8-2023

GENETIC PROGRAMMING TO OPTIMIZE PERFORMANCE OF MACHINE LEARNING ALGORITHMS ON UNBALANCED DATA SET

Asitha Thumpati

GENETIC PROGRAMMING TO OPTIMIZE PERFORMANCE OF MACHINE LEARNING ALGORITHMS ON UNBALANCED DATA SET

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Asitha Thumpati

August 2023

GENETIC PROGRAMMING TO OPTIMIZE PERFORMANCE OF MACHINE LEARNING ALGORITHMS ON UNBALANCED DATA SET

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Asitha Thumpati

August 2023

Approved by:

Dr. Yan Zhang, Committee Advisor, Computer Science and Engineering

Dr. Jennifer Jin, Committee Member

Dr. Amir Ghasemkhani, Committee Member

ABSTRACT

Data collected from the real world is often imbalanced, meaning that the distribution of data across known classes is biased or skewed. When using machine learning classification models on such imbalanced data, predictive performance tends to be lower because these models are designed with the assumption of balanced classes or a relatively equal number of instances for each class. To address this issue, we employ data preprocessing techniques such as SMOTE (Synthetic Minority Oversampling Technique) for oversampling data and random undersampling for undersampling data on unbalanced datasets. Once the dataset is balanced, genetic programming is utilized for feature selection to enhance performance and efficiency.

For this experiment, we consider an imbalanced bank marketing dataset from the UCI Machine Learning Repository. To assess the effectiveness of the technique, it is implemented on four different classification algorithms: Decision Tree, Logistic Regression, KNN (K-Nearest Neighbors), and SVM (Support Vector Machines). Various metrics including accuracy, balanced accuracy, recall, F-score, ROC (Receiver Operating Characteristics) curve, and PR (Precision-Recall) curve are compared for unbalanced data, oversampled data, undersampled data, and cleaned data with Tomek-Links for each algorithm.

The results indicate that all four algorithms perform better when oversampling the minority class to half of the majority class and undersampling the majority class examples to match the minority class, followed by performing Tomek-Links on the balanced dataset.

ACKNOWLEDGEMENTS

I wish to express my deep sense of gratitude to my project advisor Dr. Yan Zhang, Computer Science of Engineering for her able guidance and useful suggestions, which helped me in completing the project work, in time. She has been a great source of inspiration. I would like to thank my project committee members Dr. Jennifer Jin and Dr. Amir Ghasemkhani for their continuous support. I also thank all my faculty and university who have helped me through this entire process of completing my degree.

Finally, yet importantly, I would like to express my heartfelt thanks to my beloved parents for their blessings and continuous encouragement, my classmates for their help and wishes for the successful completion of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Classification problems are one of the tasks performed by machine learning where a program uses data set or observations provided to learn how to categorize new data into respective classes or labels. Many algorithms have evolved over time to perform this task, but these algorithms are built on top of the assumption that the provided learning data set has equal number of instances over different classes and identical misclassification cost.

Advancements in science and technology have led to a rapid increase in the generation and accessibility of raw data. However, this data collected from various sources is often highly imbalanced. Datasets with a significant disparity in the number of instances between different classes are considered imbalanced datasets.

1.1 Problem Statement

In a classification data set if the distribution of the examples across the known classes is biased or skewed it is considered as an imbalanced data set. Class imbalance can either be a binary imbalance or a multiclass imbalance with constraints on time and resources we will be focusing on the binary imbalance problem in this paper. In binary imbalance data we have an enormous number of samples set for one class and comparatively a smaller number of samples for the other class called majority class and minority class, respectively.

The class imbalance ratio is a metric used to assess how highly the data is skewed. It is given by the ratio of the sample size of the majority class to the sample size of the minority class. A data set having a high class imbalance ratio is skewed towards the majority class. Machine Learning models are built to perform better on balanced datasets, when such a skewed dataset is fed into a regular classification model the model's predictions are biased towards the majority class as there are a smaller number of records to learn from the minority class for the model. Additionally, these classification models assign an equal misclassification error for both false negative and false positive, which does not benefit the model while training on imbalanced data sets. Two of the common ways to handle this problem are balancing the imbalanced classes at data level by implementing data preprocessing techniques or handling at algorithmic level by incorporating advanced techniques like bagging and boosting into regular classifier. When metrics like accuracy are used to assess the predictability of the model it is highly biased towards the majority class with the majority class having accuracy close to 100% and the minority class having accuracy between 0% to 10%. As [1] mentioned despite intense work on imbalance learning over the past two decades it still remains an open problem that has to be identified and addressed with respect to the specific data set. In this project, we will be using a combination of advanced preprocessing techniques to balance the data set along with genetic programming for feature selection.

1.2 Organization of Project

In Chapter 2 we will be discussing the existing solutions to the class imbalance problem. With that knowledge in Chapter 3 we will be discussing the proposed solution to this same problem. Chapter 4 evaluates these techniques on different classification models by comparing the chosen metrics and finally Chapter 5 gives a conclusion of the proposed system.

CHAPTER TWO

LITERATURE REVIEW

To learn more about this issue we explore the existing solutions for imbalanced data classification.

In [1] the author uses a combination of preprocessing and ensemble techniques on imbalanced dataset. Preprocessing techniques SMOTE and random under sampling are applied independently on the data set and each modified data set is fed into ensemble classification models random forest and XGboost [6]. By comparing various metrics like AUC score, sensitivity and specificity the author concludes using SMOTE along with random forest yields highest sensitivity of 79.19% [1].

In [2] the author uses Tomek-Links on the original imbalance data set to remove any noise from the data set. Random under sampling and SMOTE are applied on this data set to assess the performance of different classification models on these preprocessed techniques. The author states that Using Tomek-Links and random under sampling as combined sampling method has an improved performance in terms of specificity, w-accuracy, precision, G-mean and F-statistics using SVM, ANN, random forest and logistic regression [2].

In [3] the authors proposed two techniques based on clustering majority class to handle the imbalance data set. The first technique involves grouping the majority class instances into clusters that are equal in number with the minority

class data points. A well-known technique called K-Nearest Neighbors is used to form these clusters and calculate the centroid of each cluster which is used as a new instance in the majority class. In the second technique instead of considering centroid of the cluster as a new data point, an existing record from the original data set which has shortest euclidean distance between the center of the cluster and the record itself is selected [5]. To evaluate the effectiveness of these two techniques five classification models are chosen along with AdaBoost algorithm for ensemble learning. By conducting these two experiments on forty four small scale and two large scale datasets the author concludes that using the second technique to under sample the majority class and using MLP(multilayer perceptron) as classifier gave better accuracy and area under ROC curve score for all other tested combinations.

In [4] a preprocessing technique using Genetic Programming(GP) is proposed for feature selection and feature construction [8]. Using filters and genetic programming effective features are selected and multiple features are constructed from the original data set named as FS and FCM algorithms [4]. Considering these two as the base algorithms a combined technique called FCMFS is implemented where multiple features are constructed from existing features using FCM and then the most effective features are selected with FS. Experimental results from nine different datasets showed that FS and FCM gave higher performance score with comparison to original feature sets whereas

FCMFS surpassed these two algorithms by giving higher classification accuracy with reduced number of features thereby improving the performance.

CHAPTER THREE

PROPOSED METHOD

In the proposed system, we will be handling the discussed imbalance data problem at data level by combining two data preprocessing techniques data oversampling and undersampling. In this chapter, discussion about the data source ,preprocessing techniques and genetic programming algorithms used to implement the proposed system is done.

3.1 Data Source

To perform the proposed technique an imbalanced data set from the UCI machine learning repository called Bank Marketing data set is chosen. This data is collected by a Portuguese banking institution during a direct marketing campaign through phone calls where its clients are recommended to subscribe for a bank term deposit. The classification problem would be to predict if a client would subscribe to the term deposit based on given input features. The original file bank-full.csv has 45211 records with sixteen input attributes and one class variable. Table.1 gives detailed information about each individual client .It contains data related to user demographics like age, education, marital status, job etc. along with users financial information like bank balance, home or personal loan and data related to campaign call. The output variable ‘y’ has a class value *yes* as the client subscribing to the term deposit and *no* as the client

not opting for the term deposit. Out of 45211 records 39922 instances are classified as no resulting in the majority class and 5289 as yes pertaining to the minority class i.e. 89% of data consist of majority class no. The class imbalance ratio for this data set is 7.5. Figure 1 shows an instance of the original data set.

Table 1. Information about the Dataset Features

Attribute name	**Description**	**Variable type**	**values**
age	Age of the client	numeric	18-95
job	Type of job	categorical	"admin.","unknown","unemployed","management","housemaid","entrepreneur","student","blue-collar","self-employed","retired","technician","services"
marital	Marital status	categorical	"married","divorced","single"; note: "divorced" means divorced or widowed
education	Level of education a person received	categorical	"unknown","secondary","primary","tertiary"
default	has credit in default?	categorical	“yes”,”no”
balance	average yearly balance, in euros	numeric	-8019 to 102127
housing	has housing loan?	categorical	“yes”,”no”
loan	has personal loan?	categorical	“yes”,”no”
contact	How a person was contacted	categorical	"unknown","telephone","cellular"
day	last contact day of the month	numeric	1-31
month	last contact month of year	categorical	"jan", "feb", "mar", ..., "nov", "dec"
duration	last contact duration, in seconds	numeric	0 – 4918 sec

campaign	number of contacts performed during this campaign and for this client	numeric	1-63
pdays	number of days that passed by after the client was last contacted from a previous campaign	numeric	-1 to 871 .numeric, -1 means client was not previously contacted
previous	number of contacts performed before this campaign and for this client	numeric	0-275
poutcome	outcome of the previous marketing campaign	categorical	"unknown","other","failure","success"
y	has the client subscribed a term deposit?	categorical	“yes”,”no”

age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no
35	management	married	tertiary	no	231	yes	no	unknown	5	may	139	1	-1	0	unknown	no
28	management	single	tertiary	no	447	yes	yes	unknown	5	may	217	1	-1	0	unknown	no
42	entrepreneur	divorced	tertiary	yes	2	yes	no	unknown	5	may	380	1	-1	0	unknown	no
58	retired	married	primary	no	121	yes	no	unknown	5	may	50	1	-1	0	unknown	no
43	technician	single	secondary	no	593	yes	no	unknown	5	may	55	1	-1	0	unknown	no

Figure 1. An Instance of Original Dataset

3.2 Data Preprocessing and Feature Selection

Data preprocessing is one of the initial and crucial steps of the machine learning workflow. As bank marketing data set contains a combination of categorical, discrete and continuous attributes label encoding is used to convert all attributes to numeric type which reduces the burden on the processor. Duration which has values in seconds is converted into minutes to reduce the range value. Considering distance-based algorithms like K-Nearest Neighbors, Support Vector Machines and logistic regression it is important to reduce the feature range of balance and duration using standardizing technique standard scalar.

Our proposed system handles imbalanced data problems at the data level by oversampling the minority class using SMOTE and passing this modified data to random undersampling such that the final class imbalance ratio is one, which is then cleaned with Tomek-Links [2, 9, 34]. Genetic Programming is used on top of this balanced and cleaned data set to select the most efficient features for model training. This proposed system is compared with original data, oversample data, oversampled+cleaned data, undersampled data and undersampled+cleaned data.

In general oversampling duplicates the existing minority class data points and adds it to the original data set to minimize the imbalance ratio. Although this method decreases the imbalance ratio, adding identical examples to the data set increases data redundancy and false information. An advanced technique for

oversampling is introduced in [9] called as Synthetic Minority Oversampling Technique. SMOTE is built on top of data augmentation where synthetic data points are created from the original data set. Data augmentation is adding minor changes to the duplicated minority class instances in the direction of the original data points. Random space N is generated from the minority class and K-Nearest Neighbors are calculated for the observations in N. The distance between the observation and a selected random neighbor is multiplied by a number between zero and one which is then added to the original observation. This leads to a new data vector in the direction of the chosen neighbor. This helps to create data points that are not exact duplicates of the original data set [33].

Undersampling is one of the data resampling techniques where records from the majority class are reduced by keeping the minority class examples [10]. The random undersampling technique deletes majority class samples based on some randomness. It is advised to use random undersampling when the number of majority class records is abundant compared to the minority class [34].

Tomek-Links is one of the undersampling techniques which can also be used for data cleaning. It was developed by Ivan Tomek in the year 1976 by modifying Condensed Nearest Neighbors (CNN) [11]. CNN algorithm produces training set consistency from the original data set i.e., a subset that can accurately classify data points in the original data set [12,13]. Using randomness to select subset results in the retention of unnecessary samples and not including the boundary samples, to overcome this issue two modifications to CNN are

suggested [14]. Two modifications to CNN Chooses a pair of examples from the original data set such that they belong to different classes and have the minimum distance between them. This pair of examples are called Tomek-Links. Tomek-Links consists of boundary instances and noise instances which affect the learning curve. In an imbalanced data set Tomek-Links can be used to locate all the majority class samples that are nearest neighbors to the minority class which can then be removed from the data set [11,14].

It is important that the classification model learns from data that is not redundant and noise free for better performance. Feature selection is a technique used to handle such data where the most relevant and effective features are selected from the entire data set. Different methods have evolved over time for feature selection like filter based methods, wrapper methods and embedded methods [15].

Genetic Programming is an evolutionary algorithm based on the theory of evolution proposed by Charles Darwin. Genetic algorithm can be applied to a wide range of optimization problems. It can be applied to feature selection as we are trying to obtain the subset of features that best trains the model [35]. Every genetic algorithm has five stages Initial population, fitness function, selection, crossover and mutation. Initially, each solution to the problem is grouped into a set called population. Each individual in the population is represented by a sequence of binary bits zero and one called genes. A chromosome is a string of genes representing a solution in the population. The population for the feature

selection problem would be a power set of features in the original data set which can be restricted by selecting the number of features in the optimal solution. Next fitness function is defined and used to calculate the fitness value for each record in the population set [17]. This fitness function can be any performance metric used to evaluate the trained model like accuracy or error. The record with higher accuracy and lower error is assigned with higher fitness value compared to the record with lower accuracy and higher error. Based on this fitness value the individuals in the population are ranked from higher to lower fitness value. In the selection phase, the most fitted individuals are chosen for recombination to produce the next generation. Two individuals are selected from the result set of the selection phase as parents. Offspring are generated from these parents by interchanging the genes called crossover. These offspring are added to the population. The mutation is the final stage where the genes of the offspring are slightly altered to maintain diversity in the population and avoid early convergence [16]. The algorithm continues to perform the last four steps that are generating fitness value, selection, crossover and mutation until the optimal solution is achieved or the population starts converging [17].

3.3 Methodology

In this paper, we will be training four classification models on our proposed preprocessing and feature selection techniques. The well-known classification

models are decision tree, logistic regression, KNN and SVM. We will be seeing how each of these models function with training data.

3.3.1 Decision Tree

Decision Tree is a supervised non-parametric model that can be used for both classification and regression. A decision tree is a graphical representation of all possible solutions to a decision based on certain conditions on each note of a decision tree. It forms a condition on the features to separate all the classes contained in the data set to the fullest purity [18]. To measure the impurity of the node Gini index, entropy and information gain are used [19].

Gini index is the difference between one and some of squared probabilities of each class(P_j). It favors large partitions and is easy to implement [36].

$$gini(T) = 1 - \sum_{j=1}^{n} P_j^2 \qquad (1)$$

Entropy is used to measure the impurity or randomness of a data set. It is optimal to choose the split with minimal entropy value [37]. Entropy value for a node N is given by Equation (2).

$$Entropy(N) = \sum_{i}^{k} \frac{(N)_i}{N} Entropy\ (N)_i \qquad (2)$$

where k is the number of partitions the parent node N is split into, and $(N)_i$ is the number of instances in the partition.

Information gain is used to find the best feature which serves as a root node in terms of information gain, it is given by subtracting weighted entropy of

each branch from the original entropy of the data set. The node with highest information gain is considered for splitting [36]. Decision tree algorithms CART and ID3 use Gini index and information gain for node classification respectively [19].

3.3.2 Logistic Regression

Logistic Regression is a parametric statistical model used only for binary classification [7]. This model uses the likelihood function to estimate the weights of parameters by studying the training data to predict the dependent variable of new observations. The likelihood function is given by Equation (3).

$$L(\bar{\theta}) = \prod_{i=1}^{m} P\left(y^{(i)} \middle| x(\bar{\imath}) \cdot \bar{\theta}\right) \qquad (3)$$

Algorithms like gradient ascent are used to find the optimal weight vector $L(\bar{\theta})$ to maximize the likelihood function. A logistic or sigmoid function is used to map a real value to probability between zero and one which is given as in Equation (4).

$$\sigma(\mathbb{Z}) = \frac{1}{(1+e\wedge(-z))} \qquad (4)$$

Where Z represents the linear combination of the input features and their associated weights. When z is positive, $\sigma(\mathbb{Z})$ is close to one, indicating a high probability of the positive class. In contrast, when z is negative, $\sigma(\mathbb{Z})$ is close to zero, indicating a high probability of the negative class [20].

3.3.3 K-Nearest Neighbors

KNN(K-Nearest Neighbors) is a lazy learning algorithm as it just stores train data until test data is provided for prediction [21]. KNN classifies new

observations by considering its K nearest neighbors classes, so it is important to choose the optimal K value with respect to each data set. A smaller value of K makes the model sensitive to noise and a larger value of K leads to misclassification [21]. It uses distance metrics like Manhattan distance, Euclidean distance, Minkowski distance, etc. to calculate the nearest neighbors which require scaling of feature values to reduce bias [26]. The KNN model is effective when the training data is huge and noise prone.

3.3.4 Support Vector Machines

SVM (Support Vector Machine) is one of the advanced classification techniques using linear models. SVM finds a decision boundary (line or hyperplane) that divides the data points based on its classes. The distance between this hyperplane and the nearest data point of each class is considered as margin. To perform better on test data a hyperplane with the largest margin is preferred [22]. If the data points are not linearly separable at lower dimensions additional features are added to separate the values at higher dimensions. It is highly challenging to find a plane that perfectly separates all data points when dealing with data from real world, so a concept called soft margin is introduced where a small misclassification is allowed in training data while adding a penalty for misclassifying test data [23].

CHAPTER FOUR

SYSTEM DESIGN

In this chapter, we will be going over the entire process of converting imbalanced data to balanced data and the predefined classes used in this process. Section 4.3 shows how four different classifiers are trained on this balanced dataset and imported to the User Interface.

4.1 System Architecture

Figure 2 shows the architecture of proposed machine learning processes which encompasses the significant stages undertaken to convert raw and unprocessed data to balanced train dataset that is fed to classification models.

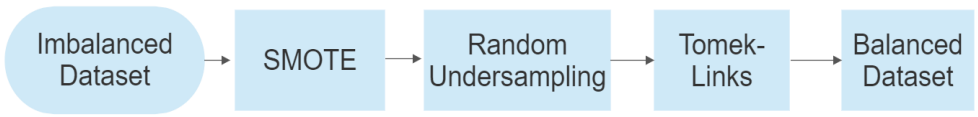


Figure 2. Proposed Design to Balance Imbalanced Dataset

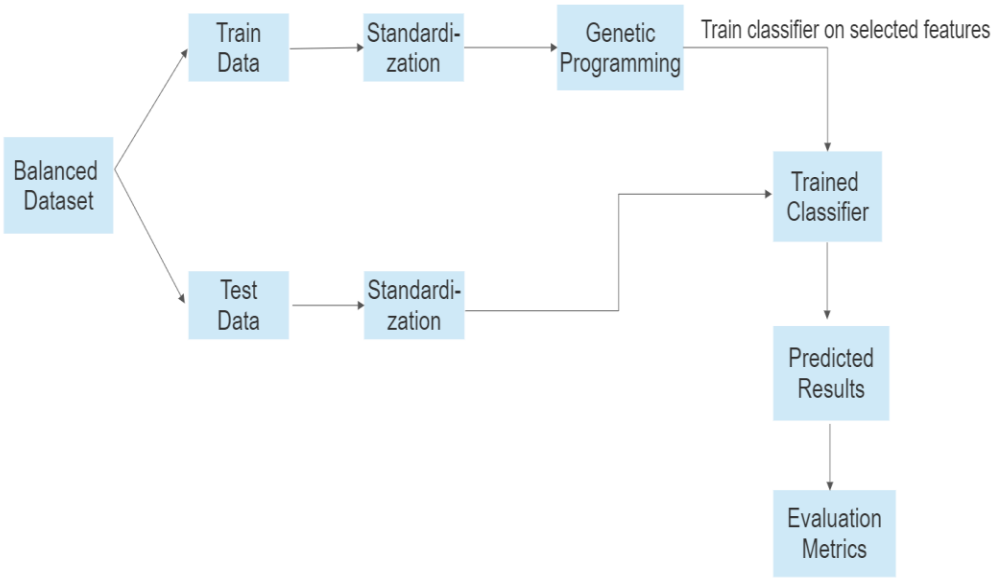


Figure 3. Training the Classification Model with Balanced Dataset

In Figure 3, we see how the entire balanced data is split into train and test data for training the classifier and testing its performance. Each sample space goes under normalization separately upon which train data is passed to genetic programming algorithm for selecting optimal subset of input features for each classifier to be trained on. This trained classifier is to predict the output of test data, which is used to compute the performance metrics.

4.2 Hardware and Software Requirements

Hardware Requirements

Random Access Memory : A minimum of 8 GB RAM

CPU : Intel core i7

Graphics Processing Unit : NVIDIA GPU from RTX series.

Operating System : Windows 10

Software Requirements

Programming Languages : python 3.8

Scripting Languages : HTML, CSS, Flask

Libraries : scikit-learn is used for classification models, imblearn is used for preprocessing dataset , sklearn-genetic-opt is used for feature selection, matplotlib, NumPy and pandas are used for mathematical calculations, pickle module is used to serialize the ML model.

IDE : Jupyter Notebook and Visual Studio Code.

4.3 Implementation

As shown in Figure 2 the imbalanced dataset is imported to Jupyter Notebook to perform the following operations analyzing the data, data preprocessing, balancing the data set, selecting the best features, and training classification models. Initial data preprocessing is done by converting all categorical data type features to discrete or ordinal data type features with label encoding. *Duration*, which gives information on how long the conversation lasted

with the client is converted to minutes unit from seconds there by reducing the range 0 to 90 minutes. This modified data is oversampled using *SMOTE* class from *imblearn.over_sampling.* The minority class samples are increased to half of the majority class samples by passing a value *0.5* to attribute *sampling_strategy* which is the ratio of desired minority class samples to majority class samples. This technique resampled the minority class from 5289 to 19961 instances by keeping the majority class records constant. This oversampled data is undersampled with *RandomUnderSampler* class from *imblearn.under_sampling* library by setting the attribute *sampling_strategy* to *one* which is given by number of desired samples in majority class after resampling divided by number of samples in minority class. Setting *sampling_strategy* as one results a balanced dataset of 19961 records in majority and minority class. Calculating class imbalance ratio for this dataset results in value one. This balanced data is cleaned using *TomekLinks* class from *imblearn.under_sampling* to remove noise data points. Border line noise data points are removed from both majority and minority class by setting *sampling_strategy* as ‘all’ which resulted in 19831 records for both the classes.

This resampled and cleaned data set is randomly divided into 60% and 40% for training the machine leaning algorithms and testing the performance of trained classifiers. Test data and train data are standardized separately using *StandardScalar* from *sklearn.preprocessing* library to shift the distribution to have zero mean and one standard deviation.

Figure 3 depicts how the train dataset is passed into *GAFeatureSelectionCV* class from *sklearn-genetic-opt* library to identify the optimal subset of features for each classification model chosen. This class provides numerous attributes out of which *estimator* and *scoring* are used in this project. The attribute *estimator* takes any classification model as input on which train data is learned and *scoring* acts as a fitness function to select the best subset of features for generating optimal solution. By setting the attribute *scoring* to *balanced_accuracy* the model is trained with optimal features until balanced accuracy score starts converging or deteriorating from the maximum value. This trained model is used to predict the output of test data to calculate the evaluation metrics.

Pipeline class from *sklearn*.*pipeline* is used to automate oversampling ,undersampling and data cleaning for the complete dataset i.e. before splitting the data for test and train purpose. Once the dataset is split, a pipeline is used to standardize train dataset and train the model on chosen features. This model is serialized with the help of *sklearn.pickle library* to export these trained classifiers to development environment visual studio code. Using *pickle.load* class these serialized *.pkl* files are imported to app.py file. With the help of HTML (Hyper-Text Markup Language) and CSS (Cascading Style Sheets) a UI (User Interface) is developed where user can select a model from four classifiers provided in a dropdown list as shown in Figure 4.

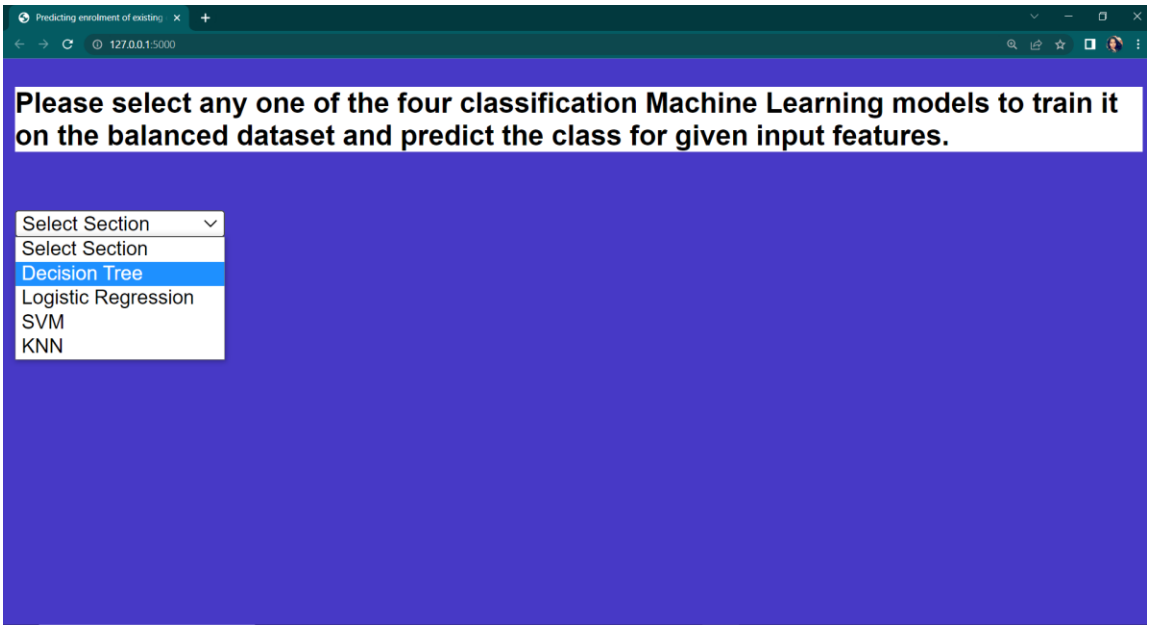


Figure 4. Home Page for the Application Where User Can Choose a Classification Model

The user's choice is stored in *selection* variable and passed to app.py file. Based on the obtained value it is navigated to the respective model.html page where user can provide input values for all 16 features as shown in Figure 5. Entered input values are converted into JSON (JavaScript Object Notation) format and transferred to app.py file where it is converted to a NumPy array, to pass as a variable to the pickled classification model. The output predicted by the model is stored in result variable and converted to JSON format. Figure 6 shows the output page where user can see the class predicted by the classifier for the given input values.

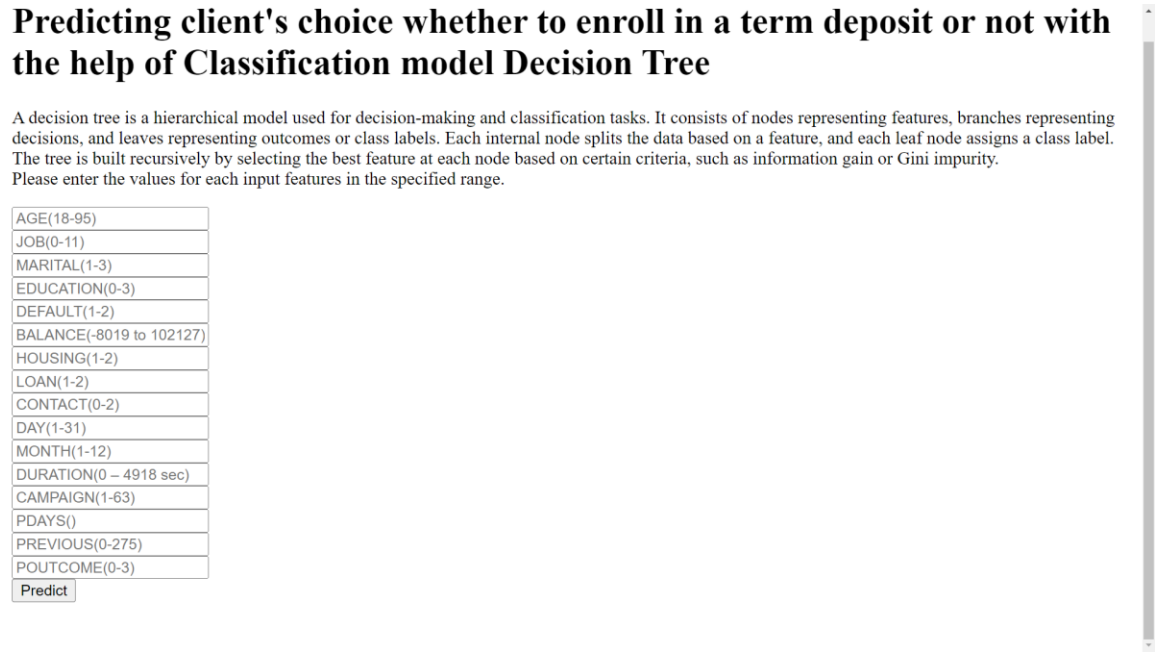


Figure 5. Predict Page Where User Can Give Inputs for the Selected Model

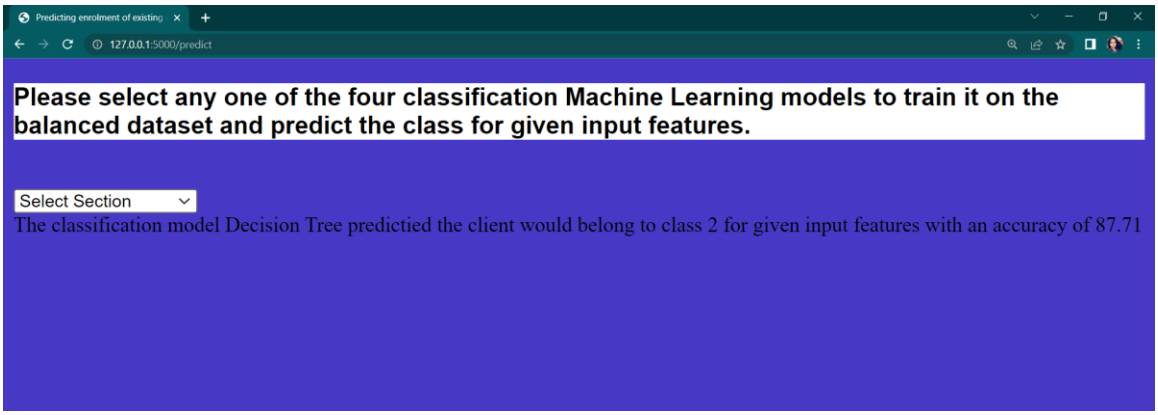


Figure 6. Predicted Result Page

CHAPTER FIVE

EXPERIMENTAL RESULTS

In order to assert if the proposed preprocessing techniques perform better than the original imbalanced dataset, it is important to identify appropriate metrics that are not biased towards a single class. These metrics are computed for each classifier at different stages of the resampling procedure. Chapter five section 5.1 gives detailed information on eight different metrics used in this project.

5.1 Evaluation Metrics

Different evaluation metrics are developed over time to assess the performance of machine learning models out of which accuracy, balanced accuracy, recall, F1 score, area under ROC curve, Receiver Operating Characteristics(ROC) curve, Precision-Recall(PR) Curve and Confusion Matrix are used in this project.

Accuracy measures the number of correct predictions made out of the total number of predictions by a trained model. Accuracy leads to misleading results when there is a high class imbalance in the dataset [27]. Balanced accuracy gives equal weight to each class, which is calculated by summing the recall for individual classes and dividing it by the number of classes [28]. Recall, also known as sensitivity or true positive rate, measures the ability to correctly identify positive or class of interest samples. It is calculated as the number of true

positives divided by the sum of true positives and false negatives, ranging from zero to one. A classification model with a higher recall value can predict most of the positive samples correctly [29]. Precision measures the consistency and stability of a model, calculated for the positive class as the number of true positives divided by the sum of true positives and false positives [29]. The F1 score simultaneously considers both recall and precision, given as the harmonic mean of recall and precision [30]. A model with an F1 score of one has perfect recall and precision, which is highly impractical to achieve in real-world data.

The ROC Curve is a widely used evaluation metric when dealing with imbalanced datasets. It is a graphical representation of how well a binary classification model can perform across different classification thresholds. It is calculated by plotting recall/TPR on the Y-axis and the corresponding FPR on the X-axis at different classification thresholds [31]. A point where TPR=1 and FPR=0 on the graph represents an ideal classification model. The area under the ROC Curve measures the separability or ability of a classifier to distinguish between positive and negative classes [32]. The area under the curve is calculated with the help of a trapezoid estimate. A model with a roc_auc_score of one can distinguish all the records perfectly correctly, whereas a model with a *roc_auc_score* of zero misclassifies every input. A classifier with a *roc_auc_score* between 0.5 and 1 distinguishes most of the records correctly. Nevertheless, if there is a significant imbalance in the distribution of classes,

ROC curves may provide an excessively positive assessment of a classifier's effectiveness. In such cases, Precision-Recall (PR) curves are suggested to address the issue of class distribution skew [24]. The PR curve for a model is given by plotting precision on the Y-axis and corresponding recall on the X-axis at different thresholds. As both precision and recall consider true positives, the PR curve is more focused on how well a model predicts minority class samples correctly. The confusion matrix shows the overall performance of a model in tabular form for easy interpretation, especially for imbalanced datasets [25]. It uses an NxN matrix for N different class classification, which shows actual and predicted outcomes for test data as true positive and true negative for correct predictions, and false positive and false negative for incorrect predictions.

5.2 Performance of Proposed System

The proposed preprocessing technique was evaluated on four classification models by comparing the performance metrics for each model. Tables 2 to 5 depict the performance of each classifier using different preprocessed techniques. The second row unbalanced data of Table 2 has evaluation metrics calculated on the original imbalanced data set which has 39922 majority class and 5289 minority class records. Applying SMOTE on this original data set increased the minority class records to 19961 on which the metrics resulted as noted in third row. SMOTE +TOMEK-Links has data that’s resampled using SMOTE and Tomek-Links on minority class resulting in 19795

minority records. Fourth row undersampled has records that are obtained by randomly deleting majority class samples from the original data set resulting in 12900 majority and 5289 minority records. Undersampling + TOMEK-Links has records that are resampled using random undersampling and Tomek-Links on majority class of original dataset resulting in 12060 majority and 5289 minority class samples. Sixth row overall balanced has records that are resampled by first oversample the minority class and passing the same data to undersample the majority class which gave a balanced records of 19961 in both the classes. Cleaning this balanced data with Tomek-Links resulted in Overall Balanced + TOMEK-Links having 19831 records .The last row Overall Balanced + TOMEK-Links+GP has metrics for classifier that is trained on overall balanced and cleaned data by using optimal set of input features from genetic programming algorithm. Tables 3, 4 and 5 show metrics for the same resampled data as in Table 2 for logistic regression, KNN and SVM classifiers respectively.

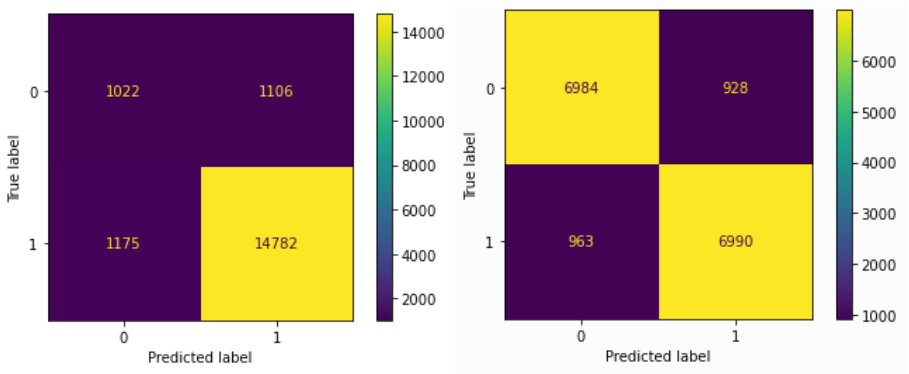
All four classification algorithms showed improved performance when trained on balanced data compared to the original unbalanced dataset. Figures 4 to 7 show how the performance of predicting minority class samples increased when compared to original dataset with the help of confusion matrix. It is observed that the decision tree model performed the best when trained on oversampled, undersampled, and cleaned data. Comparing the recall and F1 score of the proposed model to the original data reveals that these values nearly doubled, indicating better prediction by the model. Among the four classification

models, the KNN classifier exhibited the highest recall and F1 score of 0.94 and 0.9, respectively.

Table 2 shows that decision tree classifier has highest balanced accuracy, recall, F1 score and area under ROC curve for the proposed technique, which is 0.8834, 0.8828, 0.8831 and 0.8834 respectively. Figures 7 to 10 show the confusion matrix for original imbalanced dataset and balanced dataset with proposed technique. Figures 11 to 14 illustrates how the balanced accuracy varied across different generations during the evolution of the best features with respect to each binary classifier. Plotting the population generated with a frequency of 10 on X-axis and respective fitness value i.e. balanced accuracy achieved for that particular population on Y-axis for decision tree classifier resulted the graph in Figure 11. Comparing the graphs for all four classifiers we see that balanced accuracy for initial population is the lowest value. Once this fitness value starts to converge the generation of new population is halted and the best population that generated highest balanced accuracy is used to predict the test data.

Table 2. Evaluation Metrics for Decision Tree Classifier

Decision Tree	Accuracy	Balanced Accuracy	Recall	F1 Score	roc_auc_score
Unbalanced Data	0.8729	0.7007	0.4755	0.4684	0.7007
SMOTE	0.8869	0.8748	0.8386	0.8318	0.8748
SMOTE +TOMEK-Links	**0.8882**	0.8757	0.8384	0.833	0.8757
Undersampling	0.8081	0.7676	0.6751	0.6644	0.7676
Undersampling + TOMEK-Links	0.8247	0.7887	0.7	0.7034	0.7887
Overall Balanced	0.8737	0.8737	0.8754	0.874	0.8737
Overall Balanced + TOMEK-Links	0.8771	0.8771	0.876	0.8767	0.8771
Overall Balanced + TOMEK-Links+GP	0.8834	**0.8834**	**0.8828**	**0.8831**	**0.8834**

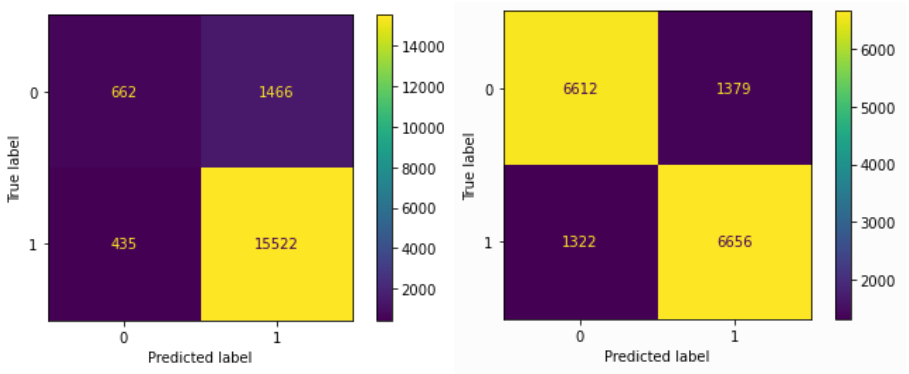


Unbalanced Dataset *Balanced Dataset*

Figure 7. Comparison of Confusion Matrix for Decision Tree Model Trained on Unbalanced and Balanced Dataset

Table 3. Evaluation Metrics for Logistic Regression Classifier

Logistic Regression	Accuracy	Balanced Accuracy	Recall	F1 Score	roc_auc_score
Unbalanced Data	**0.8984**	0.6419	0.311	0.4105	0.6419
SMOTE	0.8196	0.7817	0.668	0.7118	0.7817
SMOTE+TOMEK-Links	0.8236	0.7842	0.6667	0.7154	0.7842
Undersampling	0.8287	0.758	0.5964	0.6621	0.758
Undersampling + TOMEK-Links	0.8425	0.7861	0.6475	0.7093	0.7861
Overall Balanced	0.8308	**0.8308**	**0.8274**	**0.8303**	**0.8308**
Overall Balanced + TOMEK-Links	0.8305	0.8304	0.8234	0.8289	0.8304
Overall Balanced + TOMEK-Links+GP	0.8301	0.8301	0.8229	0.8285	0.8301

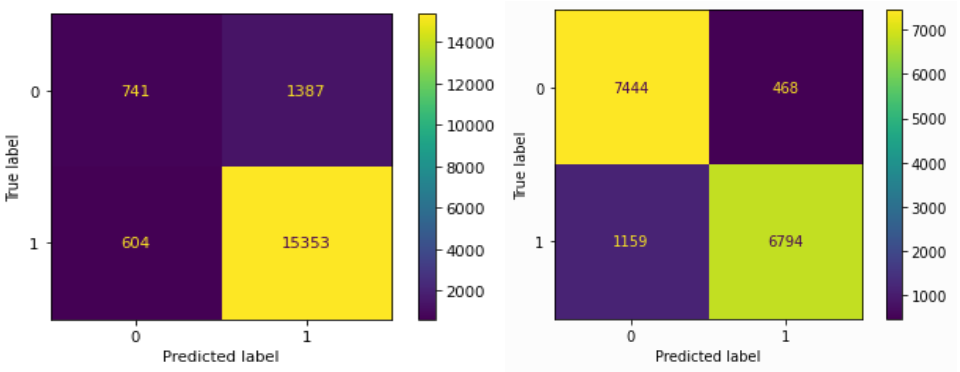


Unbalanced Dataset *Balanced Dataset*

Figure 8. Comparison of Confusion Matrix for Logistic Regression Model Trained on Unbalanced and Balanced Dataset

Table 4. Evaluation Metrics for K-Nearest Neighbors Classifier

KNN	Accuracy	Balanced Accuracy	Recall	F1 Score	roc_auc_score
Unbalanced Data	0.8899	0.6551	0.3482	0.4267	0.6551
SMOTE	0.9023	0.9019	0.9007	0.8601	0.9019
SMOTE +TOMEK-Links	**0.9054**	**0.9056**	0.9062	0.8643	**0.9056**
Undersampling	0.8196	0.7516	0.5969	0.6501	0.7516
Undersampling + TOMEK-Links	0.8358	0.7789	0.6388	0.6979	0.7789
Overall Balanced	0.8968	0.8968	**0.941**	0.9013	0.8968
Overall Balanced + TOMEK-Links	0.8974	0.8975	0.9408	**0.9014**	0.8975
Overall Balanced + TOMEK-Links+GP	0.893	0.8931	0.9261	0.8962	0.8931

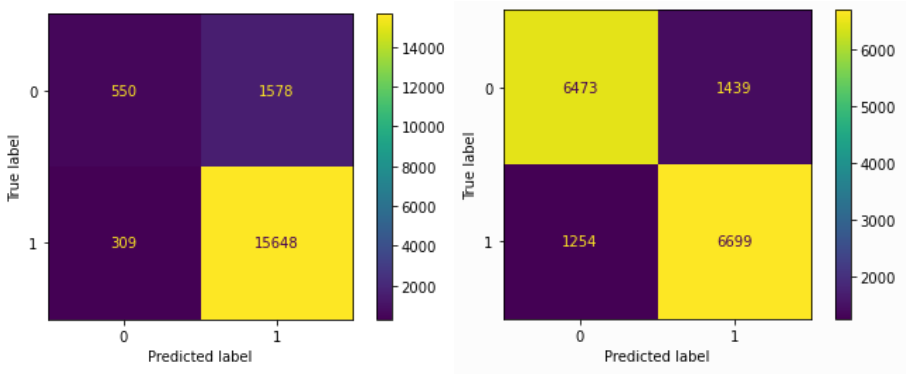


Unbalanced Dataset *Balanced Dataset*

Figure 9. Comparison of Confusion Matrix for K-Nearest Neighbors Model Trained on Unbalanced and Balanced Dataset

Table 5. Evaluation Metrics for Support Vector Machines Classifier

SVM	Accuracy	Balanced Accuracy	Recall	F1 Score	roc_auc_score
Unbalanced Data	**0.8956**	0.6195	0.2584	0.3682	0.6195
SMOTE	0.8168	0.7736	0.6438	0.7011	0.7736
SMOTE +TOMEK-Links	0.8203	0.7747	0.6384	0.7026	0.7747
Undersampling	0.8261	0.7466	0.5647	0.6463	0.7466
Undersampling + TOMEK-Links	0.838	0.7748	0.6194	0.6942	0.7748
Overall Balanced	0.8294	0.8294	**0.8195**	0.8278	0.8294
Overall Balanced + TOMEK-Links	0.8298	0.8298	0.8179	0.8274	0.8298
Overall Balanced + TOMEK-Links+GP	0.8302	**0.8302**	0.8181	**0.8279**	**0.8302**



Unbalanced Dataset *Balanced Dataset*

Figure 10. Comparison of Confusion Matrix for Support Vector Machine Model Trained on Unbalanced and Balanced Dataset

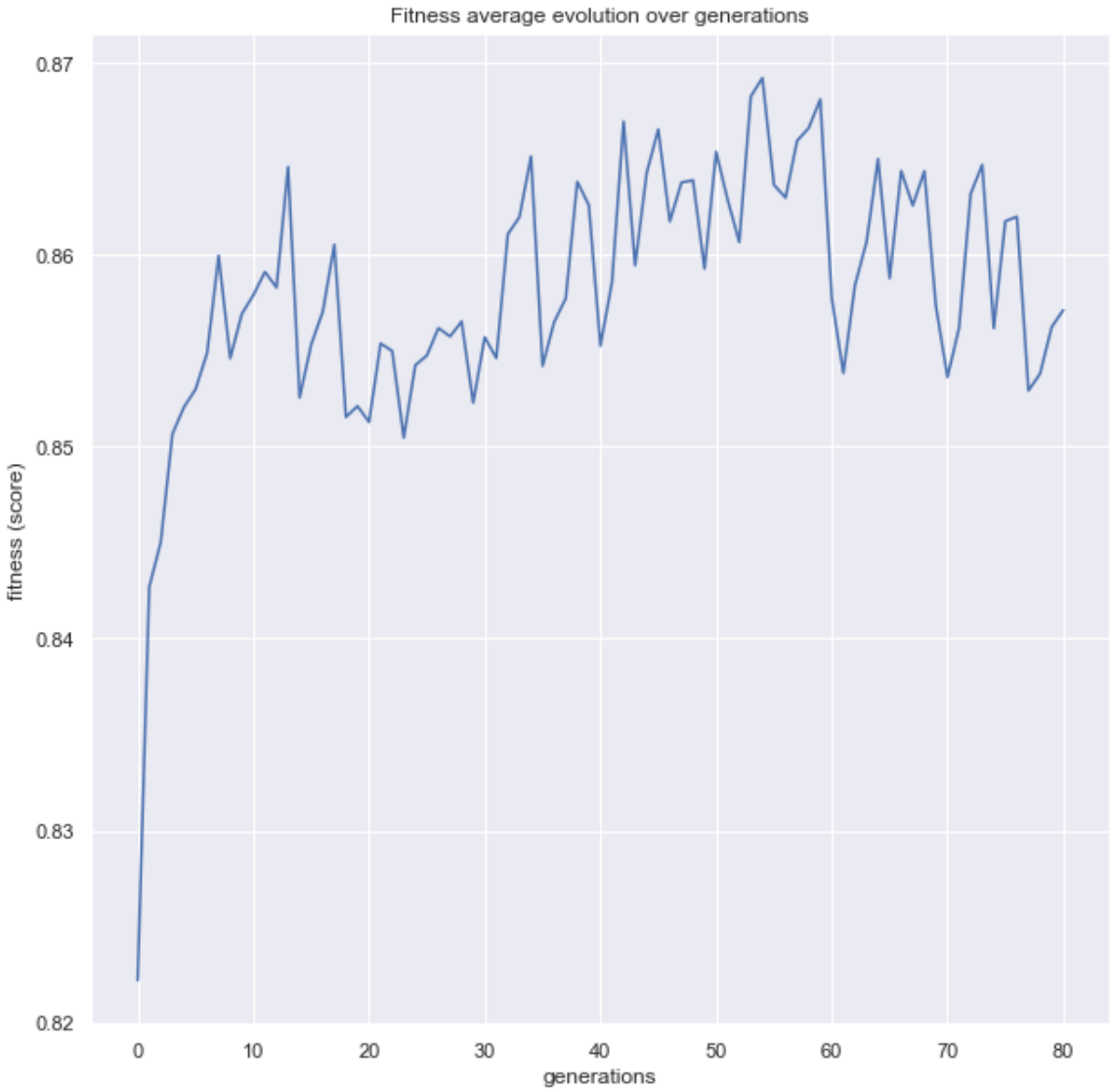


Figure 11. Balanced Accuracy over Evolved Generations with Decision Tree

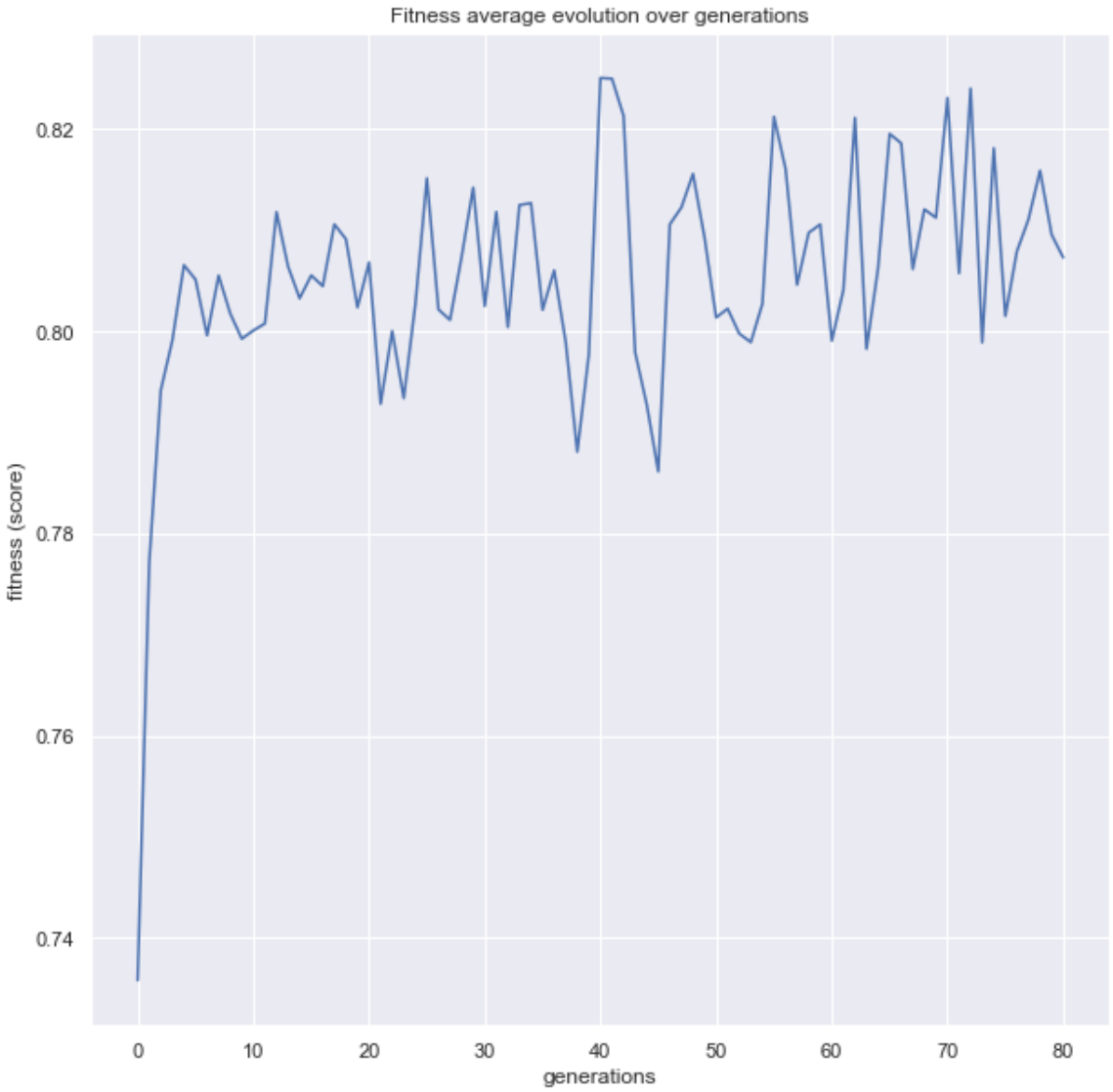


Figure 12. Balanced Accuracy over Evolved Generations with Logistic Regression

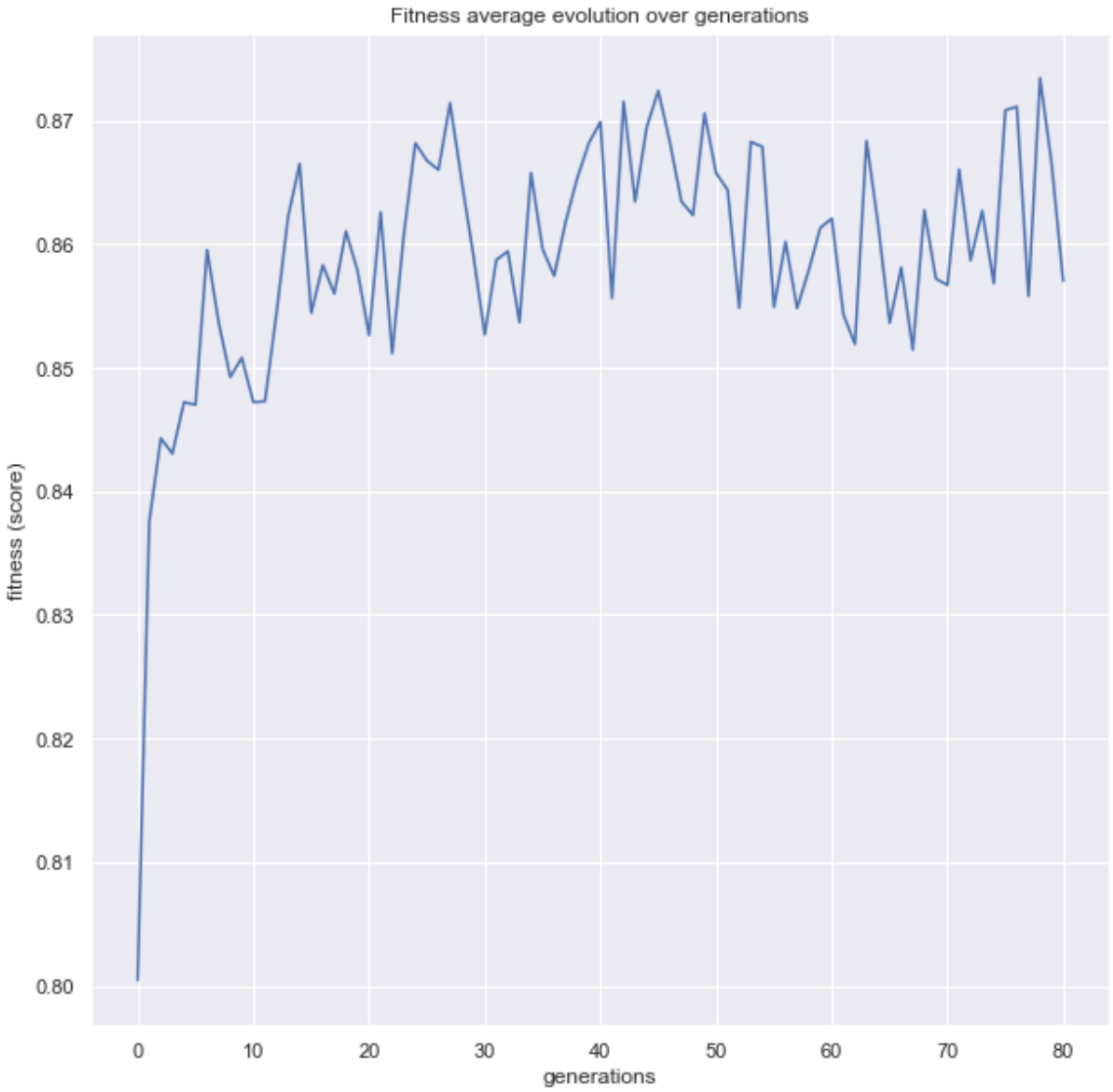


Figure 13. Balanced Accuracy over Evolved Generations with K-Nearest Neighbors

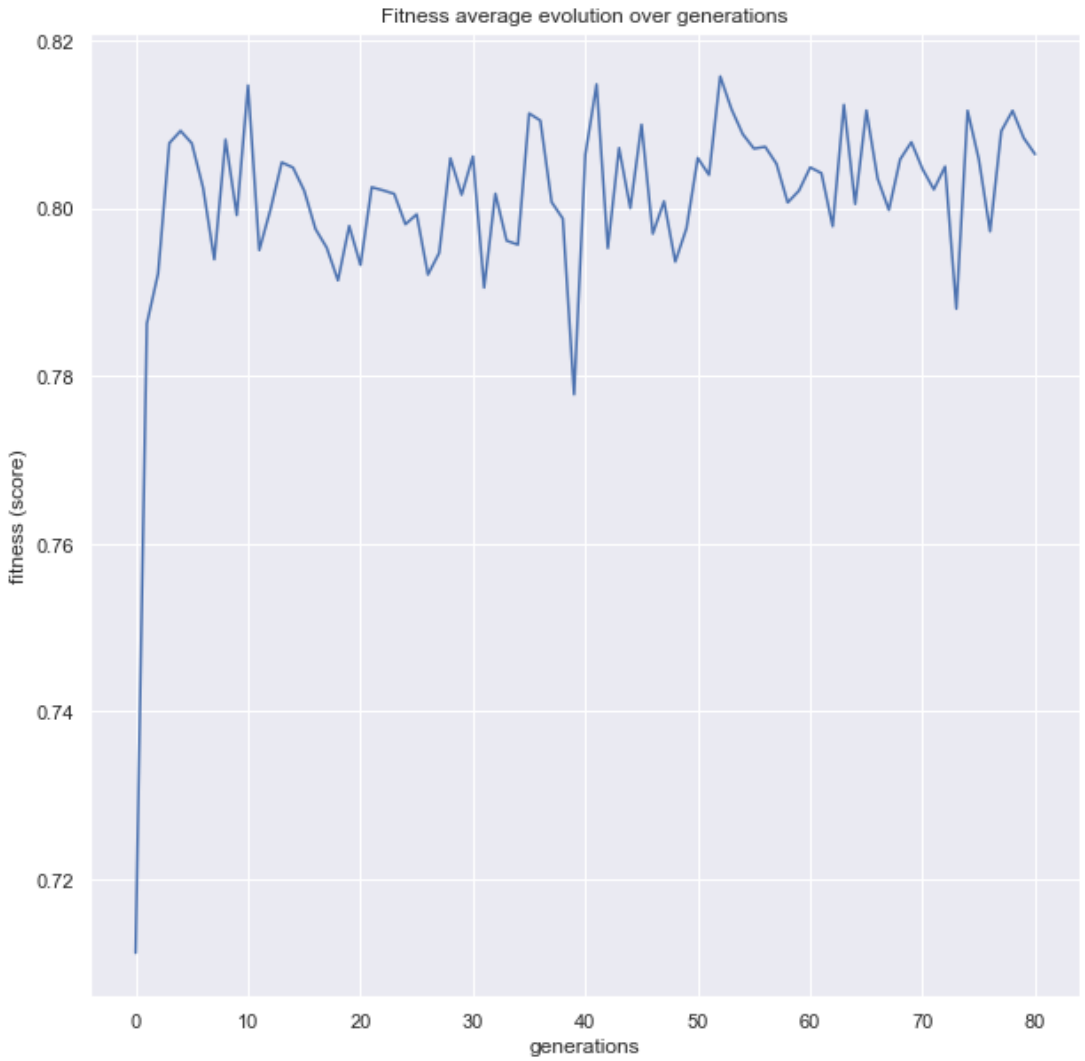


Figure 14. Balanced Accuracy over Evolved Generations with Support Vector Machines

CHAPTER SIX

CONCLUSION

In this project, a data preprocessing technique is proposed to handle the imbalanced data problem, which combines oversampling, undersampling, data cleaning, and feature selection. This technique is implemented on a bank marketing dataset to reduce the class imbalance ratio. Four classification models are trained on different balanced datasets to assess the performance of the proposed method. Metrics that are appropriate for imbalanced datasets, such as recall, F-1 score, balanced accuracy, and area under the ROC curve, are compared with existing preprocessing techniques and the proposed technique for each classifier. The results show that using both oversampling and undersampling as data preprocessing technique enhances the performance of all four classifiers. The decision tree and KNN algorithms achieved the highest balanced accuracy of 88.34% and 89.31%, respectively. As future work, this technique can be combined with other algorithmic-level data preprocessing techniques, such as ensemble machine learning classifiers. Additionally, besides using genetic programming for feature selection, further studies can explore the incorporation of this evolutionary algorithm into data-level preprocessing to improve the model's performance.

REFERENCES

[1] S. Mishra, “Handling Imbalanced Data : SMOTE vs. Random Undersampling”, International Research Journal of Engineering and Technology (IRJET), vol. 04, Aug. 2017. Accessed: Jun. 5, 2023. [Online]. Available: https://www.irjet.net/archives/V4/i8/IRJET-V4I857.pdf

[2] T. Elhassan, M. Tusneem, F. Aljourf, and M. Shoukri, "Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-sampling (RUS) as a Data Reduction Method", Global Journal of Technology and Optimization, vol. 01, 2016, Art. no. 10.4172/2229-8711.S1111.

[3] W. C. Lin, C. F. Tsai, Y. H. Hu, and J. S. Jhang, "Clustering-Based Undersampling in Class-Imbalanced Data", Information Sciences, vol. 409-410, pp. 17-26, 2017. DOI: 10.1016/j.ins.2017.05.008.

[4] J. Ma and X. Gao, "A Filter-Based Feature Construction and Feature Selection Approach for Classification Using Genetic Programming", Knowledge-Based Systems, vol. 196, pp. 105806, 2020, doi: 10.1016/j.knosys.2020.105806.

[5] K. Khamar, "Short Text Classification Using KNN Based on Distance Function", International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, no. 4, pp. 1916-1919, 2013.

[6] A. Asselman, M. Khaldi, and S. Aammou, "Enhancing the Prediction of Student Performance Based on the Machine Learning XGBoost Algorithm", Interactive Learning Environments, pp. 1-20, 2021.

[7] R. E. Wright, "Logistic Regression", in L. G. Grimm and P. R. Yarnold (Eds.), Reading and understanding multivariate statistics, pp. 217–244, American Psychological Association, 1995.

[8] W. B. Langdon and R. Poli, "Foundations of Genetic Programming", Springer Science & Business Media, 2013.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique", Journal of Artificial Intelligence Research, 2002, doi: 10.1613/jair.953.

[10] "What Is Undersampling?" mastersindatascience.org, Available: https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/. Accessed on: Jun. 5, 2023.

[11] I. Tomek, "Two Modifications of CNN", IEEE Trans. Systems, Man and Cybernetics, vol. 6, pp. 769-772, 1976.

[12] P. Hart, "The Condensed Nearest Neighbor Rule (corresp.)", IEEE Transactions on Information Theory, vol. 14, no. 3, pp. 515-516, 1968.

[13] A. Abhishek, "CNN (Condensed Nearest Neighbors)", medium.com, Available: https://abhic159.medium.com/cnn-condensed-nearest-neighbors-3261bd0c39fb. Accessed on: Jun. 5, 2023.

[14] J. Brownlee, "Undersampling Algorithms for Imbalanced Classification", machinelearningmastery.com, Available: https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification. Accessed on: Jun. 5, 2023.

[15] "What is Feature Selection? Definition and FAQs | HEAVY.AI", heavy.ai, Available: https://www.heavy.ai/technical-glossary/feature-selection. Accessed on: Jun. 5, 2023.

[16] V. Mallawaarachchi, "Introduction to Genetic Algorithms", towardsdatascience.com, Available: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3#:~:text=A%20genetic%20algorithm%20is%20a,offspring%20of%20the%20next%20generation. Accessed on: Jun. 5, 2023.

[17] F. Gomez, et al., "Genetic Algorithms for Feature Selection", neuraldesigner.com, Available: https://www.neuraldesigner.com/blog/genetic_algorithms_for_feature_selection. Accessed on: Jun. 5, 2023.

[18] A. Navada, A. N. Ansari, S. Patil and B. A. Sonkamble, "Overview of Use of Decision Tree Algorithms in Machine Learning", 2011 IEEE Control and System Graduate Research Colloquium, Shah Alam, Malaysia, 2011, pp. 37-42, doi: 10.1109/ICSGRC.2011.5991826.

[19] Y. Lu, T. Ye and J. Zheng, "Decision Tree Algorithm in Machine Learning", 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 2022, pp. 1014-1017, doi: 10.1109/AEECA55500.2022.9918857.

[20] S. Domínguez-Almendros, N. Benítez-Parejo, and A. R. Gonzalez-Ramirez, "Logistic Regression Models", Allergologia et Immunopathologia, vol. 39, no. 5, pp. 295-305, 2011. [Online]. Available: https://doi.org/10.1016/j.aller.2011.05.002.

[21] G. Guo, H. Gongde, D. Wang, Y. Bi, and K. Greer, "KNN Model-Based Approach in Classification", Lect. Notes Comput. Sci., vol. 2888, pp. 986-996, 2003. DOI: 10.1007/978-3-540-39964-3_62.

[22] Y. Zhang, "Support Vector Machine Classification Algorithm and Its Application", pp. 179-186, 2012. DOI: 10.1007/978-3-642-34041-3_27.

[23] Y. Hamasuna, Y. Endo, and S. Miyamoto, "Support Vector Machine for Data with Tolerance Based on Hard-Margin and Soft-Margin", in Proceedings of the 2008 IEEE International Conference on Fuzzy Systems (IEEE World Congress on Computational Intelligence), Hong Kong, China, 2008, pp. 750-755. DOI: 10.1109/FUZZY.2008.4630454.

[24] J. Davis and M. Goadrich, "The Relationship Between Precision-Recall and ROC Curves", in Proceedings of the 23rd International Conference on Machine Learning (ICML '06), Association for Computing Machinery, New York, NY, USA, 2006, pp. 233-240. https://doi.org/10.1145/1143844.1143874

[25] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The Impact of Class Imbalance in Classification Performance Metrics Based on the Binary Confusion Matrix", Pattern Recognition, vol. 91, pp. 216-231, 2019. https://doi.org/10.1016/j.patcog.2019.02.023.

[26] K. Chomboon, et al., "An Empirical Study of Distance Metrics for K-Nearest Neighbor Algorithm", in Proceedings of the 3rd International Conference on Industrial Application Engineering, vol. 2, 2015.

[27] G. Menardi and N. Torelli, "Training and Assessing Classification Rules with Imbalanced Data", Data Min Knowl Disc, vol. 28, pp. 92-122, 2014. [Online]. Available: https://doi.org/10.1007/s10618-012-0295-5.

[28] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The Balanced Accuracy and Its Posterior Distribution", in 2010 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010, pp. 3121-3124, doi: 10.1109/ICPR.2010.764.

[29] H. Shang, et al., "Precision/Recall on Imbalanced Test Data", in International Conference on Artificial Intelligence and Statistics, PMLR, 2023.

[30] A. Maratea, A. Petrosino, and M. Manzo, "Adjusted F-measure and Kernel Scaling for Imbalanced Data Learning", Information Sciences, vol. 257, pp. 331-341, 2014. https://doi.org/10.1016/j.ins.2013.04.016.

[31] C. D. Brown and H. T. Davis, "Receiver Operating Characteristics Curves and Related Decision Measures: A Tutorial", Chemometrics and Intelligent

Laboratory Systems, vol. 80, no. 1, pp. 24-38, 2006. [Online]. Available: https://doi.org/10.1016/j.chemolab.2005.05.004.

[32] F. Melo, "Area Under the ROC Curve", in W. Dubitzky, O. Wolkenhauer, KH. Cho, and H. Yokota (eds), Encyclopedia of Systems Biology, Springer, New York, NY, 2013, pp. 34-36. DOI: 10.1007/978-1-4419-9863-7_209.

[33] J. Brownlee, "SMOTE for Imbalanced Classification with Python", machinelearningmastery.com, Available: https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/. Accessed on: Jun. 5, 2023.

[34] T. Hasanin and T. Khoshgoftaar, "The Effects of Random Undersampling with Simulated Class Imbalance for Big Data", in 2018 IEEE International Conference on Information Reuse and Integration (IRI), 2018, DOI: 10.1109/IRI.2018.00051.

[35] N. Grafeeva, L. Grigorieva, and N. Kalinina-Shuvalova, "Genetic Algorithms and Genetic Programming", International Journal of Advanced Computer Science and Applications, vol. 3, no. 9, pp. 465-480, 2013.

[36] V. Jain, A. Phophalia and J. S. Bhatt, "Investigation of a Joint Splitting Criteria for Decision Tree Classifier Use of Information Gain and Gini Index", TENCON 2018 - 2018 IEEE Region 10 Conference, Jeju, Korea (South), 2018, pp. 2187-2192, DOI: 10.1109/TENCON.2018.8650485.

[37] M. Y. Lee and C. S. Yang, "Entropy-Based Feature Extraction and Decision Tree Induction for Breast Cancer Diagnosis with Standardized Thermograph

Images", Computer Methods and Programs in Biomedicine, vol. 100, no. 3, pp. 269-282, 2010.