions (RBF), and Sigmoid.

1. **Support Vectors:**

The data points that are in the neighborhood of the hyperplane due to which the orientation and the position of the hyperplane can change are known are support vectors (Gandhi, 2018).

1. **Hyperplane:**

A Decision boundary that divides the data points into different classes is known as Hyperplane. The number of features in a data frame is used to determine the dimension of the hyperplane. In other words, if the dataset contains two features the hyperplane would be a line, or else if the dataset contains three features the hyperplane will be a two-dimensional plane.

1. **Hyperparameters (Cost and Gamma):**A non-linear support vector machine (SVM) along with the kernel being a radial basis function (RBF) takes two input parameters Cost (“C”) and Gamma, C is used to control errors in SVM where lower C results in a lower number of errors and the higher C results in a higher number of errors. However, it is not the case for all the datasets for some cases higher C may result in higher accuracy. While Gamma is used to provide the amount of curvature of the decision boundary. Higher gamma results in a more curvy and complex decision boundary, whereas lower gamma results in a more straight line.
2. **Margin:**

The space between the decision boundary and the support vector is called the margin. A higher margin space gives some room for reinforcements while making it easy to classify possible data points.

## 3.6 Evaluation:

To evaluate the performance of the models before and after the oversampling there are multiple matrices used such as confusion matrix, Accuracy, F1 score, Recall, and precision.

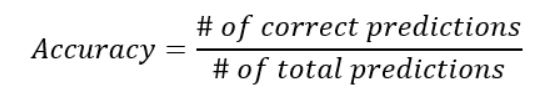
##### 3.6.1 Confusion matrix:

Table

Description automatically generatedThe confusion matrix provides the actual reading predictions for classification problems where the model predicted true and the actual value is positive they fall under True positive(TP) if the model predicted false but the actual value is positive it falls under False Negative, and if the model predicts true but the actual value is negative then this criteria fall under False Positive. Finally, if the model has predicted False and the actual value is negative then this is known as True Negative, all four of these values are used to calculate other evaluation matrices such as precision and recall.

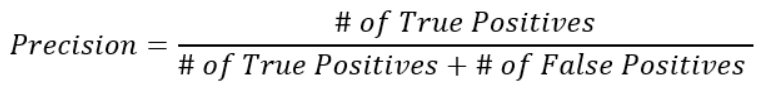
##### 3.6.2 Accuracy:

This determines the amount of correctly classified instances for a given machine learning model.



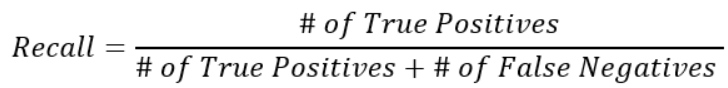
##### 3.6.3 Precision:

This is calculated by dividing the amount of true positives from matrix by the amount of true positives and number of false positives from the confusion matrix



##### 3.6.4 Recall:

The recall is calculated similarly to the precision but the only change is that the denominator is generated by adding the number of true positives and the number of false negatives



##### 3.6.5 F1 Score:

F1 Score is the amalgamation of two important matrices that is precision and recall the formula for the F1 score can be seen in below figure



## 3.7 Oversampling Algorithms:

For comparison SMOTE and ADASYN are used as benchmark oversampling techniques where we further use them as a comparison to gauge the performance of the proposed algorithm.

### 3.7.1 SMOTE:

SMOTE is used as a comparison unit against the new proposed oversampling algorithm in this project.

1. SMOTE is used to generate synthetic data points upon providing an oversampling percentage *P*

If *P* = 100, then *P* is used to get the number of artificial samples to be generated for the minority class.

Where Ngenerate gives us the number of artificial data points required to be generated.

2. SMOTE works by picking the minority class data points randomly and searching for their “**K**” nearest minority datapoint while joining those synthetic data points in the feature space to form a line. ( Nitesh V. Chawla, 2002)

4. The figures of SMOTE-generated oversamples can be observed below

Icon

Description automatically generatedChart, scatter chart

Description automatically generated

**Chart, scatter chart

Description automatically generated**

Figure SMOTE Oversampling

### 3.7.2 ADASYN:

ADASYN works on the principle of adaptively generating minority data points as per their distribution. In the case of minority samples that are difficult to interpret ADASYN tries to create more samples than those earlier to learn.

Implementation ADASYN.

**1.** Evaluate the amount of imbalance:

No. of Samples (Psamples) =No. of Majority Samples(Pmajor) /No. of minority samples (Pminor)

**2.** If Psamples < Psamplesth in that case (Psamplesth is the imbalance ratio for the highest tolerated degree)

a. Get the amount of artificial data points to be produced is defined by

I = (Pmajor - Pminor) \* δ

Where δ is a parameter that is used to specify the optimal balance level after generating

artificial data points.When δ = 1, the generalization process builds a completely balanced data set.

**3.** overview of generating all minority class data points.

a. locate K-nearest neighbors in m-dimensional space for each point relying on their Euclidean distance.

b. Next step would be to find the actual number of majority data samples in Pmajor

c. obtain the ratio by dividing Pmajor with G

sys\_ratio = Pmajor /G

d. Normalizing the given ratio from the previous formula for every minority data sample.

e. Calculate the total amount of artificial datapoint examples that are required to be generated (*ci*).

*ci =*  sys\_ratio \* I

4**.**  Below are the figures after applying ADASYN to the experimental data sets.

Chart, scatter chart

Description automatically generated

NA

(Cannot be created for Stright lines)

**Chart, scatter chart

Description automatically generated**Scatter chart

Description automatically generated

Figure ADASYN Oversampling

## 3.8 Proposed Algorithm:

### **Chart, scatter chart Description automatically generated**Chart, scatter chart Description automatically generatedChart, scatter chart Description automatically generatedChart, scatter chart Description automatically generated3.8.1. SVM-based oversampling

Figure Proposed Algorithm Oversampling (Gamma = 0.1, C = 0.1)

The main idea is to create oversampling using SVM along with increasing the accuracy of the models when compared to other oversampling techniques. Steps as per the synthetic dataset are. Firstly to transform the 2-dimensional dataframe into a multi-dimensional space using SVM kernel RBF(radial-based function) and the required hyperparameters for the model are C and Gamma. To fetch these values “Grid Search” pipeline is created to get the best set of values from the input range.

This Algorithm works by creating minority data points on the minority side of the decision boundary and setting a distance threshold with all points within the plane.

**Implementation of the proposed SVM oversampling algorithm.**

**1.** The First step would be to perform a Grid Search in the provided training dataset and get the optimum parameter of cost “C” and gamma.

1. Find the difference in number between majority and minority instances.

PointsGenerate = datamajority – dataminority.

1. To generate data points for the minority:

* Perform a for a loop until datamajority =dataminority.
* Generating artificial data points while maintaining gaussian distribution. using mean\_x and mean\_y as a center.
* Pass the generated points to the SVM model and save their result in a list
* Merge the newly generated data points with the new dataset.
* While loop for each point.
  + 1. If condition to check if the predicted value is within the given decision boundary.

If yes, then append that point to the list.

* + 1. Else

Continue

End loop

# 4 Implementation:

This project is implemented in various stages of experiments where the source code for different experimental changes for oversampling algorithm and synthetic dataset generation are added to the document.

## Tools and Techniques:

* **Jupyter Notebook**. It offers a simple web application that is used for creation as well as for sharing the computed data results, and visualizations.
* **Pandas**. This library is used to perform various data frame modifications
* **Matplotlib**. This library is used to plot graphs and other visualizations.
* **Seaborn.** This library is used to perform various visualizations.
* **Numpy** This Library is used to work with arrays and also to perform mathematical functions such as Fourier transformation, linear algebra, etc.
* **Scikit Learn** This library is used to fetch the models on which training is performed such as Naïve Bayes, Decision Tree, and Random Forest.

## 4.2 Artificial Data Generation:

Data is very essential for training the machine learning algorithms as good and structured data tends to give an accurate and more consistent machine learning model, Even if we provide millions of rows of data it would be worthless if it does not have any meaningful information. Thereby performing Exploratory Data Analysis and preprocessing is highly important to evaluate the dataset that is generated synthetically or taken from any open source.

In this research, The synthetic imbalanced dataset is generated with a normal or gaussian distribution. The dataset generated is of different orientations, overlaps, separations, and sizes that a fully controlled by two covariance matrices one for the majority and the other for the minority class. In the generated dataset, the minority class is assigned as ‘1’ while the majority class is assigned as ‘0’. The imbalance dataset has been generated using the NumPy library in python where multivariate normal and random functions are used to produce the dataset with a Gaussian distribution. whereas the sample classes, ‘0’ and ‘1’ are created using the ‘ones’ and ‘zeros’ functions from the NumPy library. the orientation and shape of the majority and minority classes are managed by using covariance. Moreover, The overlapping percentage is managed by making a rigid focal point for the majority class and a movable focal point for the minority class while making that focal point a minority means.

The formula to control the desired focal point is as follows:

***Minority Focal Point* = 0.5 + *Separation distance* \* (100 – *Overlapping percentage*) /100**

* **Separation distance:** the amount of separation between the minority and majority class provided as input by the user.
* **Overlapping Percentage:** It is the percentage at which minority samples overlap with the majority samples

Pandas library is used to convert the NumPy array into a data frame. The data frame contains three columns namely Feature\_1, Feature\_2, and the Results.

The Dataset is generated for four different overlaps 10,30,50, 70, and with 2 different shapes respectively. Their scatter plot is shown below.

***Following Input parameters are passed inside the function to obtain the desired data frame as output****:*

Minority\_class = 250 # input for number of Minority Samples

Majority\_class = 2000 # input for number of Majority Samples

Sep = 10 # Degree of Separation

Overlapping = 50 # percentage of Overlap

GraphMinLimit = -10 # to provide minimum limitations to the scatter plot

GraphMaxLimit = 20 # to provide maximum limitations to the scatter plot

MajorCov1 = [[8,0],[0,8]] # Covariance for a cicular shape for majority class

MinorCov1 = [[10,0],[0,10]] # Covariance for a cicular shape for minority class

MajorCovFlat = [[8,0],[0,1]] # Covariance for a Thicker Flat Line for majority class

MinorCovInclined = [[8,7],[7,8]] # Covariance for a Thicker Flat Line for majority class

**4.2.1 Circular Shaped Dataset:**

Scatter chart

Description automatically generated

Figure Circular Scatter Plot

**4.2.2** **Thick Flat Line Shaped Dataset:**

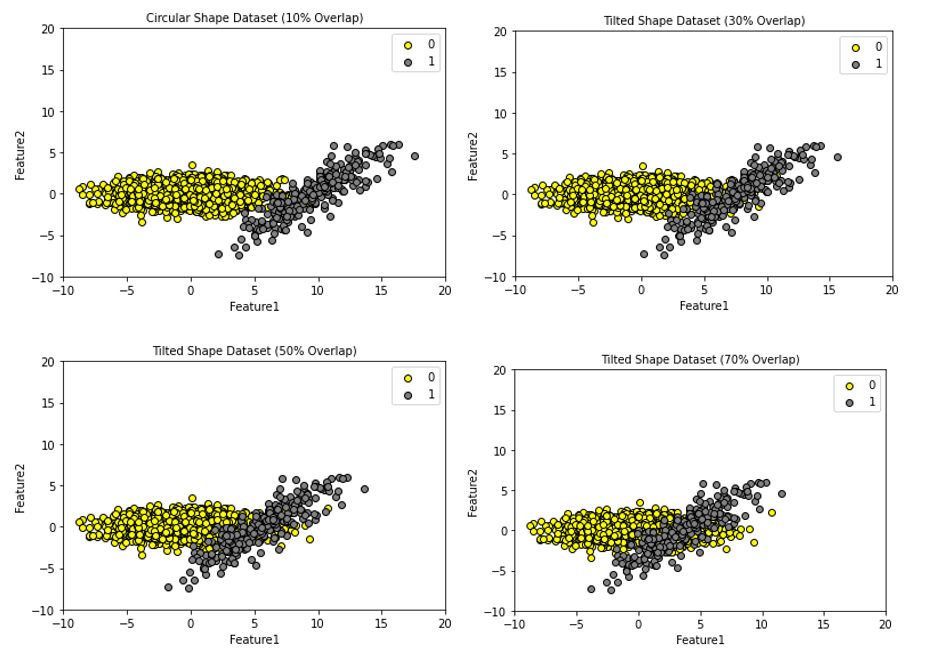


Figure Thick Flat Line Scatter Plot

***Function to create synthetic data using pandas and NumPy libraries:***

Import numpy as ny

Import pandas as pa

def SyntheticDataset(Minority\_class,Majority\_class, Sep, Overlapping,MajCov,MinCov):

ny.random.seed(0)

major\_mean = [0, 0]

major\_cov = MajCov

minor\_cov = MinCov

minor = Minority\_class

major = Majority\_class

separation = Sep

ovrlap = Overlapping

minor\_mean = 0.5 + separation \* (100 – overlap) / 100

training\_data = ny.c\_[

numpy.r\_[

ny.random.multivariate\_normal( major\_mean, major\_cov, major),

ny.random.multivariate\_normal([minor\_mean, 0], minor\_cov, minor)],

ny.r\_[ny.zeros((major, 1), dtype = int), ny.ones((minor, 1), dtype = int)],]

df = pa.DataFrame(training\_data, columns = [‘Feature1’, ‘Feature2’, ‘Results’])

return df

## Data Processing and Exploratory Data Analysis (EDS):

##### Outlier Detection:

A function is written to find the number of outliers in the dataset and the boxplot is created using the seaborn library for the given column the data outliers for the columns are any data points less than 25% and greater the 75% as a range of the outliers to be calculated.

Below is the function used to provide the number of outliers in the data.

Def LocatingOutliers(dataframe):

‘’’

Description: Finds Outliers as per the percentile provided

‘’’

I1=dataframe.quantile(0.25)

I2=dataframe.quantile(0.75)

Range\_Data=I2-I1

Outlier\_data = dataframe [((dataframe <( I1-1.5\* Range\_Data)) | (dataframe >( I2+1.5\* Range\_Data)))]

return Outlier\_data

##### 4.3.2 Scatter Plot:

A scatter plot is created to identify the shape of the minority class as well as the majority class and to visualize the dataset along with their multiple overlap and separations.

*Note: outputs for the given function are provided in sections 4.1.1 and 4.1.2.*

***The function to perform the following action is added below.***

Def PlotImbalanceDataset (data : pa.DataFrame, x1 : str, x2 : str, y : str, MinPlotLimit : int, MaxPlotLimit : int, Plottitle: str = ‘ ‘,save: bool = False,c = ‘YlOrRd’, figname = ‘Scatter.png’):

plt.figure (figsize = (6, 4))

plt.scatter (x = data [ data [y] == 0][x1], y = data [ data [y] == 0][x2], label = ‘Majority Class’, c = ‘yellow’, edgecolors = ‘black’)

plt.scatter (x = data [ data [y] == 1][x1], y=data [ data [y] == 1][x2], label = ‘Minority Class’, c = ‘gray’, edgecolors = ‘black’)

plt.title (Plottitle, fontsize = 12)

plt.xlabel (x1, fontsize = 12)

plt.ylabel (x2, fontsize = 12)

plt.xlim (MinPlotLimit, MaxPlotLimit)

plt.ylim (MinPlotLimit, MaxPlotLimit)

plt.legend()

if save:

plt.savefig (figname, dpi = 300, bbox\_inches = ‘tight’, pad\_inches = 1)

plt.show()

##### 4.3.3 Oversampling Scatter Plot:

A function to create a scatter plot is used to produce the plot of the newly generated points from the oversampling algorithms i.e. from SMOTE, ADASYN, and Proposed algorithm (SVM Oversampling). Furthermore, this output would give us insight and let us understand how the proposed algorithm is working when compared to SMOTE and ADASYN. While giving us the idea of where the newly generated points are created in a 2-Dimensional space.

*Note: Plots from the following functions are provided in sections 4.3 and 4.4.*

**The function to perform the following action is provided below:**

Def Plot\_Newly\_Genrated\_points (data: pa.DataFrame, x1: str, x2 : str, y : str, MinPlotLimit : int, MaxPlotLimit : int, title: str = ‘ ‘,save: bool = False,c = ‘YlOrRd’, figurename = ‘Oversamplingplot.png’):

plt.figure (figsize = ( 6 , 4 ))

plt.scatter (x = data [ data [y] == 0][x1], y = data [ data [y] == 0][x2], label = ‘Majority Class’, c = ‘yellow’ , edgecolors = ‘black’ )

plt.scatter ( x = data [ data [y] == 1][x1], y = data [ data [y] == 1][x2], label = ‘Minority Class’, c = ‘gray’, edgecolors = ‘black’)

plt.scatter(x = data [data [y] == 2][x1], y = data [ data [y] == 2][x2], label = ‘Newly Generated’, c = ‘red’, edgecolors = ‘black’ )

plt.title (title, fontsize = 10)

plt.xlabel (x1, fontsize = 10)

plt.ylabel (x2, fontsize = 10)

plt.xlim (MinPlotLimit, MaxPlotLimit)

plt.ylim(MinPlotLimit, MaxPlotLimit)

plt.legend()

if save:

plt.savefig (figurename, dpi = 400, bbox\_inches = ‘tight’, pad\_inches = 1)

plt.show()

## Oversampling:

After generating the imbalanced dataset, the next step is to use existing oversampling techniques such as SMOTE and ADASYN along with the proposed oversampling algorithm. The oversampling is performed on datasets with different shapes, orientations, and multiple degrees of overlapping. The dataset is split into two train and test sets. However, the oversampling is performed on the training dataset. The Python library “imblearn.over\_sampling” from this library SMOTE and ADASYN function are imported to perform oversampling.

##### 4.4.1 SMOTE Oversampling:

The Python Library imbalance learn is used to import the function SMOTE. The SMOTE function is fitted with two input parameters Training\_X and Training\_y, the function then creates synthetic data points for the minority class, The number of samples for each minority and majority class is printed before fitting and after fitting SMOTE to validate the data. The data from the oversampling is then plotted onto a scatter plot that provides us with insight into how the newly generated data is created for the SMOTE oversampling. (N. Chawla., 2002).

Examples of the scatter plot with different orientations, Shapes and Overlaps are provided below.

*Note: Python code for performing SMOTE oversampling is provided below:*

SMT = SMOTE()

print(‘Oversampled dataset shape %s’ % Counter(Training\_y))

X\_SMOTE\_oversample, y\_SMOTE\_oversample = SMT.fit\_resample(Training\_X, Training\_y)

print(‘Oversampled dataset shape %s’ % Counter(y\_SMOTE\_oversample))

1. Circular shape

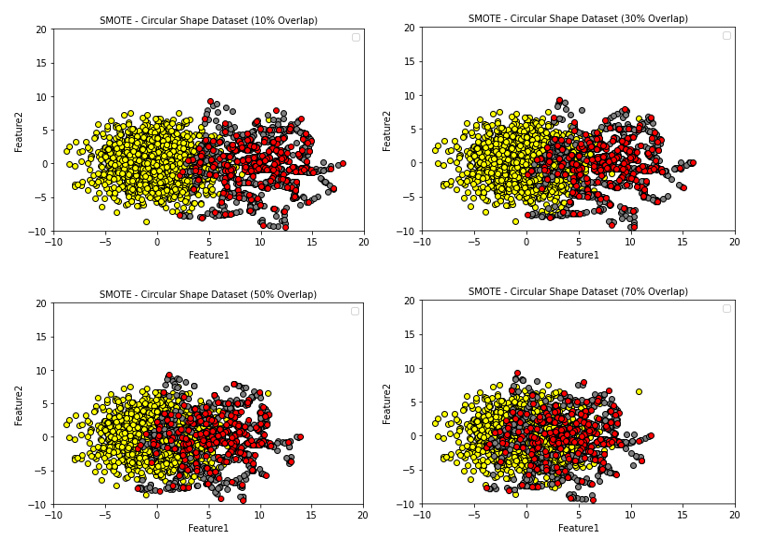


Figure SMOTE Circular Scatter Plot

1. Thick Flat Line Shaped

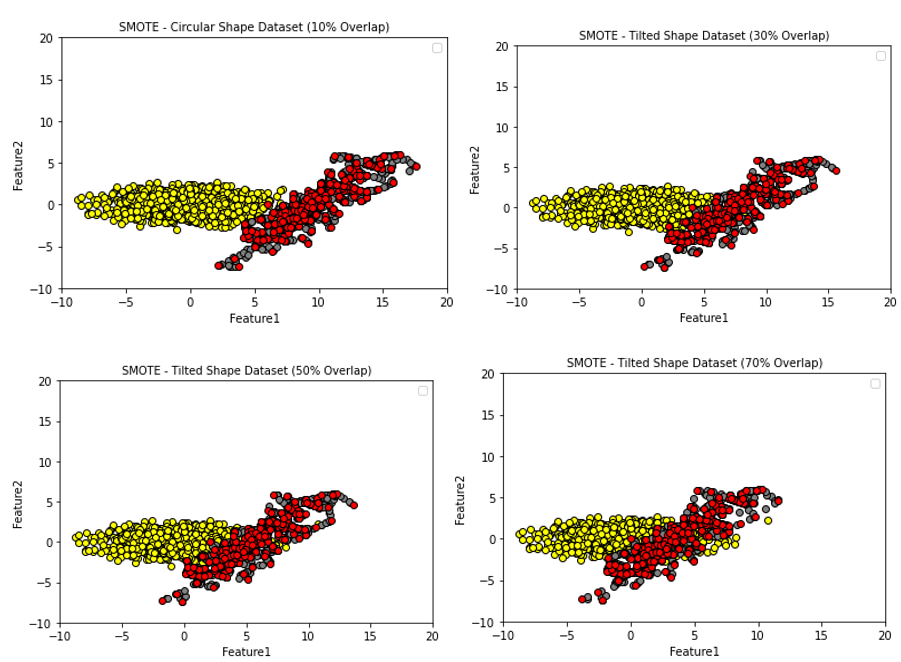


Figure Thick Flat Line Scatter Plot

##### 4.4.2 ADASYN Oversampling:

The Python Library imbalance learn is used to import the function ADASYN. The ADASYN function is fitted with two input parameters Training\_X and Training\_y, the function then creates synthetic data points for the minority class, The number of samples for each minority and majority class is printed before fitting and after fitting ADASYN to validate the data. The data from the oversampling is then plotted onto a scatter plot that provides us with insight into how the newly generated data is created for the ADASYN oversampling. (H. He, 2008)

*Note: Python code for performing ADASYN oversampling is provided below:*

ADA = ADASYN()

print('Resampled dataset shape %s' % Counter(Training\_y))

X\_ADA\_oversampling, y\_ADA\_oversampling = ADA.fit\_resample(Training\_X, Training\_y)

print('Resampled dataset shape %s' % Counter(y\_ADA\_oversampling))

Examples of the scatter plot with different orientations, Shapes and Overlaps are provided below.

1. Circular

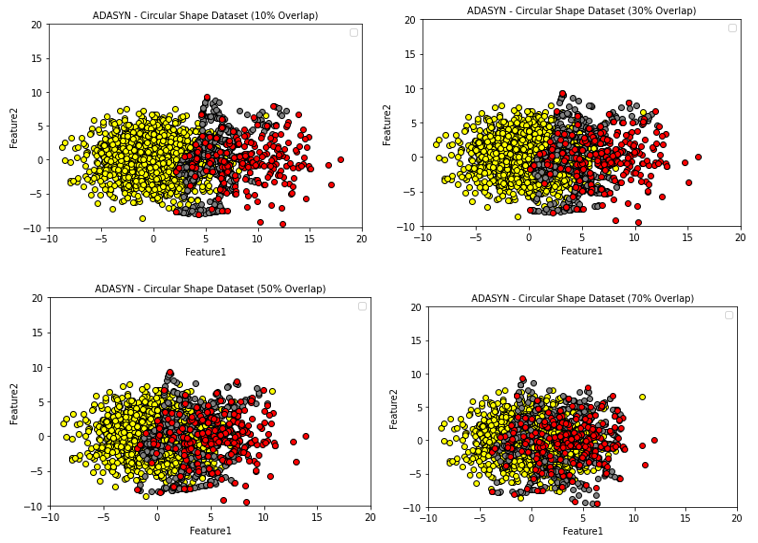


Figure ADASYN Circular Oversampling

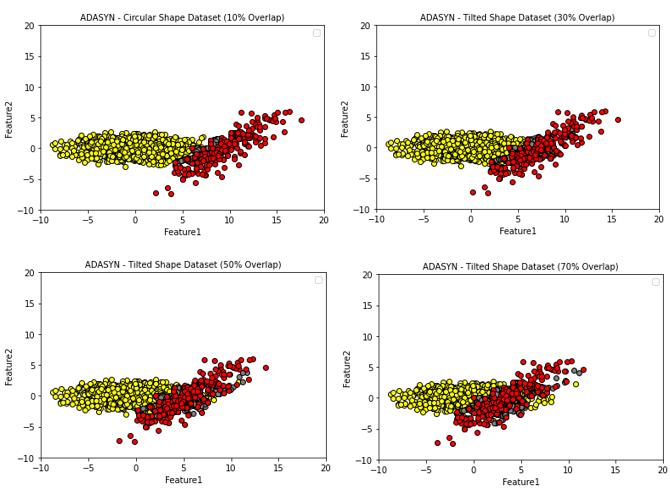
1. Thick Flat Line Oversampling 

Figure ADASYN Thick Flat Line Oversampling

##### 4.4.3 SVM Oversampling Algorithm (Proposed Algorithm):

Multiple steps are required to perform SVM based oversampling. SVC class is imported from python library scikit learn to get SVM package which is used to produce oversampling datapoints. Firstly, a grid search is performed on Training\_X and Training\_y giving us the best hyperparameters C and Gamma from the given range as per the dataset for tunning the proposed algorithm. Inside the parameter selection function, the grid search is tuned to be more inclined towards minority accuracy by keeping the recall\_score position to the minority class label ‘1’. As our focus is to increase the minority class accuracy.

**Python function for SVM hyperparameter selection:**

def svm\_hyperparam\_Grid(Training\_X, Training\_y):

Cost = [0.1,1,10,50,100]

Gamma = [1,50,100,500]

scoring = { 'AUCe': 'roc\_auc', 'Accuracy': 'accuracy', 'prec': 'precision', 'recall': 'recall', 'f1s': 'f1', 'spec' : make\_scorer( recall\_score, pos\_label = 1 ) }

param\_grid = { 'C' : Cost, 'gamma' : Gamma }

SVM\_param = GridSearchCV(SVC(kernel = 'rbf'), param\_grid = param\_grid, scoring = scoring, refit = 'recall', return\_train\_score = True)

SVM\_param.fit(Training\_X, Training\_y)

return SVM\_param.best\_params\_

The best parameter obtained from the above function cost and gamma will be used to train an SVM model along with the Radial Basis Function (RBF) Kernel.

**Python function for generating SVM classification:**

def generating\_SVM\_Cls(Cs, gammas, Training\_X, Training\_y, Testing\_X, Testing\_y):

class = SVC (kernel =' rbf', gamma = gammas, C = Cs)

class.fit (, Training\_X, Training\_y)

ypredict = class.predict(Testing\_X)

print("Training Accuracy = {}".format ( accuracy\_score(Testing\_y, ypredict)))

return class

**Python function to plot SVM Decicsion Boundary:**

def decision\_boundry\_SVM\_plot(X,y, model):

svm\_model = model.fit(X,y)

x\_mini, x\_maxi = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_mini, y\_maxi = X[:, 1].min() - 1, X[:, 1].max() + 1

height = (x\_maxi - x\_mini)/100

xxx, yyy = na.meshgrid( na.arange( x\_min, x\_max, height), na.arange (y\_min, y\_max, height))

plt.subplot(1, 1, 1)

zz = svm\_model.predict(na.c\_[ xxx.ravel(), yyy.ravel()])

zz = zz.reshape(xxx.shape)

plt.contourf(xxx, yyy, zz, cmap=plt.cm.Paired, alpha=0.8)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)

plt.xlabel('Length')

plt.ylabel('Width')

plt.xlim(xxx.min(), xxx.max())

plt.title('SVM Decision Boundary')

plt.show()

As the main SVM model and Hyperparameters are obtained, the next part is important to generate the minority samples for oversampling. To generate synthetic data points following actions are performed.

The data are grouped as per the class in our case Class ‘1’ and class ‘0’, the group from the minority class is stored in an object, and from that object mean of each column is obtained and stored in the variables (“mean\_x”, “mean\_y” ), the difference of minority and majority is calculated and stored as counts, A while loop is initiated and all these points are then fed into the trained SVM model. New samples are produced using a NumPy function “multivariate normal” which takes the mean values stored in the variable mean\_x and mean\_y as a center and “Mincov” as the covariance. These inputs are taken and synthetic data points are generated along the hyperplane. As per the results from the model, the algorithm decides whether to keep the point or drop the data point. The Algorithm drops and picks the points based on the decision boundary that is done by using the Decision\_Function() function from the SVC class.

The value obtained from the function is propositional to the space from sample Z to the dividing hyperplane. If the minority samples lies on the left area of the hyperplane then the (< “Less Than”) is used to produce oversampling data points on the left side of the decision boundary, Similarly if the minority class lies on the right area of the hyperplane then (> “Greater Than”) is used to generate the oversampling data points on the right area of the decision boundary.

For example, the data points on the decision function are less than “0” were used, If the provided decision value is 0.1 then the function will drop all the left side of the decision boundary. However, if the decision boundary value is -0.1 then the right side values will be dropped.

*Python Code to produce SVM based oversampling:*

mainclass = MainClass

parameters = mainclass. svm\_hyperparam\_selection (Training\_X,Training\_y)

print(parameters)

model\_svm = mainclass. generate\_SVM\_Cls (parameters['C'], parameters ['gamma'], Training\_X, Training\_y, Testing\_X,Testing\_y)

print(model\_svm)

plt.figure(2)

mainclass. decision\_boundry\_SVM\_plot (np.array(Training\_X), Training\_y, model\_svm)

group\_class = data.groupby(data.Results)

only\_minor = group\_class.get\_group(1)

mean\_x = statistics.mean(na.array(only\_minor.Feature1))

mean\_y = statistics.mean(na.array(only\_minor.Feature2))

print(mean\_x, mean\_y)

n\_unique, n\_counts = na.unique(y, return\_counts=True)

number\_train = dict (zip (n\_unique, n\_counts))

print('Before oversampling ', number\_train)

remaining\_train = number\_train [0.0] - number\_train [1.0]

print(remaining\_train)

oversampling\_samples = []

counting = 0

while counting!= remaining\_train:

synthetic\_points = na.c\_[

na.r\_[

na.random.multivariate\_normal([mean\_x, mean\_y], MinCov1, 1000)]

]

test\_predict = model\_svm.decision\_function(synthetic\_points) # to get the point location on hyperplane

test\_predict\_transfer = test\_predict.reshape((-1, 1))

synthetic\_points = na.c\_[ synthetic\_points, test\_predict\_transfer]

for j in synthetic\_points:

if counting < remaining\_train:

if j [2] > 0:

oversampling\_samples.append (j)

counting = counting + 1

j[2] = 2

else:

continue

else:

break

oversampling\_samples = na.array (oversampling\_samples)

oversampling\_T = na.array ((oversampling\_samples [:, 0], oversampling\_samples [:, 1])). T

oversampled\_X\_Training = na.concatenate ((na.array (Training\_X), oversampling\_T))

oversampled\_y\_Training = na.concatenate (( na.array (Training\_y), oversampling\_samples [:, 2]))

n\_unique, n\_counts = na.unique(oversampled\_y\_Training, return\_counts = True)

oversample\_counts = dict (zip (n\_unique, n\_counts))

print ('After oversampling ', oversample\_counts)

oversampled\_Training\_y = [1.0 if value == 2.0 else value for value in oversampled\_Training\_y]

1. Circular Dataset:

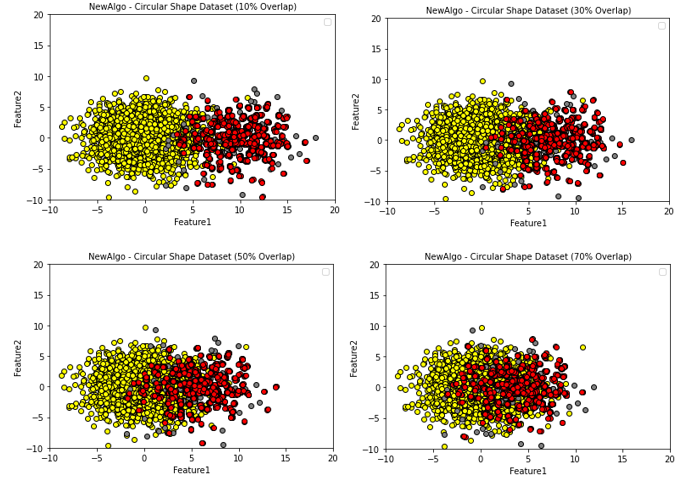


Figure Proposed Algorithm Circular Oversampling

1. Thick Flat Line Oversampling:

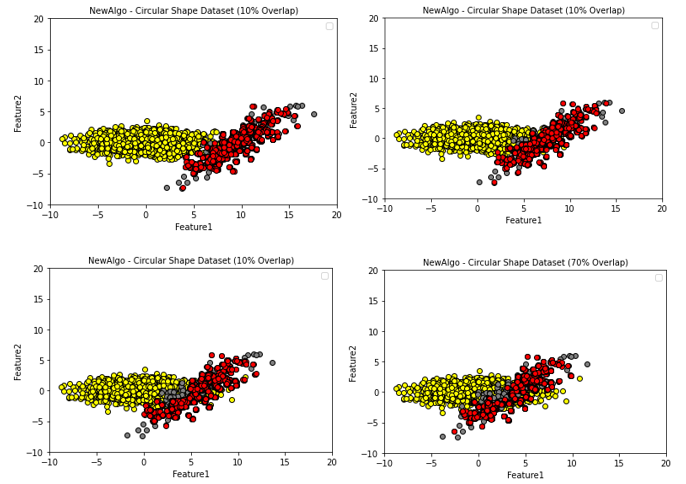


Figure Proposed Algorithm Thick Flat Line Oversampling

## 4.5 Model Training and Evaluation:

In this research, we have used existing oversampling techniques along with the proposed algorithm provided in the above unit to deal with the imbalance data problem. Moving forward it is very important to evaluate the oversampling techniques to understand how the proposed algorithm competes with the existing oversampling techniques. To evaluate the performance of oversampling algorithms we will train the classification models using the oversampled data from three different sources SMOTE, ADASYN, and SVM Oversampling while comparing the results of the classification models to evaluate the performance of each oversampling algorithm.

The python library ‘sklearn’ is used to import the classification model that is used to evaluate. There are three libraries imported from the library that is provided below:

1. Gaussian Naïve Bayes that is from sklearn.naive\_bayes
2. SVC that is from sklearn.svm (S and Dr. S V, 2014)
3. Multi-layer perceptron that is from sklearn.neural\_network

In this report, the modelling implementation is done as an individual function where all the functions for the classification model are stored under the class ‘evaluation\_models’. Furthermore, details for evaluations of the datasets are added below.

Upon completion of training the classification models using different oversampled datasets, performance evaluation is done using the techniques provided in the implementation section. The functions used for evaluating are :

* Confusion Matrix
* Accuracy
* Precision
* Recall
* F1-Score

From these five functions used to evaluate performance, the main focus of our report is to compare the confusion matrix, F1 score, and Recall scores of minority cases. This method enables us to understand how the classification models are working with different oversampling techniques.

**4.5.1 Naïve Bayes:**

def naive\_bayes(Training\_X,Training\_y,Testing\_X,Testing\_y,name):

ganb = GaussianNB()

ganb.fit(Training\_X, Training\_y)

y\_prediction\_gnab = ganb.predict(Testing\_X)

accuracTesting\_y = accuracy\_score(Testing\_y,y\_prediction\_ganb)

cm\_ganb = confusion\_matrix(Testing\_y, y\_prediction\_ganb)

report = classification\_report(Testing\_y, y\_prediction\_ganb)

print(name, report)

print(cm\_ganb)

return accuracTesting\_y, cm\_ganb

**4.5.2 Support Vector Machine (SVM):**

def svm\_model(Training\_X,Training\_y,Testing\_X,Testing\_y,name):

svc = SVC()

svc.fit(Training\_X, Training\_y)

y\_prediction\_svc = svc.predict(Testing\_X)

accuracTesting\_y = accuracy\_score(Testing\_y,y\_prediction\_svc)

cm\_svm = confusion\_matrix(Testing\_y, y\_prediction\_svc)

report = classification\_report(Testing\_y, y\_prediction\_svc)

print(name, report)

print(cm\_svm)

return accuracTesting\_y, cm\_svm

* + 1. **Multi-layer Perceptron (MLP):**

def MLP\_model(Training\_X,Training\_y,Testing\_X,Testing\_y,name):

mlp = MLPClassifier(solver = 'lbfgs', alpha = 1e-5, hidden\_layer\_sizes = (5, 2), random\_state = 1)

mlp.fit(Training\_X, Training\_y)

y\_prediction\_mlp = mlp.predict(Testing\_X)

accuracTesting\_y = accuracy\_score(Testing\_y,y\_prediction\_mlp)

cm\_mlp = confusion\_matrix(Testing\_y, y\_prediction\_mlp)

report = classification\_report(Testing\_y, y\_prediction\_mlp)

print(name, report)

print(cm\_mlp)

return accuracTesting\_y, cm\_mlp

# Experiments And Results:

In this report, there are multiple experimental designs executed and compared their results to identify which oversampling technique performs better for each experiment. The recall technique used in the oversampling algorithm for performance evaluation provides outcomes for both majority and minority classes respectively.

This experiment provides a detailed comparison between the existing oversampling techniques and proposed SVM oversampling techniques. furthermore, to increase the accuracy and effectiveness of the proposed algorithm the Gamma and Cost values are compared to get the best hyperparameter value for the prediction of the algorithm. The classification models used to train and test the dataset are “Naïve Bayes”, “SVM”, and “Multi-Layer Perceptron (MLP)”.

Multiple experiments are carried out in this research are:

1. Multiple degrees of overlap in minority and majority classes with different shapes
2. Proposed oversampling with multiple hyperparameter values of Cost and Gamma Function.
3. Proposed oversampling with multiple values of Decision boundaries.

The code for creating a synthetic dataset with different shapes and orientations and plots is available in the Implementation section. furthermore, three classification models were used Naïve Bayes, Support Vector Machine, and Multi-Layer Perceptron.

## 5.1 Phase I – Multiple degrees of overlapping:

This experimental approach aims to compare the results of oversampling algorithms depending on the overlapping between minority and majority classes. The different degrees of overlap used in this experiment are 10, 30, 50, and 70, A uniform interval of thirty is maintained for the difference in the degree of overlap between the experiments which helps us in identifying the results depending on the degree of overlap. The first dataset used for the experiment is circular in shape for both minority and majority classes. There are three oversampling techniques used on the experimental data SMOTE, ADASYN, and the proposed algorithm (SVM oversampling). Furthermore, for this report, three classification models were used for training and testing the dataset Naïve Bayes, Support vector machines (SVM), and Multilayer Perceptron.

The experiments were performed in jupyter notebook (A Python development environment). The steps of conducting the experiments are provided below:

* To begin with, an artificial dataset is generated using the NumPy library where the majority samples are 2250 and the minority samples are 250 making the ratio of majority to minority 10:1 the first experiment is performed on the dataset with an overlap of 10% followed by 30%,50%, and 70%. Specifications on the shape of the dataset are provided in the results figures.
* Generate a scatter plot of the synthetic dataset to visualize the data points.
* Generate a distribution plot to identify if the data is in normal/gaussian distribution.
* Split the Dataset into two training and testing, the split is performed at a ratio of 4:1 where the 20% of data is for testing, and 80% of data is for training the model.
* The oversampling is applied to the training data to add more artificial data points on the minority class and stored as a separate data frame the oversampling algorithms applied are SMOTE, ADASYN oversampling
* A scatter plot is generated the show the majority points, old minority points, and newly generated oversampled minority points.
* The proposed oversampling technique is applied to the training data where the costa and gamma hyperparameters are selected from the grid search, for this experiment the decision boundary is kept at ‘0’ to select all the points greater than 0.
* A Decision boundary plot is created showing us the hyperplane which divides the classes. Higher the gamma value higher the curvature of the decision boundary.
* A scatter plot is generated that shows us the majority, existing minority points, and newly generated minority points by the proposed algorithm.
* The classification models are applied i.e., Naïve Bayes, SVM, and Multi-layer perceptron where the results of each oversampling algorithm are compared with one another. To evaluate the performance of the algorithm confusion matrix, Recall, F1, and Accuracy are used.

The table below provides the outputs from their respective oversampling algorithms depending on their overlapping percentage. The primary focus of the results is on the confusion matrix followed by Recall and F1 Score.

##### 5.1.1 Circular Dataset:

The experiment is performed on a circular dataset with different degrees of overlap. However, the gamma and cost remained constant at 50 and 1 respectively. Furthermore, the decision boundary of this is maintained at I > 0 for new point selection. The scatter plot of this experiment can be seen in section 4.3.3 and figure 14

###### 5.1.1.1 Naïve Bayes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| SMOTE | **Acc = 0.95**  0 1  0 [ 384 19 ]  1 [ 2 45 ] | **Acc = 0.92**    0 1  0 [ 371 32 ]  1 [ 4 43 ] | **Acc = 0.84**  0 1  0 [ 343 60 ]  1 [ 10 37 ] | **Acc = 0.78**  0 1  0 [ 322 81 ]  1 [ 19 28 ] |
| ADASYN | **Acc = 0.88**  0 1  0 [ 350 53 ]  1 [ 1 46 ] | **Acc = 0.83**  0 1  0 [ 327 76 ]  1 [ 1 46 ] | **Acc = 0.78**  0 1  0 [ 307 96]  1 [ 5 42 ] | **Acc = 0.71**  0 1  0 [ 283 120 ]  1 [ 11 36 ] |
| Proposed Algo (SVM Based) | **Acc = 0.96**  0 1  0 [ 386 17 ]  1 [ 2 45 ] | **Acc = 0.92**  0 1  0 [ 374 29 ]  1 [ 6 41 ] | **Acc = 0.85**  0 1  0 [ 344 59 ]  1 [ 10 37 ] | **Acc = 0.76**  0 1  0 [ 306 97 ]  1 [ 13 34 ] |

Table Confusion Matrix Naïve Bayes (Circular)

Recall score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | **0.96** | **0.92** | 0.82 | 0.70 |
| **ADASYN** | 0.92 | 0.90 | **0.83** | 0.73 |
| **SVM Based** | **0.96** | 0.90 | 0.82 | **0.74** |

Table Recall Score Naïve Bayes (Circular)

F1 Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.89 | **0.83** | **0.71** | **0.61** |
| **ADASYN** | 0.78 | 0.72 | 0.66 | 0.58 |
| **SVM Based** | **0.90** | **0.83** | **0.71** | **0.61** |

Table F1 Score Naïve Bayes (Circular)

###### 5.1.1.2 Support Vector Machine (SVM):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| SMOTE | **Acc = 0.94**  0 1  0 [ 379 24 ]  1 [ 2 45 ] | **Acc = 0.91**    0 1  0 [ 364 39 ]  1 [ 3 44 ] | **Acc = 0.86**  0 1  0 [ 351 52 ]  1 [ 10 37 ] | **Acc = 0.79**  0 1  0 [ 329 74 ]  1 [ 20 27 ] |
| ADASYN | **Acc = 0.88**  0 1  0 [ 350 53 ]  1 [ 1 46 ] | **Acc = 0.80**  0 1  0 [ 313 90 ]  1 [ 2 45 ] | **Acc = 0.78**  0 1  0 [ 307 96]  1 [ 5 42 ] | **Acc = 0.73**  0 1  0 [ 279 106 ]  1 [ 14 33 ] |
| Proposed Algo (SVM Based) | **Acc = 0.96**  0 1  0 [ 385 18 ]  1 [ 2 45 ] | **Acc = 0.92**  0 1  0 [ 371 32 ]  1 [ 5 42 ] | **Acc = 0.86**  0 1  0 [ 349 54 ]  1 [ 10 37 ] | **Acc = 0.77**  0 1  0 [ 312 19 ]  1 [ 14 33 ] |

Table Confusion Matrix SVM (Circular)

Recall Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.95 | **0.92** | **0.83** | 0.70 |
| **ADASYN** | 0.92 | 0.87 | 0.82 | 0.72 |
| **SVM Based** | **0.96** | 0.91 | **0.83** | **0.74** |

Table Recall Scores SVM (Circular)

F1 Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.87 | 0.81 | **0.73** | **0.62** |
| **ADASYN** | 0.78 | 0.68 | 0.66 | 0.59 |
| **SVM Based** | **0.90** | **0.82** | **0.73** | **0.62** |

Table F1 Scores SVM (Circular)

###### 5.1.1.3 Multi-Layer Perceptron:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| SMOTE | **Acc =0.94**  0 1  0 [ 378 25 ]  1 [ 2 45 ] | **Acc = 0.90**    0 1  0 [ 361 42 ]  1 [ 3 44 ] | **Acc = 0.86**  **0 1**  **0 [ 351 52 ]**  **1 [ 9 38 ]** | **Acc = 0.77**  0 1  0 [ 319 84 ]  1 [ 20 27 ] |
| ADASYN | **Acc = 0.89**  0 1  0 [ 354 49 ]  1 [ 1 46 ] | **Acc = 0.80**  0 1  0 [ 317 86 ]  1 [ 2 45 ] | **Acc = 0.76**  0 1  0 [ 300 103]  1 [ 6 41 ] | **Acc = 0.72**  0 1  0 [ 296 107 ]  1 [ 20 27 ] |
| Proposed Algo (SVM Based) | **Acc = 0.96**  0 1  0 [ 386 17 ]  1 [ 2 45 ] | **Acc = 0.92**  0 1  0 [ 375 28 ]  1 [ 6 41 ] | **Acc = 0.85**  0 1  0 [ 347 56 ]  1 [ 11 36 ] | **Acc = 0.76**  **0 1**  **0 [ 313 90 ]**  **1 [ 16 31 ]** |

Table Confusion Matrix MLP (Circular)

Recall Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.95 | **0.92** | **0.84** | 0.68 |
| **ADASYN** | 0.93 | 0.87 | 0.81 | 0.65 |
| **SVM Based** | **0.96** | 0.90 | 0.81 | **0.72** |

Table Recall Scores MLP (Circular)

F1 Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.87 | 0.80 | **0.74** | 0.60 |
| **ADASYN** | 0.79 | 0.69 | 0.64 | 0.56 |
| **SVM Based** | **0.90** | **0.83** | 0.71 | **0.61** |

Table F1 Scores MLP (Circular)

##### 5.1.2 Thick Flat Line Shaped Dataset:

The experiment is performed on a majority flat and minority 45-degree incline dataset with different degrees of overlap (10,30,50, and 70). However, that gamma and cost remained constant at 50 and 1 respectively. Furthermore, the decision boundary of this is maintained at I > 0 for new point selection. The scatter plot of this experiment can be seen in section 4.3.3 and figure 15

###### 5.1.2.1 Naïve Bayes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| SMOTE | Acc = 0.97  0 1  0 [ 388 15 ]  1 [ 0 47 ] | Acc = 0.94  0 1  0 [ 381 22 ]  1 [ 3 44 ] | Acc = 0.90  0 1  0 [ 367 36 ]  1 [ 7 40 ] | Acc = 0.85  0 1  0 [ 350 53 ]  1 [ 13 34 ] |
| ADASYN | Acc = 0.94  0 1  0 [ 377 26 ]  1 [ 0 47 ] | Acc = 0.92  0 1  0 [ 366 37 ]  1 [ 0 47 ] | Acc = 0.84  0 1  0 [ 333 70 ]  1 [ 3 44 ] | Acc = 0.85  0 1  0 [ 293 110 ]  1 [ 6 41 ] |
| Proposed Algo (SVM Based) | Acc = 0.97  0 1  0 [ 389 14 ]  1 [ 0 47 ] | Acc = 0.94  0 1  0 [ 379 24 ]  1 [ 2 45 ] | Acc = 0.89  0 1  0 [ 359 44 ]  1 [ 6 41 ] | Acc = 0.82  0 1  0 [ 333 70 ]  1 [ 10 37] |

Table Confusion Matrix Naïve Bayes (Thick Flat Line)

Recall Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | **0.98** | 0.94 | **0.88** | 0.80 |
| **ADASYN** | 0.87 | **0.95** | **0.88** | 0.80 |
| **SVM Based** | **0.98** | **0.95** | **0.88** | **0.81** |

Table Recall Scores Naïve Bayes (Thick Flat Line)

F1 Score:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.92 | **0.87** | **0.80** | **0.71** |
| **ADASYN** | 0.88 | 0.83 | 0.72 | 0.62 |
| **SVM Based** | **0.93** | **0.87** | 0.78 | 0.69 |

Table F1 Scores Naïve Bayes (Thick Flat Line)

###### 5.1.2.2 Support Vector Machine (SVM):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| SMOTE | Acc = 0.98  0 1  0 [ 397 6 ]  1 [ 1 46] | Acc = 0.96  0 1  0 [ 387 16 ]  1 [ 3 44 ] | Acc = 0.90  0 1  0 [ 363 40 ]  1 [ 5 42 ] | Acc = 0.82  0 1  0 [ 327 76 ]  1 [ 7 40 ] |
| ADASYN | Acc = 0.98  0 1  0 [ 391 11 ]  1 [ 0 47 ] | Acc = 0.94  0 1  0 [ 376 27 ]  1 [ 1 46 ] | Acc = 0.86    0 1  0 [ 345 58 ]  1 [ 3 44 ] | Acc = 0.75  0 1  0 [ 292 111 ]  1 [ 5 42 ] |
| Proposed Algo (SVM Based) | Acc = 0.98  0 1  0 [ 394 9 ]  1 [ 1 46 ] | Acc = 0.96  0 1  0 [ 388 15 ]  1 [ 3 44 ] | Acc = 0.92  0 1  0 [ 374 29 ]  1 [ 7 40 ] | Acc = 0.86  0 1  0 [ 351 52 ]  1 [ 10 37 ] |

Table Confusion Matrix SVM (Thick Flat Line)

Recall Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.98 | 0.95 | **0.90** | **0.83** |
| **ADASYN** | **0.99** | **0.96** | **0.90** | 0.81 |
| **SVM Based** | 0.98 | 0.95 | 0.89 | **0.83** |

Table Recall Scores SVM (Thick Flat Line)

F1 Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | **0.96** | **0.90** | 0.80 | 0.69 |
| **ADASYN** | 0.94 | 0.87 | 0.75 | 0.63 |
| **SVM Based** | 0.94 | **0.90** | **0.82** | **0.73** |

Table F1 Scores SVM (Thick Flat Line)

###### 5.1.2.3 Multi-layer perceptron:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| SMOTE | Acc = 0.98  0 1  0 [ 397 6 ]  1 [ 3 44] | Acc = 0.96  0 1  0 [ 390 13 ]  1 [ 4 43 ] | Acc = 0.90  0 1  0 [ 364 39 ]  1 [ 5 42 ] | Acc = 0.84  0 1  0 [ 340 63 ]  1 [ 10 37 ] |
| ADASYN | Acc = 0.95  0 1  0 [ 392 11 ]  1 [ 0 47 ] | Acc = 0.95  0 1  0 [ 383 20 ]  1 [ 1 46 ] | Acc = 0.88  0 1  0 [ 351 51 ]  1 [ 4 43 ] | Acc = 0.77  0 1  0 [ 306 97 ]  1 [ 5 42 ] |
| Proposed Algo (SVM Based) | Acc = 0.98  0 1  0 [ 396 7 ]  1 [ 1 46 ] | Acc = 0.96  0 1  0 [ 389 14 ]  1 [ 4 43 ] | Acc = 0.92  0 1  0 [ 376 27 ]  1 [ 7 40 ] | Acc = 0.85  0 1  0 [ 346 57 ]  1 [ 10 37 ] |

Table Confusion Matrix MLP (Thick Flat Line)

Recall Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | 0.96 | 0.94 | **0.90** | 0.82 |
| **ADASYN** | **0.99** | **0.96** | 0.89 | **0.83** |
| **SVM Based** | 0.98 | 0.94 | 0.89 | 0.82 |

Table Recall Scores MLP (Thick Flat Line)

F1Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Overlapping Techniques** | **Degrees of Overlap Percentage** | | | |
| **10%** | **30%** | **50%** | **70%** |
| **SMOTE** | **0.95** | **0.91** | 0.80 | 0.70 |
| **ADASYN** | 0.94 | 0.89 | 0.77 | 0.65 |
| **SVM Based** | **0.95** | 0.90 | **0.83** | **0.72** |

Table F1 Scores MLP (Thick Flat Line)

In a nutshell, the proposed algorithm (SVM oversampling) works well with increasing complexity due to an increase in overlapping for all the classification models i.e. Naïve Bayes, Support Vector Machine, and Multi-layer perceptron. Moreover, ADASYN provided the best results for Minority prediction while compromising the results on the majority side. However, in most cases, the proposed algorithm sits right in the middle of SMOTE and ADASYN by providing a balanced performance between the minority and majority prediction. Furthermore, the proposed model provides better Recall and F1 scores in several cases when compared to SMOTE and ADASYN.

## 5.2 Phase II – Modifying Gamma:

The goal of this experiment is to compare the performance of oversampling techniques on different imbalance data points with various overlapping and the effects of gamma value on the accuracy and prediction of the model for this experiment.

There are two shapes used to test and evaluate the results of different oversampling algorithms with the proposed algorithm. Firstly, a circular shape dataset is used. Secondly, a 45-degree minority and a horizontal majority are used. The artificial dataset contains “2250” whereas the minority cases generated are “250”, furthermore the dataset is split into train and test with a ratio of 4:1 while the majority to minority ratio is “10:1 ”, Moreover the overlap degree used in this experiment is 50 along with the value of gamma used in the experiments are 1,50,100, and 500 and the value of the cost remains constant as 1.

The value of Gamma and cost are directly passed into the grid search function manually.

##### 5.2.1 Circular Dataset:

###### 5.2.1.1 Decision Boundaries:

**Overlapping = 50**

|  |  |
| --- | --- |
| Gamma = 1, C = 1 | Gamma = 50 C = 1 |
| Gamma = 100 , C= 1 | Gamma = 500, c = 1 |

Figure Decision Boundary (Circular)

###### 5.2.1.2 Oversampling Scatter Plot:

Below are the scatter plot of the SVM based on different Gamma value for the same datasets.

**Overlapping 50:**

|  |  |
| --- | --- |
| Gamma = 1 , C =1 | Gamma = 50, C =1 |
|  |  |
| Gamma = 100 , C = 1 | Gamma = 500 , C = 1 |
|  |  |

Figure Modified Gamma Scatter Plot (Circular)

###### 5.1.2.3 Confusion matrix and Accuracies:

Overlapping = 50

|  |  |  |
| --- | --- | --- |
| Naïve Bayes | SVM | MLP |
| SMOTE | SMOTE | SMOTE |
| ADASYN | ADASYN | ADASYN |
| Gamma = 1, C = 1 | Gamma = 1 C =1 | Gamma = 1 C = 1 |
| Gamma = 50 , C = 1 | Gamma = 50 C = 1 | Gamma = 50, C = 1 |
| Gamma = 100 , C = 1 | Gamma = 100, C = 1 | Gamma = 100, C=1 |
| Gamma = 500 , C= 1 | Gamma = 500 , C = 1 | Gamma = 500, C = 1 |

Table Modified Gamma Scatter Plot

##### 5.2.2 Thick Flat Line:

###### 5.2.2.1 Decision Boundary:

|  |  |
| --- | --- |
| Gamma = 1, C = 1 | Gamma = 50, C = 1 |
| Gamma = 100, C=1 | Gamma = 500, C = 1 |
|  |  |

Table Decision Boundary (Thick Flat Line)

###### 5.2.2.2 Oversampling Scatter Plot:

Below are the scatter plot of the SVM based on different Gamma value for the same datasets.

Graphical user interface, chart, scatter chart

Description automatically generated

Figure Modified Gamma Scatter Plot (Thick Flat Line)

###### 5.2.2.3 Confusion Matrix:

|  |  |  |
| --- | --- | --- |
| Naïve Bayes | Support Vector Machines | Multi-layer Perceptron |
| SMOTE | SMOTE | SMOTE |
| ADASYN | ADASYN | ADASYN |
| Gamma= 1 | Gamma =1 | Gamma = 1 |
| Gamma = 50 | Gamma = 50 | Gamma = 50 |
| Gamma = 100 | Gamma = 100 | Gamma = 100 |
|  |  |  |

##### Brief Conclusion:

In conclusion, it is observed for the datasets that if the value of gamma is low the data produced by the proposed algorithm is widespread which may lead to confusion in the classification model. From the results of the confusion matrix, it can be concluded that low gamma leads to low prediction for the minority class. On the other hand, the scatter plots with higher gamma give us more curvy decision boundary data points while creating samples near the minority space. However, if we keep the value of gamma too high the sample space to create new data points gets too restricted which may have a positive impact on the minority class but overall the accuracy is reduced, For the experiment the analyzed sweet spot of gamma is at 50 or 100 that provides a balance between most accuracy for minority class with compromising on the majority accuracy.

## Experiment Phase III – Modifying Decision Boundary:

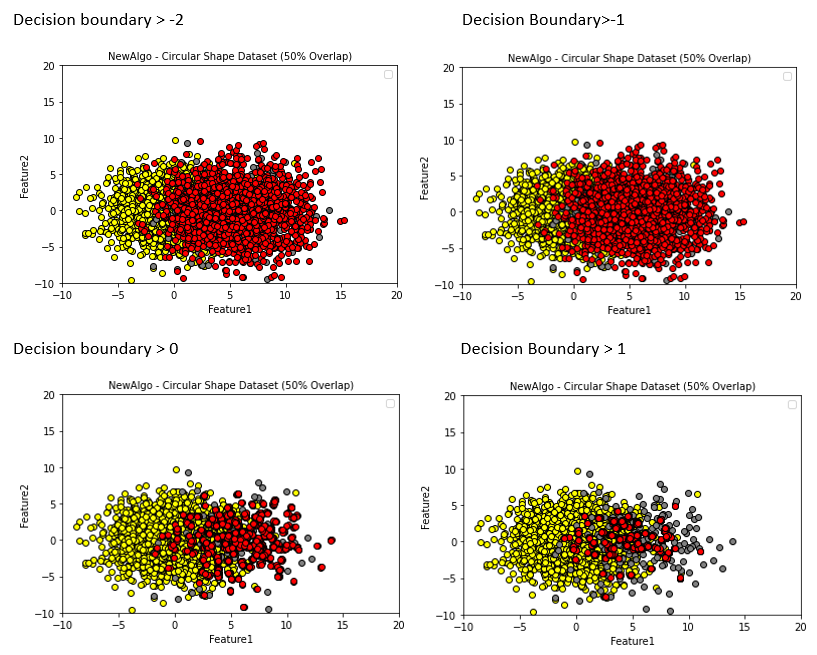
In this experiment we aim to modify the decision boundary value on creating oversampled data on the proposed algorithm and compare their performance, the decision boundary value used in this experiment are -2, -1, 0, 1, and 2, and train the model using this data. The classification models used in this experiment are Naïve Bayes (NB), Support Vector Machines (SVM), and Multi-layer perceptron(MLP).

There are two shapes used to test and evaluate the results of different oversampling algorithms with the proposed algorithm. Firstly, a circular shape dataset is used. Secondly, a 45-degree minority and a horizontal majority are used. The artificial dataset contains “2250” whereas the minority cases generated are “250”, furthermore the dataset is split into train and test with a ratio of 4:1 while the majority to minority ratio is “10:1 ”, Moreover the overlap degree used in this experiment is 50 along with the value of gamma used in the experiments are 50 while the value of is 1.

The value of Gamma and cost are directly passed into the grid search function manually.

##### 5.2.1 Circular Dataset:

###### 5.2.1.1 Scatter Plot:



Decision Boundary > 2

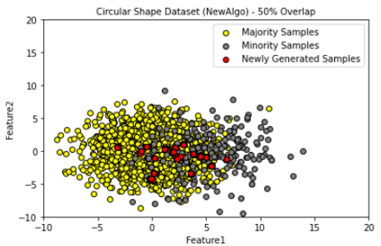


Figure Decision Boundary (Circular)

###### 5.2.1.2 Confusion Matrix:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification Models** |  | **Decision Boundary** | | | |
| **-2** | **-1** | **0** | **1** | **2** |
| **Naïve Bayes** | Acc = 0.84  0 1  0 [ 337 66 ]  1 [ 8 39 ] | Acc = 0.84  0 1  0 [ 339 64 ]  1 [ 9 38 ] | Acc= 0.85  0 1  0 [ 344 59 ]  1 [ 10 37 ] | Acc = 0.80  0 1  0 [ 321 82 ]  1 [ 6 41 ] | Acc = 0.76  0 1  0 [ 307 96 ]  1 [ 11 36 ] |
| **SVM** | Acc = 0.84  0 1  0 [ 341 62 ]  1 [ 10 37 ] | Acc = 0.84  0 1  0 [ 344 59 ]  1 [ 11 36 ] | Acc = 0.86  0 1  0 [ 349 54 ]  1 [ 10 37 ] | Acc = 0.80  0 1  0 [ 319 84 ]  1 [ 6 41 ] | Acc = 0.79  0 1  0 [ 314 89 ]  1 [ 7 40 ] |
| **MLP** | Acc = 0.84  0 1  0 [ 338 65 ]  1 [ 9 38 ] | Acc = 0.84  0 1  0 [ 340 64 ]  1 [ 8 39 ] | Acc = 0.85  0 1  0 [ 347 56 ]  1 [ 11 36 ] | Acc = 0.82  0 1  0 [ 328 75 ]  1 [ 8 39 ] | Acc = 0.81  0 1  0 [ 323 80 ]  1 [ 7 40 ] |

Table Confusion Matrix Decision Boundary

###### 5.2.1.3 Recall Scores:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification Models** |  | **Decision Boundary** | | | |
| **-2** | **-1** | **0** | **1** | **2** |
| **Naïve Bayes** | **0.83** | **0.82** | **0.82** | **0.83** | **0.76** |
| **SVM** | **0.82** | **0.81** | **0.83** | **0.83** | **0.82** |
| **MLP** | **0.82** | **0.84** | **0.81** | **0.82** | **0.83** |

Table Recall Scores Decision Boundary

###### 5.2.1.4 F1 Scores:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Classification Models** |  | **Decision Boundary** | | | |
| **-2** | **-1** | **0** | **1** | **2** |
| **Naïve Bayes** | **0.71** | **0.71** | **0.71** | **0.68** | **0.63** |
| **SVM** | **0.71** | **0.71** | **0.73** | **0.68** | **0.66** |
| **MLP** | **0.70** | **0.71** | **0.71** | **0.69** | **0.68** |

Table F1 Scores Decision Boundary (Circular)

##### 5.2.2 Thick Flat Line:

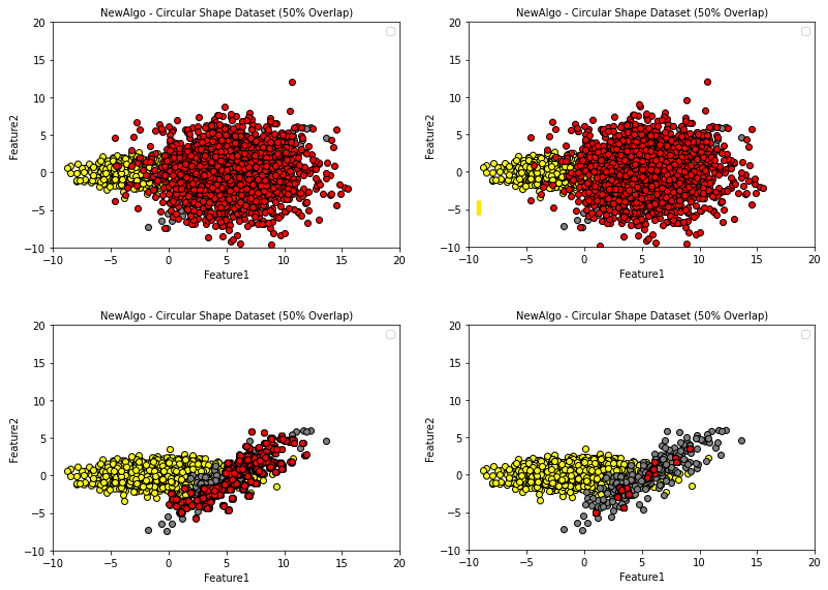


Figure Decision boundary Flat Line

###### 5.2.1.1 Confusion Matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Models** | **Decision Boundary** | | | |
| **-2** | **-1** | **0** | **1** |
| **Naïve Bayes** | Acc = 0.82  0 1  0 [ 333 70 ]  1 [ 10 37 ] | Acc= 0.91  0 1  0 [ 368 35 ]  1 [ 7 40 ] | Acc = 0.89  0 1  0 [ 359 44 ]  1 [ 6 41 ] | **Acc = 0.86**  0 1  0 [ 359 44 ]  1 [ 6 41 ] |
| **SVM** | Acc = 0.86  0 1  0 [ 351 52 ]  1 [ 10 37 ] | Acc = 0.91  0 1  0 [ 367 36 ]  1 [ 6 41 ] | Acc = 0.92  0 1  0 [ 374 29 ]  1 [ 7 40 ] | **Acc = 0.92**  0 1  0 [ 374 29 ]  1 [ 7 40 ] |
| **MLP** | Acc = 0.85  0 1  0 [ 346 57 ]  1 [ 10 37 ] | Acc = 0.85  0 1  0 [ 363 40 ]  1 [ 7 40 ] | Acc = 0.92  0 1  0 [ 376 27 ]  1 [ 7 40 ] | **Acc = 0.92**  0 1  0 [ 374 29 ]  1 [ 8 39 ] |

Table Confusion Matrix Decision Boundary

###### 5.2.2.2 Recall Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Models** | **Decision Boundary** | | | |
| **-2** | **-1** | **0** | **1** |
| **Naïve Bayes** | **0.81** | **0.88** | **0.88** | **0.88** |
| **SVM** | **0.83** | **0.89** | **0.89** | **0.89** |
| **MLP** | **0.82** | **0.88** | **0.89** | **0.88** |

Table Recall Score Decision Boundary (Thick Flat Line)

###### 5.2.2.3 F1 Scores:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classification Models** | **Decision Boundary** | | | |
| **-2** | **-1** | **0** | **1** |
| **Naïve Bayes** | **0.69** | **0.80** | **0.78** | **0.75** |
| **SVM** | **0.73** | **0.80** | **0.82** | **0.82** |
| **MLP** | **0.72** | **0.78** | **0.83** | **0.82** |

Table Recall Score Decision Boundary (Thick Flat Line)

***NOTE: for the thick flat line dataset the decision of positive 2 has higher time complexity. so for this dataset, the decision boundary of 2 is omitted.***

##### Brief Conclusion:

In closing, it can be observed from the scatter plot of the proposed oversampling that a lower decision boundary value leads to a more spread for the generation of artificial oversampling data points due to which the classification model gets confused between the majority and minority while the prediction giving us lower accuracy on majority class as well as on minority class. On the other hand, if we increase the value of the decision boundary to 2 the area for generating oversampling data points gets too restricted which leads to the stacking of newly generated data points on one another. This gives a false sense of minority class to the classification model while presenting a drastic reduction in prediction for both minority and majority classes. For this experiment, it is analyzed that the best decision boundary for both datasets is at 0 with better F1 and Recall scores followed by -1.

# 6 Conclusion and Future Work:

To sum it all up from the experiments performed it can be perceived that the proposed algorithm is very flexible and works well with the oversampling of datasets. It provides more control on oversampling when compared to existing oversampling algorithms.

**The conclusion from Experiment 1:**

For all classification models, namely Nave Bayes, Support Vector Machine, and Multi-layer Perceptron, the proposed algorithm (SVM oversampling) works well for all overlaps, especially with increasing complexity caused by an increase in overlapping.

Furthermore, ADASYN produced the best Minority prediction results while compromising the majority prediction results.

However, in the majority of cases (with greater overlap), the proposed algorithm produces better results. when compared to SMOTE and ADASYN, the proposed model provides higher Recall and F1 scores in several cases.

The proposed algorithm provides a balanced performance between minority and majority prediction, the proposed oversampling algorithm falls right in the middle of SMOTE and ADASYN.

**The conclusion from Experiment 2:**

For the datasets, it is observed that if the gamma value is low the decision boundary contains straighter lines, and the data produced by the proposed algorithm is widespread, which may cause distress in the classification model.

The confusion matrix results indicate that low gamma leads to poor prediction for the minority class.

Higher gamma scatter plots, on the other hand, produce more curvy decision boundary in feature space while creating samples near the minority space.

However, if we keep the gamma value too high, the sample space for creating new data points becomes too limited, which may benefit the minority class but reduces overall accuracy. For the experiment, the analyzed sweet spot of gamma is found at 50 or 100 which provides a balance between most accuracy for the minority class without compromising on the majority accuracy.

**The conclusion from Experiment 3:**

The scatter plot of the proposed oversampling shows that a lower decision boundary value leads to a greater spread for the generation of artificial oversampling data points, causing the classification model to become confused between the majority and minority samples, resulting in lower prediction accuracy for both the majority and minority classes.

If we increase the value of the decision boundary to 2, the area for generating oversampling data points becomes too constrained, resulting in the stacking of newly generated data points on top of one another.

This gives the classification model a false sense of minority class while presenting a drastic reduction in prediction for both minority and majority classes. The optimal decision boundary for both the datasets is 0 providing better results for confusion matrix, Recall, and F1 followed by -1. by this, we can understand the decision boundary hugely affect the classification model performance

**Future Work:**

To create Python code that not only grid searches the hyperparameter (Gamma and C) but also finds the best decision boundary for a given dataset and provide the best results from the proposed algorithm.

# Referencing:

Mathew, J. (2018) “Classification of imbalanced data by Oversampling in Kernel space of Support Vector Machines”, *IEEE transactions on neural networks and learning systems, VOL. 29.*

Available at:

Josey Mathew, Ming Luo, Chee Khiang Pang, and Hian Leng Chan*.* Kernel-based SMOTE for SVM Classification of imbalanced datasets. *IECON2015 – Yokohama,* Published on 9 November 2015.

Jia-Bao Wang, Chun-An Zou, and Guang-Hui Fu. AWSMOTE: An SVM-Based Adaptive Weighted SMOTE for Class-Imbalance Learning. *Hindawi Volume 2021, Article ID 9947621,* Published 13 May 2021.

Wenhao Xie,1,2 Gongqian Liang,1 Zhonghui Dong,3 Baoyu Tan,4 and Baosheng Zhang5 An Improved Oversampling Algorithm Based on the Samples’ Selection Strategy for Classifying Imbalanced Data 06 May 2019

Nitesh V. Chawla, K. W. B. L. O. H. P. K., 2002. SMOTE: Synthetic Minority Over-sampling. Technique. *Journal of Artificial Intelligence Research 16 (2002).*

Haibo He, Y. B. E. A. G. a. S. L., 2008. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced. *International Joint Conference on Neural Networks (IJCNN),*

Marius-Constantin Popescu, Valentina Emilia Balas, 2009. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems.*

Available at: <https://www.researchgate.net/publication/228340819_Multilayer_perceptron_and_neural_networks>

Gandhi, R., 2018. *support-vector-machine-introduction-to-machine-learning-algorithms.*

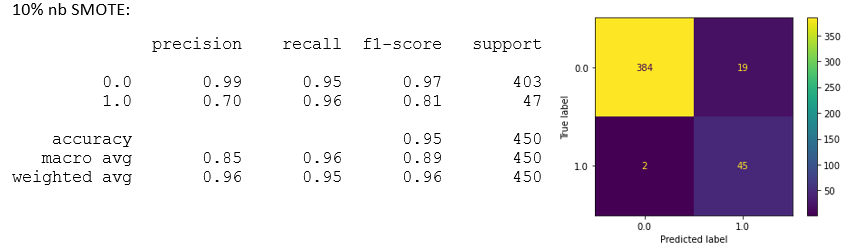
Available at:<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

Webb, G.I. (2011). “Naïve Bayes” *Encyclopedia of Machine Learning* https://doi.org/10.1007/978-0-387-30164-8\_576

# Appendices:

Confusion matrix

NB:

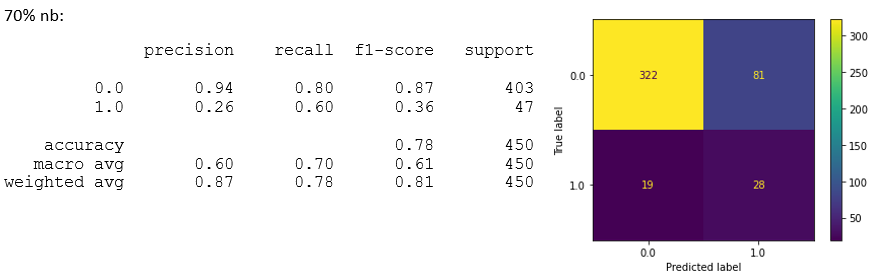


Table

Description automatically generated

Table

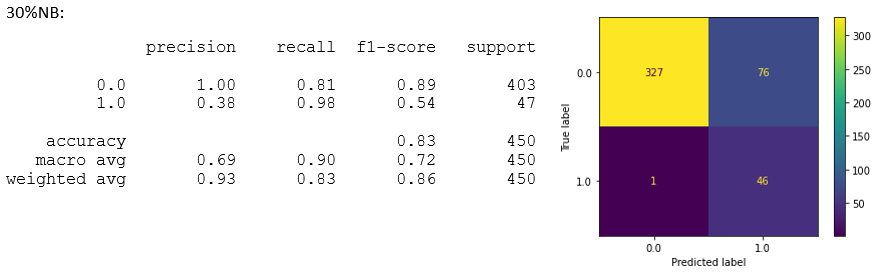
Description automatically generated



ADASYN:

Table

Description automatically generated



Table

Description automatically generated

Table

Description automatically generated with low confidence

**Proposed Algorithm (SVM):**

Table

Description automatically generated

Table

Description automatically generated

50% NB:

Table

Description automatically generated

70% NB:

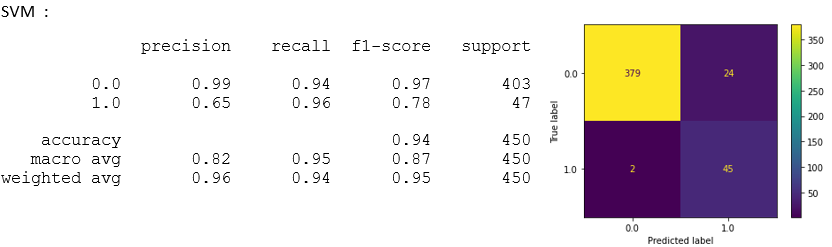
Table

Description automatically generated

SVM:

**SMOTE:**

10%

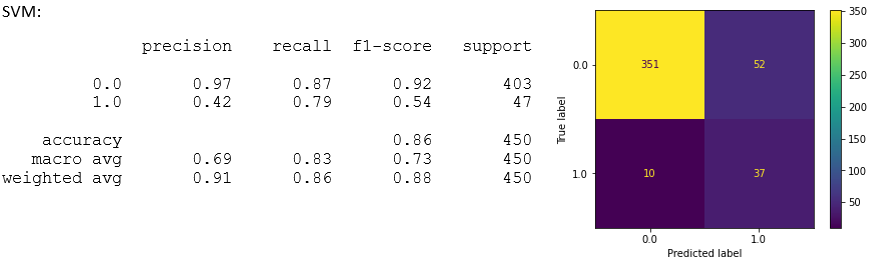


30%

Table

Description automatically generated

50%



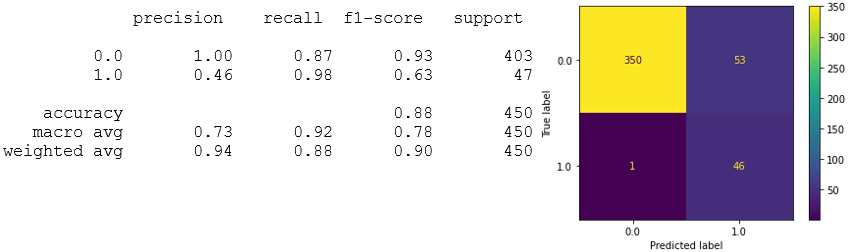
70%

Table

Description automatically generated

**ADASYN:**

10%



30%

Table

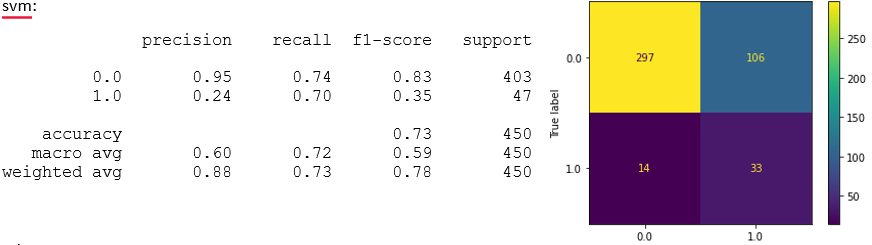
Description automatically generated with medium confidence

50%

Table

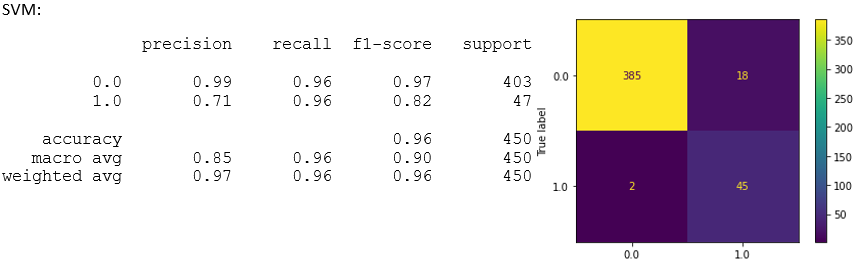
Description automatically generated with low confidence

70%

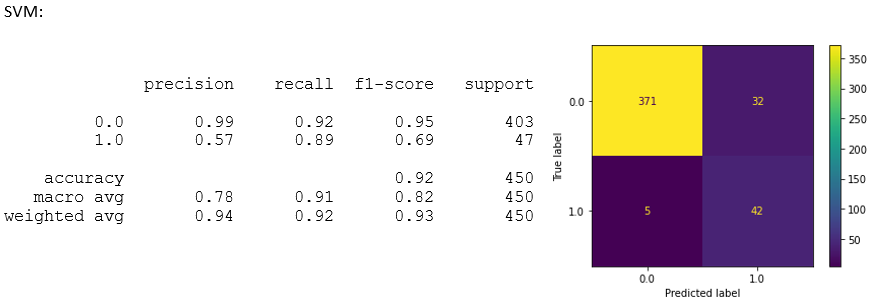


**Proposed Algorithm (SVM):**

10%:



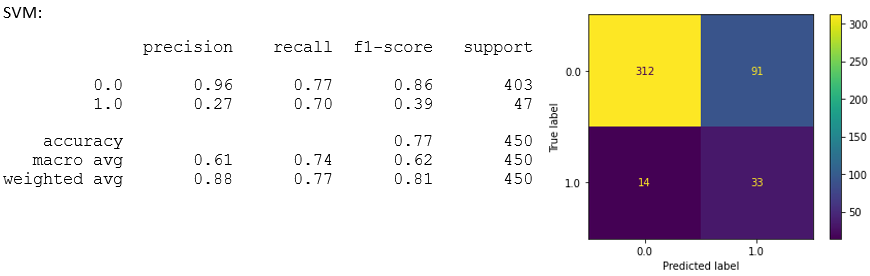
30%



50%

Table

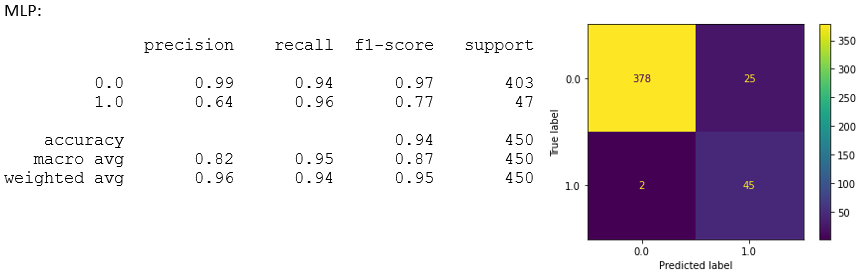
Description automatically generated with low confidence

70%

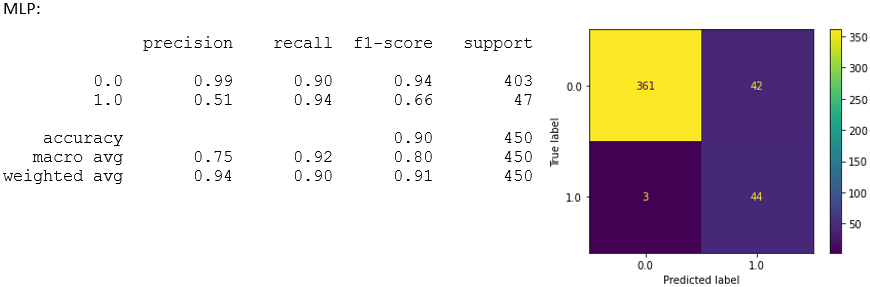
MLP:

**SMOTE:**

10%:



30%:



50%:

Table

Description automatically generated

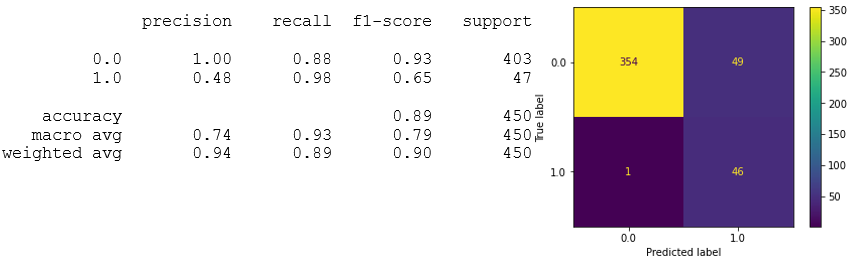
70%:

Table

Description automatically generated

**ADASYN:**

10%



30%

Table

Description automatically generated

50%:

Table

Description automatically generated

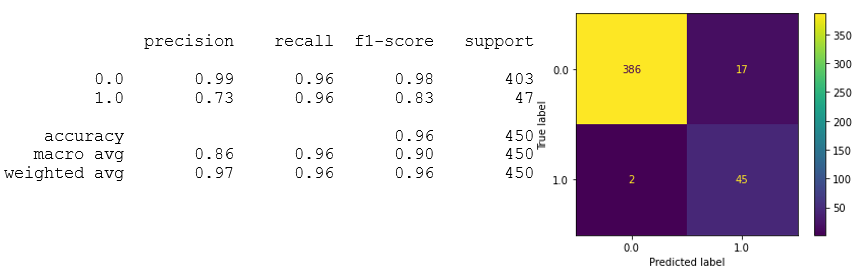
70%:

Table

Description automatically generated

**Proposed Algorithm (SVM):**

10%:



30%:

Calendar

Description automatically generated with low confidence

50%:

Table

Description automatically generated

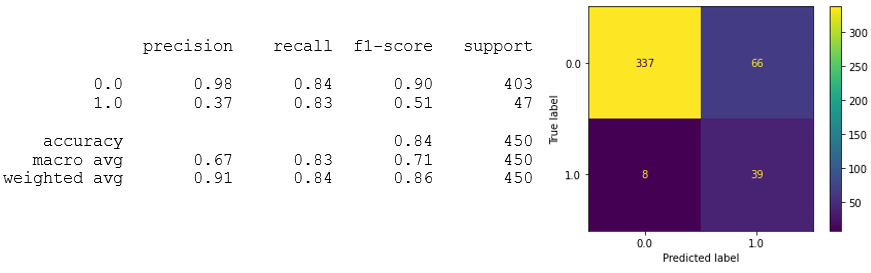
70%:

Table

Description automatically generated

**Decision Boundary**

Decision Boundary > -2: (NB)



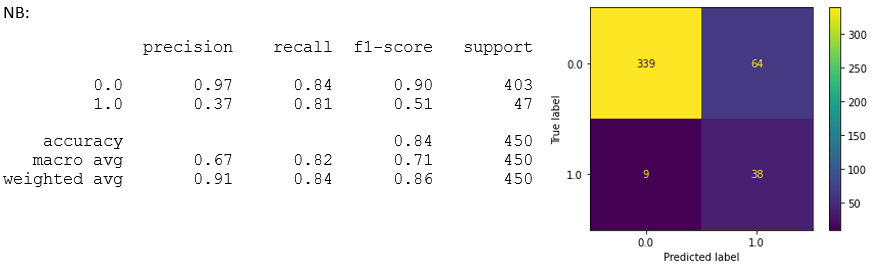
Table

Description automatically generated

Table

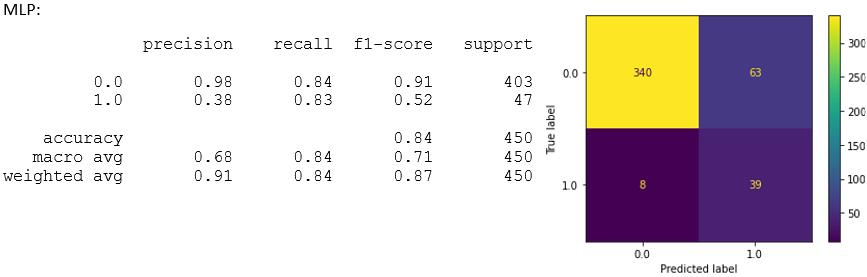
Description automatically generated

Decision Boundary > -1:

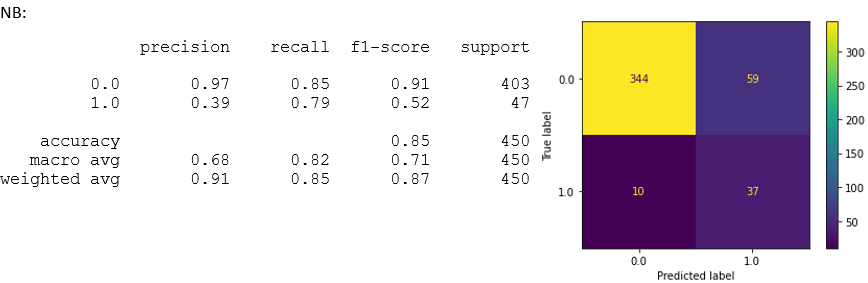


Table

Description automatically generated

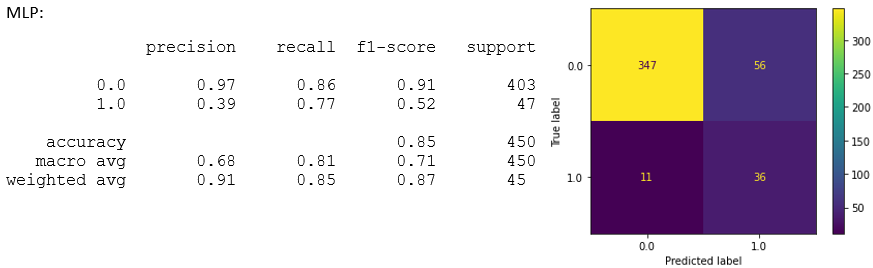


Decision Boundary > 0 :



Table

Description automatically generated



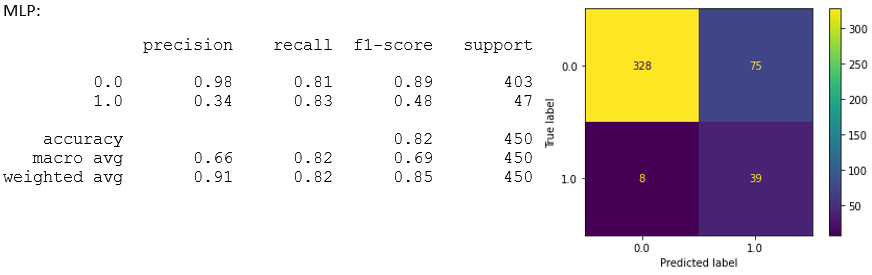
Decision Boundary > 1:

Table

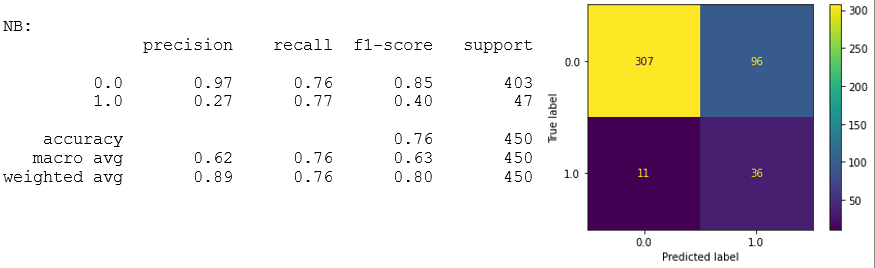
Description automatically generated with medium confidence

Table

Description automatically generated



Decision Boundary > 2:



Table

Description automatically generated

Table

Description automatically generated