Performance Analysis of Classification Algorithms by Implementing SMOTE Sampling Method

A Dissertation submitted to the Jawaharlal Nehru Technological University, Hyderabad in partial fulfilment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

Submitted by

SIRIGIRI DHARSHAN	**19B81A3310**
JALLEPALLI KARTHIK	**19B81A3316**
MANGLARAPU YASHWANTH	**19B81A3357**

Under the guidance of
Dr. C. Raghavendra
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

CVR COLLEGE OF ENGINEERING

(***An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad***)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

2022-2023

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)
Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the project report entitled **"Performance Analysis of Classification Algorithms by Implementing SMOTE Sampling Method"** bonafide record of work carried out by **Sirigiri Dharshan (19B81A3311), Jallepalli Karthik (19B81A3316)** and **Manglarapu Yashwanth (19B81A3357)** submitted to **Dr. C. Raghavendra** for the requirement of the award of **Bachelor of Technology** in **Computer Science and Information Technology** to the CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University, Hyderabad during the year 2022-2023.

Project Guide
Dr. C. Raghavendra
Associate Professor
Department of CSIT

Head of the Department
Dr. Lakshmi H N
Professor & HOD-ET

Project Coordinator

External Examiner

DECLARATION

We hereby declare that the project report entitled **"Performance Analysis of Classification Algorithms by Implementing SMOTE Sampling Method"** is an original work done and submitted to CSIT Department, CVR College of Engineering, affiliated to Jawaharlal Nehru Technological University Hyderabad in partial fulfilment for the requirement of the award of Bachelor of Technology in Computer Science and Information Technology and it is a record of bonafide project work carried out by us under the guidance of **Dr. C. Raghavendra**, Assistant Professor, Department of Computer Science and Information Technology.

We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

Signature of the Student

SIRIGIRI DHARSHAN

Signature of the Student

JALLEPALLI KARTHIK

Signature of the Student

MANGLARAPU YASHWANTH

Date:

Place:

ACKNOWLEDGEMENT

The success and outcome of this project require a lot of guidance and assistance from many people, and we are extremely privileged to have got this all along with the completion of our project. All that we have done is only due to such supervision and assistance and we should not forget to thank them.

We respect and thank our internal guide, **Dr. C. Raghavendra,** for providing us an opportunity to do this project work in CVR College of Engineering and giving us all the support and guidance, which made us complete the project duly. We are extremely thankful to her for providing such a nice support and guidance.

We would like to thank the Head of the Department, Professor **Dr. Lakshmi H N** for her meticulous care and cooperation throughout the project work. We thank **Dr. R. Raja**, Project Coordinator for providing us an opportunity to do this project and extending good support and guidance.

We are thankful for and fortunate enough to get constant encouragement, support, and guidance from all **Teaching staff of CSIT Department** which helped us in successfully completing our project work. Also, I would like to extend our sincere esteem to all staff in the laboratory for their timely support.

TABLE OF CONTENTS

List of Tables

List of Figures

ABBREVIATIONS

SMOTE	**Synthetic Minority Oversampling Technique**
SML	**Supervised Machine Learning**
SVM	**Support Vector Machine**
DT	**Decision Tree**
CM	**Confusion Matrix**
LR	**Logistic Regression**
XGBoost	**Extreme Gradient Boost**

ABSTRACT

Supervised Machine Learning (SML) is the search for algorithms that reason from externally given cases to build generalizations. The hypotheses are then used to make predictions about future events. One of the most common tasks performed by intelligent systems is supervised categorization. This project presents a variety of Supervised Machine Learning (ML) classification strategies, evaluates different supervised learning algorithms, and calculates the best effective classification algorithm based on the data set, number of instances, and variables (features). Six distinct machine learning algorithms were considered using the Confusion matrix approach as a machine learning performance evaluation tool: Support Vector Machine (SVM), Logistic Regression, Nave Bayes, Decision Tree, Random Forest, and XGBoost. In order to develop the algorithms, Predictive Maintenance of machine data collection was used. Predictive Maintenance AI4I 2020 is the data set to be used. And each algorithm employs the SMOTE sampling strategy. SMOTE is a sampling method that uses a k-Nearest Neighbor algorithm to generate synthetic data. SMOTE begins by selecting random data from the minority class, and then sets the data's k-nearest neighbors. The random data would be combined with the randomly chosen k-nearest neighbour to create synthetic data.

The goal is to compare classification algorithms in order to identify key features. Unbalanced data make developing a prediction model difficult. The Synthetic Minority Oversampling Technique (SMOTE) is used by researchers to address the issue of class imbalance. Following that, the machine learning models are trained and tested.

Keywords: Supervised Machine Learning, Classification algorithm, SMOTE, Machine Failure.

CHAPTER 1
INTRODUCTION

The Nature of the real time data is always in the form of imbalance as the occurrence of one event is always dominated by the other event. Here we using the SMOTE sampling method to overcome such problem. The dataset we taking is the predictive maintenance, it is binary categorical classification problem i.e., it has two ultimate outcomes such as machine failure and success

In addition to it, the selection of a classification algorithm to such kind of can be evaluated by using their results of the confusion matrix analysis

1.1 Motivation

In the concept of majority and minority in which one is dominated by other because the failure of a machine occurs in rare cases. if we feed the exact same data to the machine learning model it will overfit the data and gives us the errored data. To prevent such kind of result we implementing the SMOTE sampling method which is sampling method which deals with the problem of imbalanced datasets using synthetic sampling method

1.2 Problem statement

Predictive analytics drives Predictive Maintenance services. The initial goal of this technology is to detect and monitor equipment abnormalities and breakdowns to avoid catastrophic failure and downtime. This allows for the deployment of limited resources, increased device and equipment lifecycles, improved quality and supply chain procedures, and increased stakeholder satisfaction. The data of such problem involves the imbalance and it causes the erroneous result. And, the choosing the efficient and best algorithms among the other classification algorithms.

1.3 Problem Objectives

The goal of this project is to put the Synthetic Minority Over-sampling Technique into practise in order to address the problem of class imbalance that is present in the Predictive Maintenance dataset (SMOTE). Additionally, Support Vector Machine (SVM), Logistic Regression, Nave Bayes, Decision Tree, KNN classifier, and Random Forest were among

the ML techniques taken into account for this project. Each of these ML techniques was assessed individually for efficiency and classification accuracy**.**

1.4 General Overview of the project

In order to balance corrective and preventive maintenance, predictive maintenance relies on "just in time" component replacement or repair. Only when a part is within a failure prediction window is it replaced using this method. Using sensor readings or other data, this forecast window may be based. Unlike preventive maintenance, which only extends component life, predictive maintenance also reduces unplanned downtime and costs when compared to corrective repair.

The problem of the biased result will be eliminated by the synthetic minority oversampling technique which produces the synthetic data points in the class of minority to the level where the class cannot be dominated. And feeding the such kind of data to the models always eliminate the risk of overfitting.

For the Performance evaluation of the classification algorithms, a parameter called F-score can be used which is the harmonic mean of the precision and recall of the confusion matrix outcomes.

CHAPTER 2
LITERATURE SURVEY

Ainul Yaqin, Majid Rahardi, Ferian Fauzi Abdullah [1] In Accuracy Enhancement of Prediction Method using SMOTE for Early Prediction Student's Graduation in XYZ University presents balancing the data with Synthetic Minority Oversampling Technique and building models such as Artificial Neural Networks (ANN), K-Nearest Neighbor (K-NN) method, and Support Vector Machines (SVM). Based on the results in this paper it was concluded that SMOTE can increase the recall value and test the accuracy of ANN, K-NN, and SVM models with relevant features extracted.

L. J. Muhammad, Ebrahem A. Algehyne, Sani Sharif Usman, Abdulkadir Ahmad, Chinmay Chakraborty, I. A. Mohammed [2] In Supervised Machine Learning Models for Prediction of COVID-19 Infection using Epidemiology Dataset presents Explanation of Classification algorithms such as naive Bayes, logistic regression and decision tree. And the parameters that should be considered for the comparison of algorithms.

Stephen Matzka [3] presents a step-by-step procedure of Predictive Maintenance dataset, Decision tree algorithm, Model agnostic method. The important aspects of this paper are Identification of parameters, Exploratory Interface (EDA) and Approach to Classification Machine Learning Model. Main advantages (or) pros are Explanation provided by the decision trees tend to be higher quality. It was also observed that in some cases of decision tree approach no explanation was provided.

Yongyi Ran Xin Zhou , Pengfeng Lin , Yonggang Wen , Ruilong Deng [4] research consists of some of the key findings such as Fault Diagnosis, Categories of Maintenance Techniques, Fault Prognosis, Machine Learning approaches. The processes involved in this are Deep Learning, Categories of Maintenance Techniques, Predictive Maintenance, Classification Machine Learning Models. Some of the pros and cons related to this are detects important features without human supervision, easy to overfit.

Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, Zhuang Wang [5] Conference consists an overview about Predictive Maintenance. The algorithm used in this process Support Vector Machine. Main advantage is good match between data Modeling and practical

constraints. It’s drawback, is usage of poor model performance due to a smaller number of features.

Susto, Schirru, Pampuri, McLoone, Beghi. [6] used Different Maintenance approaches and overview of them. They used Classification Algorithms, Data Mining. They finalized the report on how different approaches may cause the cost effectiveness. Advantage of the respective approach is that it has Lowest operation cost.

Despite Ribeiro, Marco Tulio, Sameer Singh and Carlos Guestrin [7] broad deployment, ML models are still largely unexplored. However, whether determining whether to deploy a new model or how much faith to place in a forecast, or when deciding whether to act based on a prediction, understanding the motivations behind predictions is crucial. A trustworthy model or forecast may be created using these insights into the model, which can also be utilised to change an unreliable model or prediction. In this work, they provide LIME, an unique explanation approach that, by learning an interpretable model locally around the prediction, explains the predictions of any classifier in an understandable and accurate manner. Moreover, they provide a technique for explaining models that involves framing the job as a submodular optimization issue and giving sample individual forecasts and their explanations in a non-redundant manner.

CHAPTER 3
SOFTWARE & HARDWARE SPECIFICATIONS

3.1 Software requirements

Here the functional requirements are

- Jupyter Notebook
- Python (version 3.7)

Libraries

- pandas
- numpy
- imbleran
- seaborn
- sklearn
- matplotlib

Libraries Used:

- **Sklearn:** Scikit-learn is a free machine learning library for the Python programming language (previously known as sklearn and known as scikits. learn). For classification, predictive analytics, and many other machine learning tasks, it is very widely used throughout the entire bank.
- **Pandas:** Pandas is an open-source Python library that offers high-performance data manipulation. In contrast to the NumPy module, which offers a powerful object called Array, Pandas offers a few sets of robust tools like DataFrame and Series that are primarily used for data analysis.
- **Numpy:** It is described as a Python package used for handling the elements of multidimensional and one-dimensional arrays as well as performing various numerical computations. Numpy array calculations are quicker than standard Python array calculations.
- **Seaborn:** One of the outstanding Python libraries for visualizing graphical statistical plotting is Seaborn. To make the creation of many statistical plots in

Python more visually appealing, Seaborn offers a variety of color palettes and beautiful default styles.

- **Matplotlib:** The **matplotlib.pyplot** is the collection of command-style functions that give working with matplotlib a MATLAB-like experience. The pyplot functions are used to change some aspects of a figure, such as creating a figure, adding a plotting area, drawing some lines in the plotting area, labelling the plot, etc.

Library methods used

- Read_csv()- Read a comma-separated values (csv) file into DataFrame. Also supports optionally iterating or breaking of the file into chunks. Additional help can be found in the online docs for IO Tools.
- head() – It returns first five columns by default.
- tail() – It returns last five columns by default.
- size() – It returns size of the dataset.
- shape – It returns shape of the dataset.
- isnull().sum() – It returns count of null values in each column.
- skew() – It returns skewness value of each column.
- corr() – It returns correlation of each column with one another.
- columns – It returns column names.
- dtypes – It returns datatypes of each column.
- duplicated().sum() – It returns count of duplicated values in each column.
- sklearn.datasets.make_classification()- Generate a random n-class classification problem.This initially creates clusters of points normally distributed (std=1) about vertices of an n_informative-dimensional hypercube with sides of length 2*class_sep and assigns an equal number of clusters to each class. It introduces interdependence between these features and adds various types of further noise to the data.
- Randomoversampler()-RandomOverSampler(sampling_strategy='minority') This means that if the majority class had 1,000 examples and the minority class had 100, this strategy would oversampling the minority class so that it has 1,000 examples.

- train_test_split()-The train_test_split() method is used to split our data into train and test sets. First, we need to divide our data into features (X) and labels (y). The dataframe gets divided into X_train,X_test , y_train and y_test. X_train and y_train sets are used for training and fitting the model.
- KNeighborsClassifier(n_neighbors=3)- We can split the dataset using the sklearn function train_test_split() . The function will return two FDataGrid 's, X_train and X_test with the corresponding partitions, and arrays with their class labels. We will fit the classifier KNeighborsClassifier with the training partition.

CHAPTER 4
PROPOSED SYSTEM DESIGN

4.1 Proposed Methods

In the existing methodology we used the KNN and Random sampling method for the system, but in this approach, we used SMOTE sampling method and various supervised classification algorithms such as Support Vector Machine, Decision Tree, Random Forest, Naïve Bayes, XGBoost and Logistic Regression.

4.2 System Architecture

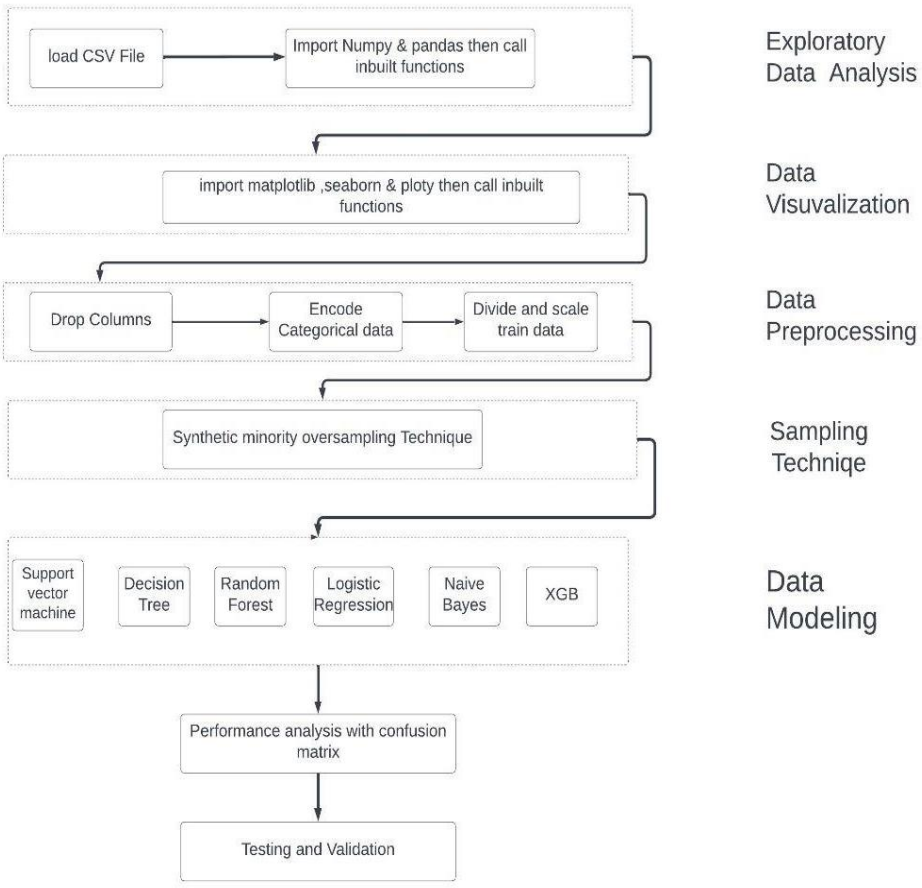

Figure. 4.1 System Arctitecture

Input: Dataset AI4I Predictive Maintenance

Tool: Jupyter - Python 3.7

Step 1: Data Pre-Processing Numerous processes are carried out in this Phase, including

- The procedure for removing null data provides 10,000 data points.
- The method of filtering is carried out by:
    - The values of the data points are verified using duplicated() call for finding duplicate records.
    - By implementing describe() call some of the key attributes can be known such as mean, std, min, 25% 50% 75 %, and max.

- The data points are subjected to the scaling process using MinMax Scaler(MMS) like the formula: $MMS = (X - Xmin)/(Xmin - Xmax)$

Step 2: Analysis of Correlation The degree of correlation between Airtemp, Processtemp, Rotational speed, Torque, Tool wear, and Machine failure was analyzed using correlation analysis.

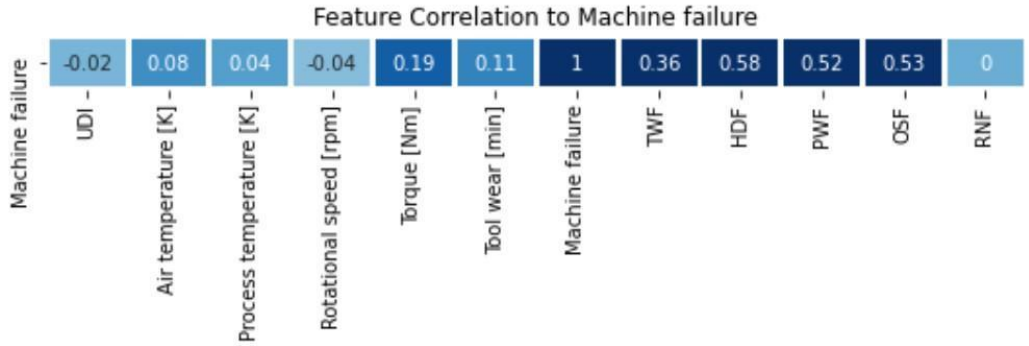

Fig. 4.2 Feature Correlation

Step 3: Features Selection This stage involves determining the features that will be implemented into the prediction model.

Step 4: Balancing using SMOTE and Split Data Train and Testing. Data distribution in this step is split into training data (80%) and testing data (20%). The training data is then balanced using SMOTE and stored in additional variables.

Step 5: Training Prediction Model. The prediction model was constructed in this phase using KNN, SVM, Decision Tree, Random Forest, Logistic Regression, and XGBoost trained using both Imbalanced data and SMOTE data.

Step 6: Evaluation. In this phase, 12 models that were tested using up to 20% of the dataset (during the model training phase) are evaluated for accuracy, recall, precision, and FMeasure.

4.3 Use case Diagram

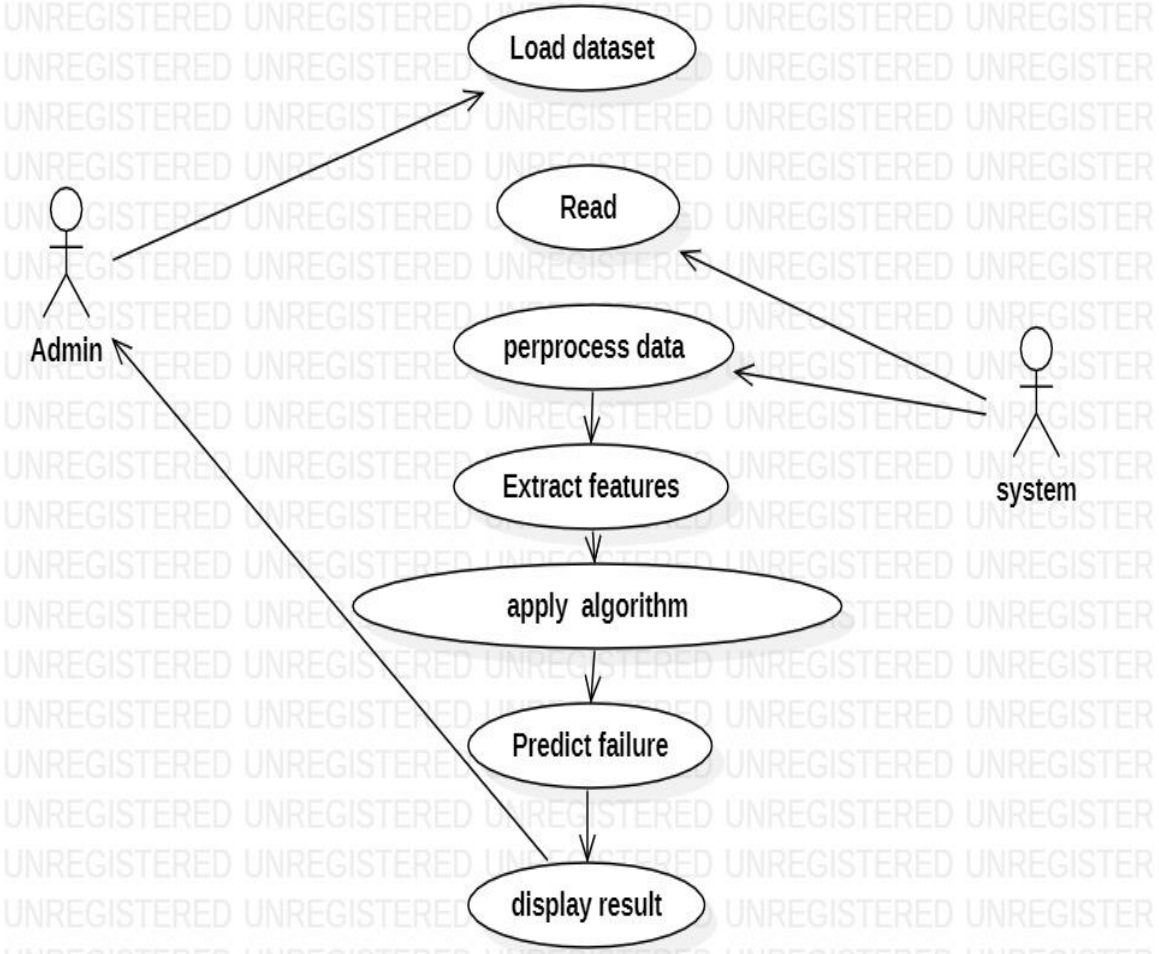

Fig. 4.3 Use Case Diagram

Here the actors are the: System and the Administrator

System has the access to the Read and Processing the Data and the Admin have the access to load the data for the project and get the end results of the prediction.

4.4 Activity Diagram

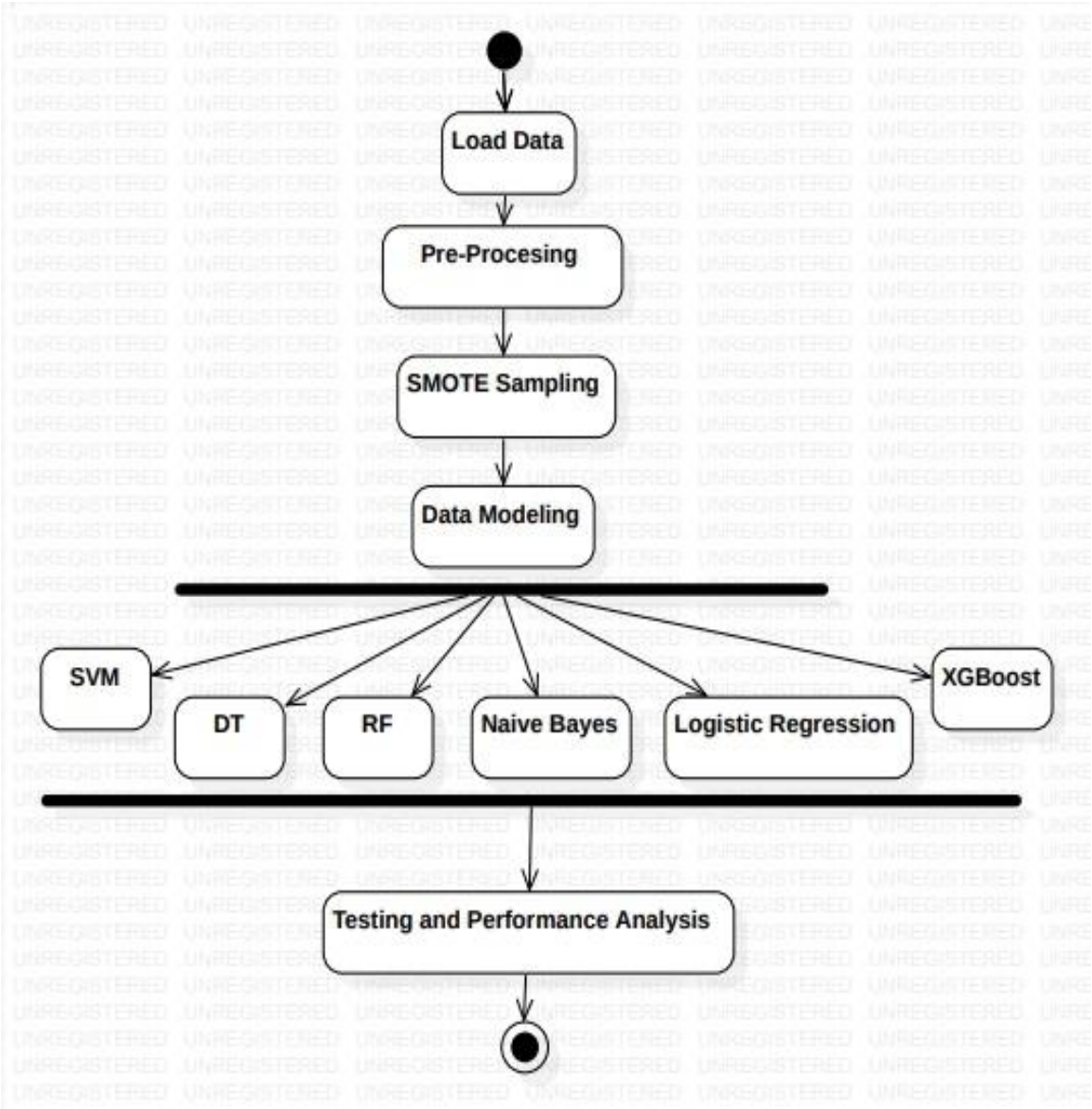

Fig 4.4 Activity Diagram

Firstly, we start the process by the loading the data set into the system and the pre process the data into usable and non-noise format the accurate and efficient results for our task to achieve. Then we sample the data to change its imbalanced nature to avoid the erroneous results such as overfitting the data models.

Here we can see, there are several supervised classification models to be used to modelling. We train them with the both raw and the resampled data to evaluate the difference between

the before and after the SMOTE sampling. Hence, we can finally select the suitable model for the prediction by testing and evaluating the best performing figures.

4.5 TECHNOLOGY DESCRIPTION

A. SMOTE (Synthetic Minority Oversampling Technique)

Under sampling, oversampling, and hybrid methods are a few strategies for handling data imbalances. To solve the issue, the oversampling technique is frequently used to create new synthetic sample data. In contrast, the under-sampling technique takes the approach of removing data from the class that makes up the majority in order to balance the distribution of classes. Presently, the Synthetic Minority Oversampling Technique (SMOTE) is the method for addressing the uneven distribution of unbalanced data learning.

The Synthetic Minority Oversampling Technique (SMOTE) is a method of oversampling that makes use of additional synthetic data. The original data obtained through SMOTE are used to create new minority data that are distinct from the original ones, reducing the negative effects of overfitting on the minority class.

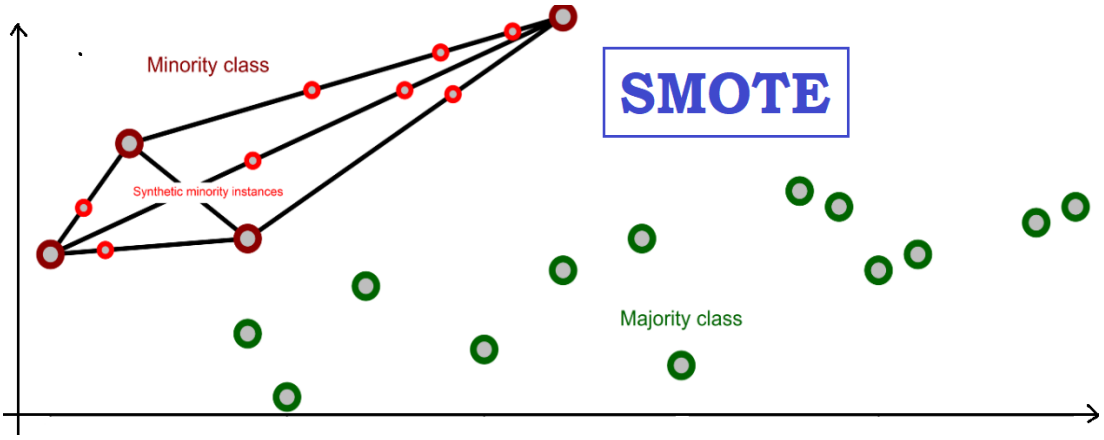

Fig 4.5 Graphical Representation of the SMOTE process.

As shown in fig 4.3 SMOTE assumes that a synthetic data sample can be interpolated between an original and one of the closest neighbours and is based on the concept of the

nearest neighbor algorithm (kNN). The SMOTE algorithm determines the neighbourhood of each data sample from the minority class, chooses one of its neighbours at random, and creates synthetic data by interpolating data between each sample and the chosen nearest neighbour.

The algorithm is randomly chosen, and an original data sample is used to create synthetic data samples when the desired number of synthetic data samples is less than the desired size of the original dataset. Conversely, when the number of synthetic data samples to be made is greater than the size of the original dataset, the algorithm iteratively creates synthetic data samples using a predetermined oversampling ratio.

$$\boldsymbol{s}^i := \boldsymbol{x} + U_i(0,1) \cdot (\hat{\boldsymbol{x}}^{r(i)} - \boldsymbol{x}), \quad i = 1, \ldots, q, \qquad (1)$$

Here, where the created instance is placed on a line between the minority instance x and its random neighbor from the k-neighborhood xˆr(i), the exact placement being defined by a random number generated using a uniform distribution Ui (0,1). The complete SMOTE algorithm is outlined in Equation No.1

Require: Minority set: $\mathcal{M}$; amount of synthetic data: q; number of nearest neighbours: k;
1: $\mathcal{S} = \emptyset$ {Synthetic data set}
2: **for all** $\mathbf{x} \in \mathcal{M}$ **do**
3: determine the k-neighbourhood $\mathcal{N}_k(\mathbf{x})$ of $\mathbf{x}$
{Synthetic instances creation}
4: **for** $i = 1 : q$ **do**
5: randomly select $\hat{\mathbf{x}}^r \in \mathcal{N}_k(\mathbf{x})$
6: $\mathbf{s}^i := \mathbf{x} + U_i(0,1) \cdot (\hat{\mathbf{x}}^{r(i)} - \mathbf{x})$
7: $\mathcal{S} = \mathcal{S} \cup \mathbf{s}^i$
8: **end for**
9: **end for**
10: **return** $\mathcal{S}$

Fig. 4.6 Outline of SMOTE

$$\boldsymbol{s}^i := \boldsymbol{x} + \boldsymbol{U}_i(0,1) \odot (\hat{\boldsymbol{x}}^{r(i)} - \boldsymbol{x}), \quad i = 1, \ldots, q, \quad (2)$$

Where Ui (0,1) is an n-dimensional vector of uniform random variables in (0,1), and ⊙ denotes the Hadamard product. In the first case, new data is created on a line segment connecting the two minority instances. In the second case, new data is created within the bounding rectangle described by the two minority instances.

B. Machine Learning algorithms

[1] Decision Tree
[2] Random Forest
[3] Support Vector Machine
[4] Naïve bayes
[5] Logistic Regression
[6] XGBoost

Above mentioned machine learning approaches are the well suited for the binary classification problems and here has the binary classification problem to resolve. Our dataset is quite biased i.e., imbalanced to begin with, we classify the problem in both raw and sampled version of the data. Hence, we can conclude which can perform well in implementation of the imbalanced natured problems.

1. Decision Tree

Supervised learning algorithm called a decision tree is utilised for both classification and regression tasks. Based on a variety of input features, it constructs a model resembling a tree of decisions and potential outcomes. In order to minimise the impurity or entropy of the resulting subsets, the tree is constructed by recursively dividing the data into subsets depending on the most crucial attributes.

2. Random Forest

A supervised learning technique called random forest is used for classification, regression, and other applications. It operates by building a collection of decision trees, each of which is based on a random selection of the input data and output features. To create a final prediction, the trees are then combined, either by majority vote (for classification) or by averaging. (For regression). The model's generalisation performance is enhanced by the

random sampling of the data and features, which also helps to reduce overfitting. As a method for handling complicated and noisy data, random forest is renowned for its excellent accuracy and robustness.

4. Support Vector Machine

A supervised learning algorithm used for classification and regression applications is the support vector machine (SVM). In order to maximise the margin or distance between the hyperplane and the nearest data points, it finds the hyperplane that best divides the data into various groups.

5. Logistic Regression

A supervised learning approach known as logistic regression is used to model the likelihood of a binary outcome based on one or more independent factors. It converts the result of a linear regression model into a probability value between 0 and 1 using the logistic function.

6. XGBoost

An improved version of the gradient boosting technique that is applied to supervised learning problems like ranking, regression, and classification. It operates by adding decision trees to a model repeatedly, with each tree aiming to fix the flaws of the preceding one. High performance, scalability, and the capacity to manage a variety of data formats are all hallmarks of XGBoost.

CHAPTER 5
IMPLIMENTATION AND TESTING

5.1 IMPLIMENTATION

A. Exploratory Data Analysis

Every research process should include exploratory data analysis or EDA. As the main means of directing the testing of your hypothesis during the exploratory analysis, look for distributions, outliers, and abnormalities in the data. Furthermore, viewing and comprehending data, which is typically done through graphical representation, provides tools for developing hypotheses. The purpose of EDA is to support the analyst's innate capacity to recognize patterns.

```
In [11]: df.describe()
```

Out[11]:

UDI	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure	TWF	HDF	PWF	OSF	RNF
000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000
000	300.004930	310.005560	1538.776100	39.986910	107.951000	0.033900	0.004600	0.011500	0.009500	0.009800	0.00190
568	2.000259	1.483734	179.284096	9.968934	63.654147	0.180981	0.067671	0.106625	0.097009	0.098514	0.04355
000	295.300000	305.700000	1168.000000	3.800000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
000	298.300000	308.800000	1423.000000	33.200000	53.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
000	300.100000	310.100000	1503.000000	40.100000	108.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
000	301.500000	311.100000	1612.000000	46.800000	162.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
000	304.500000	313.800000	2886.000000	76.600000	253.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

Figure. 5.1 Description of ai4i 2020 dataset

Figure 5.1 describe that function has been used to get the statistical information of all numerical columns in the Dataset. Describe function gives statistical information like count, mean, standard deviation, the minimum value of the column, quartile 1(25 percent of the data), quartile 2(50 percent of data or median), quartile 3(75 percent of the data) and maximum value of the column.

```
In [10]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   UDI                      10000 non-null  int64
 1   Product ID               10000 non-null  object
 2   Type                     10000 non-null  object
 3   Air temperature [K]      10000 non-null  float64
 4   Process temperature [K]  10000 non-null  float64
 5   Rotational speed [rpm]   10000 non-null  int64
 6   Torque [Nm]              10000 non-null  float64
 7   Tool wear [min]          10000 non-null  int64
 8   Machine failure          10000 non-null  int64
 9   TWF                      10000 non-null  int64
 10  HDF                      10000 non-null  int64
 11  PWF                      10000 non-null  int64
 12  OSF                      10000 non-null  int64
 13  RNF                      10000 non-null  int64
dtypes: float64(3), int64(9), object(2)
memory usage: 1.1+ MB
```

Figure. 5.2 Information of ai4i 2020 dataset

In Figure 5.2 The info function has been used to get the column information of the dataset. It gives information about the data type of the column, count of the non-null values, range index, memory usage, number of columns, and count of all data types of the columns.

B. Data Visualization

Data visualization involves putting information in a visual context, like a map or graph, so that the human brain can better understand it and draw conclusions from it. Data visualization's primary objective is to make it simple to identify patterns, trends, and outliers in huge datasets. This term is frequently used synonymously with others, including statistical graphics, information graphics, and information visualization. Before data can be visualized in order to draw a conclusion, it must first be collected, processed, and modeled as one of the steps in the data science process. A variety of data presentation architectures (DPAs) that aim to locate, search, manipulate, format, and deliver data as quickly as possible also include data visualization as a component.

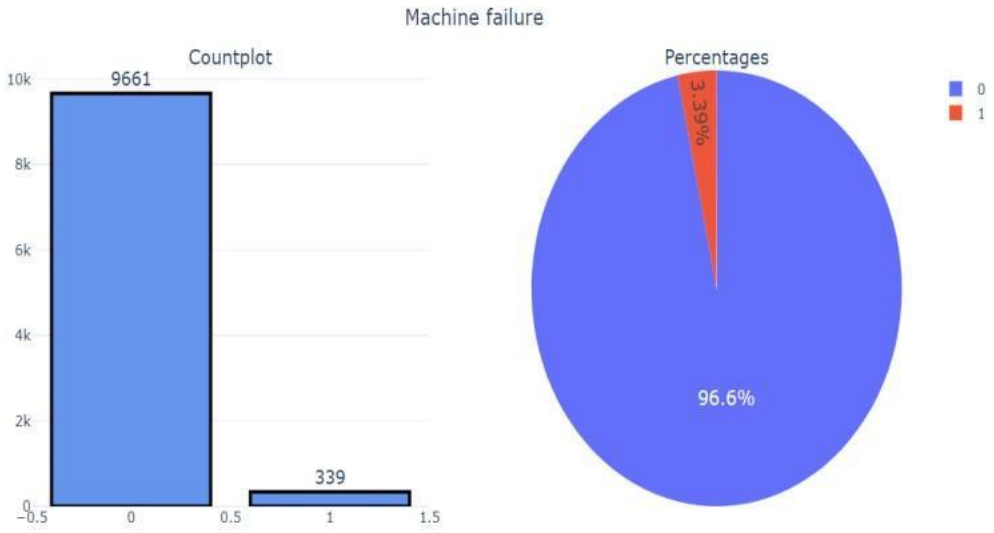

Figure. 5.3 Count plot and pie chart for machine failure

In Figure 5.3 visualization of machine failure (target variable) is done using the bar plot and pie plot. The observation from the above plot is that our dataset has more data on machines that do not fail than the machines which fail.

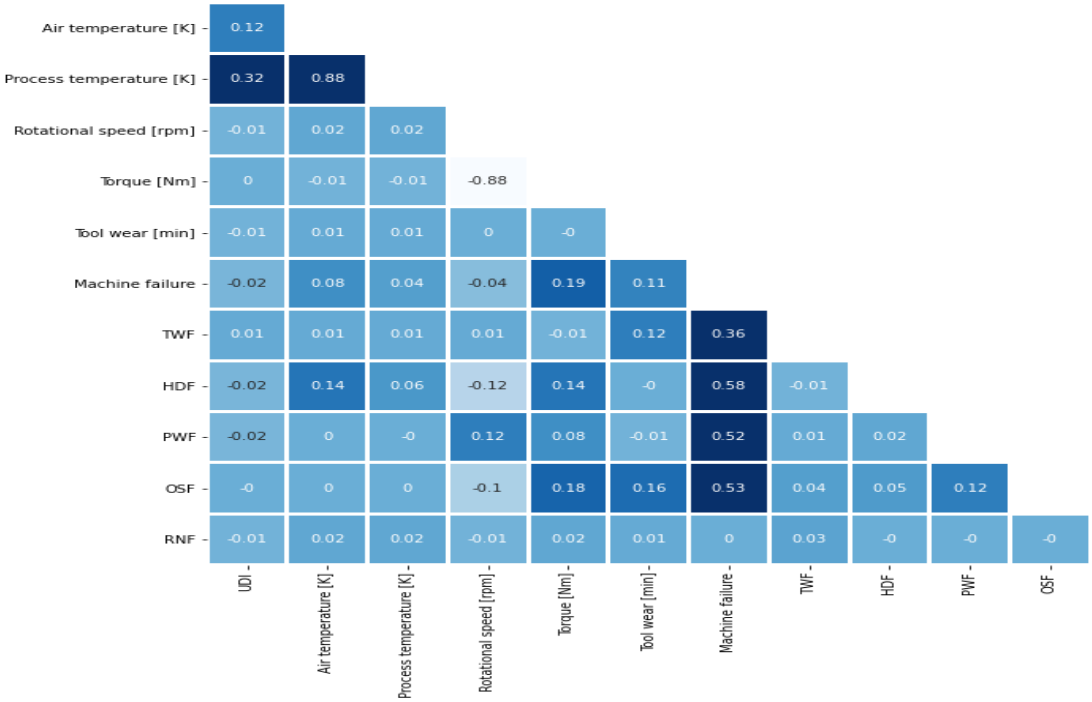

Figure. 5.4 Heatmap / Correlation matrix between all features

In Figure 5.4 shows the Analysis of Correlation. The degree of correlation between Airtemp, Processtemp, Rotational speed, Torque, Tool wear, and Machine failure was analysed using correlation analysis.

C. Data Pre-Processing

Raw data must be processed in order to be used with a machine-learning model. It's the initial and most crucial step in creating a machine-learning model. We don't always come across clean, organized data when working on a machine learning project. Additionally, data must be formatted and cleaned before being used in any way.

```
In [41]: def LABEL_ENCODING(c1):
             # label_encoder object knows how to understand word labels.
             label_encoder = preprocessing.LabelEncoder()
             # Encode labels in column 'species'.
             df[c1]= label_encoder.fit_transform(df[c1])
             df[c1].unique()
             return df

In [42]: LABEL_ENCODING('Type')
```

Out[42]:

	Product ID	Type	Airtemp	Processtemp	Rotationalspeed	Torque	Toolwear	Machine failure	TWF	HDF	PWF	OSF	RNF
0	M14860	2	298.1	308.6	1551	42.8	0	0	0	0	0	0	0
1	L47181	1	298.2	308.7	1408	46.3	3	0	0	0	0	0	0
2	L47182	1	298.1	308.5	1498	49.4	5	0	0	0	0	0	0
3	L47183	1	298.2	308.6	1433	39.5	7	0	0	0	0	0	0
4	L47184	1	298.2	308.7	1408	40.0	9	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	M24855	2	298.8	308.4	1604	29.5	14	0	0	0	0	0	0
9996	H39410	0	298.9	308.4	1632	31.8	17	0	0	0	0	0	0
9997	M24857	2	299.0	308.6	1645	33.4	22	0	0	0	0	0	0

Figure 5.5 Data frame after label encoding

Figure 5.5 Shows Label Encoding which is a popular encoding technique for handling categorical variables. Each label is assigned a unique integer based on alphabetical ordering in this technique. In our project label encoding is used to convert all categorical columns into numerical columns.

D. Experimental Results

Confusion matrix: A classification model's performance on a set of test data for which the real values are known is described in a table called a confusion matrix. Five different metrics measuring the reliability of our model can be computed using the confusion matrix.

- Accuracy (all correct / all) = TP + TN / TP + TN + FP + FN
- Misclassification (all incorrect / all) = FP + FN / TP + TN + FP + FN
- Precision (true positives / predicted positives) = TP / TP + FP
- Sensitivity aka Recall (true positives / all actual positives) = TP / TP + FN
- Specificity (true negatives / all actual negatives) =TN / TN + FP

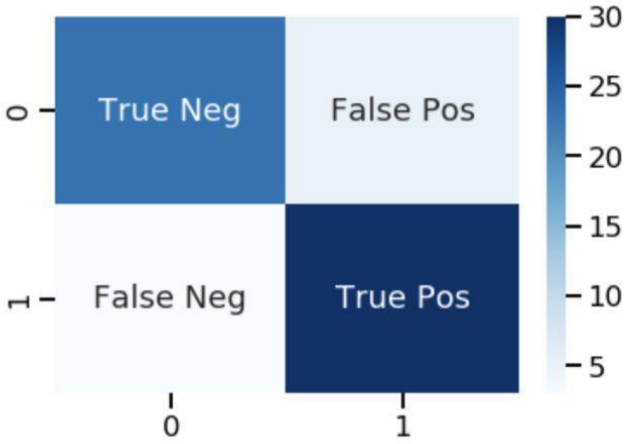

Figure 5.6 Attributes of heat map.

Figure 5.6 is a clear description of the confusion matrix that is used to show the result of the various models.

True positives (TP): These are cases in which we predicted yes (they have the disease), and they do have the disease.

True negatives (TN): We predicted no, and they don't have the disease.

False positives (FP): We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")

False negatives (FN): We predicted no, but they do have the disease. (Also known as a "Type II error.")

5.2 RESULTS

ML ALGORITHMS BEFORE AND AFTER THE SMOTE SAMPLING

1. Random Forest

Results without the SMOTE:

```
Test Result:
================================================
Accuracy Score: 98.40%

_________________________________________________
Classification Report:
                     0           1  accuracy    macro avg  weighted avg
precision     0.987053    0.846154     0.984     0.916603      0.982685
recall        0.996560    0.591398     0.984     0.793979      0.984000
f1-score      0.991784    0.696203     0.984     0.843993      0.982621
support    2907.000000   93.000000     0.984  3000.000000   3000.000000

_________________________________________________
Confusion Matrix:
 [[2897   10]
 [  38   55]]
```

Table No. 5.1 Random Forest Without the SMOTE.

As we can see that the test results were at the accuracy of 98.40% which seems to a best value but it is not because Random Forest is weak learner and it is nearly impossible to achieve such an accuracy rate for an imbalanced nature dataset that to as a raw input of it with out sampling. Hence, we can assume that it is overfitting

Here is the Graphical Representation of the Results:

As we can see in fig 5.7

- False Positive is at 10
- False Negative is at 38

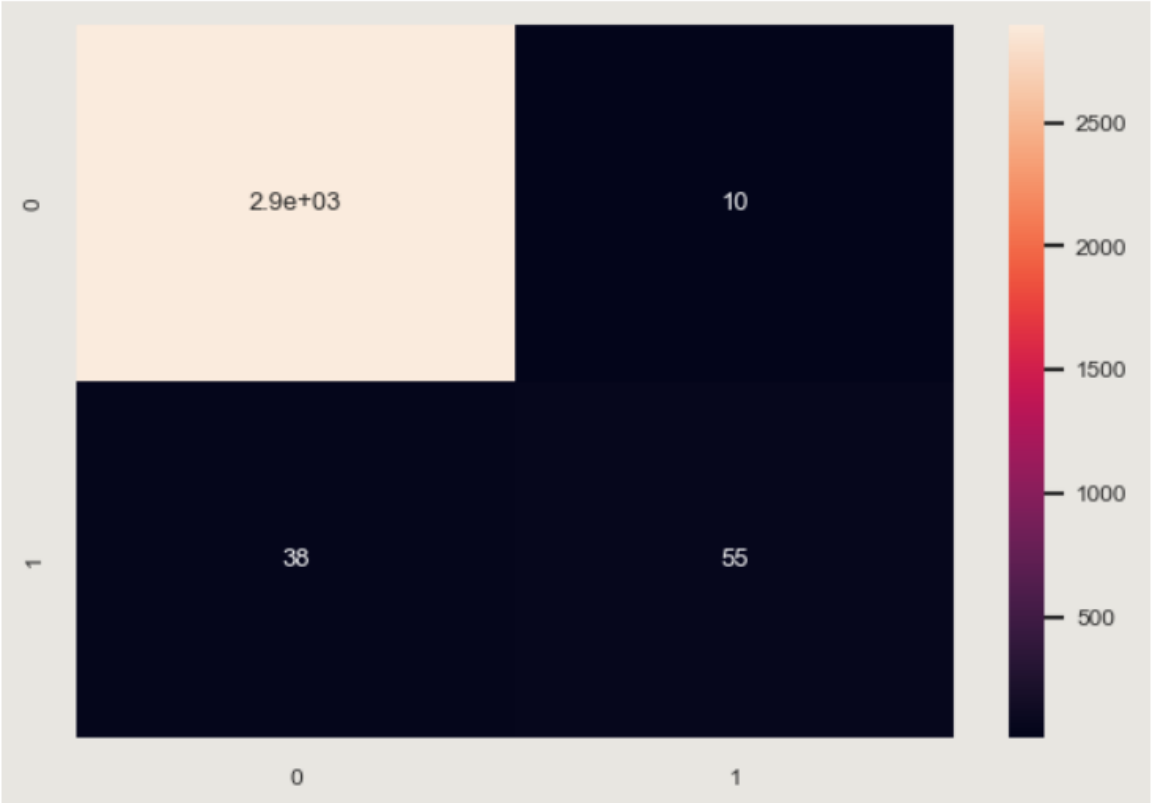

Fig No. 5.7 Random Forest Without SMOTE

Results With the SMOTE:

```
Test Result:
================================================
Accuracy Score: 96.17%

________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.990839   0.450617  0.961667     0.720728      0.973011
recall        0.969321   0.737374  0.961667     0.853347      0.961667
f1-score      0.979962   0.559387  0.961667     0.769674      0.966083
support    2901.000000  99.000000  0.961667  3000.000000   3000.000000

________________________________________________
Confusion Matrix:
 [[2812   89]
 [  26   73]]
```

Table No. 5.2 Random Forest With the SMOTE

When we compare both the table no. 5.1 and 5.2, we can see that there is an decrement in our key performance indicator i.e., Recall (Recall is the better suited performance indicator for the classification of the imbalanced problems [1]) where before the SMOTE it is at 0.591398 for the 1's and after the SMOTE it is at 0.559387. Not a significant fall though but there is slight dissapointment over the previous one.

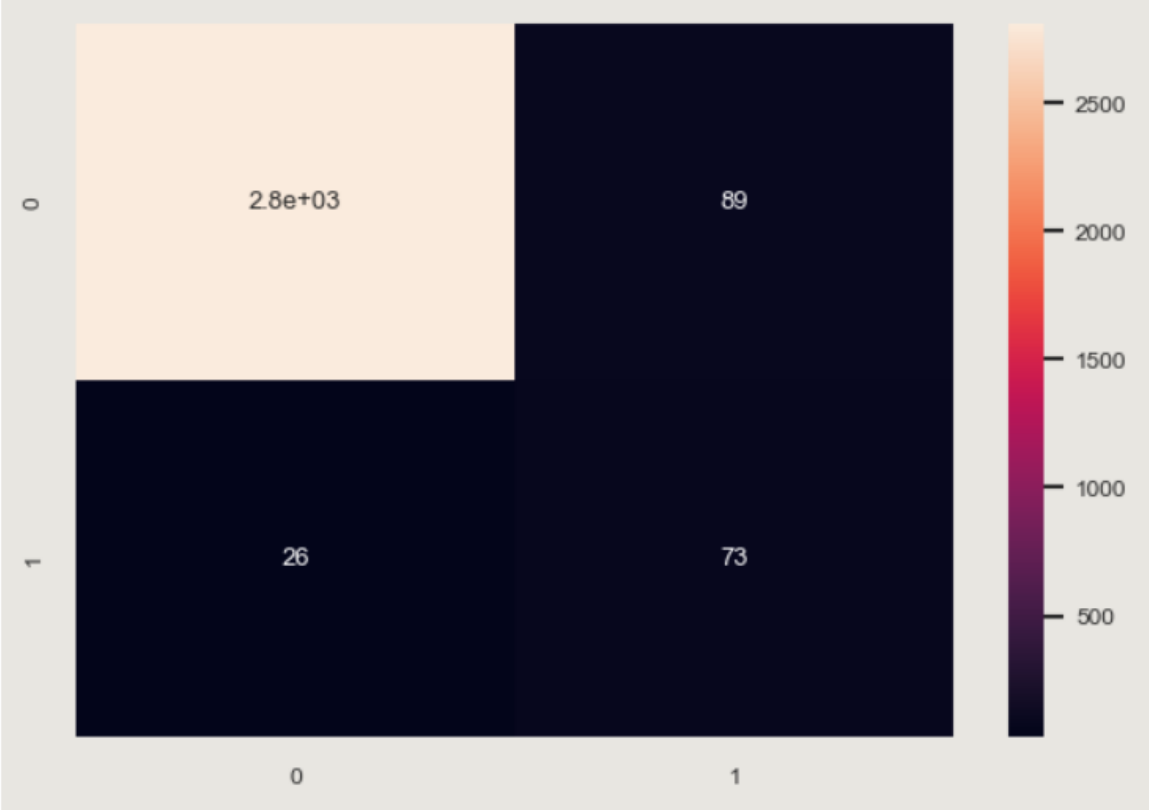

Fig No. 5.8 Random Forest After the SMOTE

Here is the Graphical Representation of the Results:

As we can see in fig 5.8

- False Positive is at 89
- False Negative is at 26

2. Logistic Regression

Results Without the SMOTE:

```
Test Result:
================================================
Accuracy Score: 97.37%

________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.976415   0.718750  0.973667     0.847583      0.968427
recall        0.996904   0.247312  0.973667     0.622108      0.973667
f1-score      0.986553   0.368000  0.973667     0.677277      0.967378
support    2907.000000  93.000000  0.973667  3000.000000   3000.000000

________________________________________________
Confusion Matrix:
 [[2898    9]
 [  70   23]]
```

Table No. 5.3 Logistic Regression Without the SMOTE

As we can see that the test results were at the accuracy of 97.37% which seems to a best value but it is not because Logistic Regression is weak learner and it is nearly impossible to achieve such an accuracy rate for an imbalanced nature dataset that to as a raw input of it without sampling. Hence, we can assume that it is overfitting.

As we can Fig 5.9 in the next page

- False Positive is at 9
- False Negative is at 70

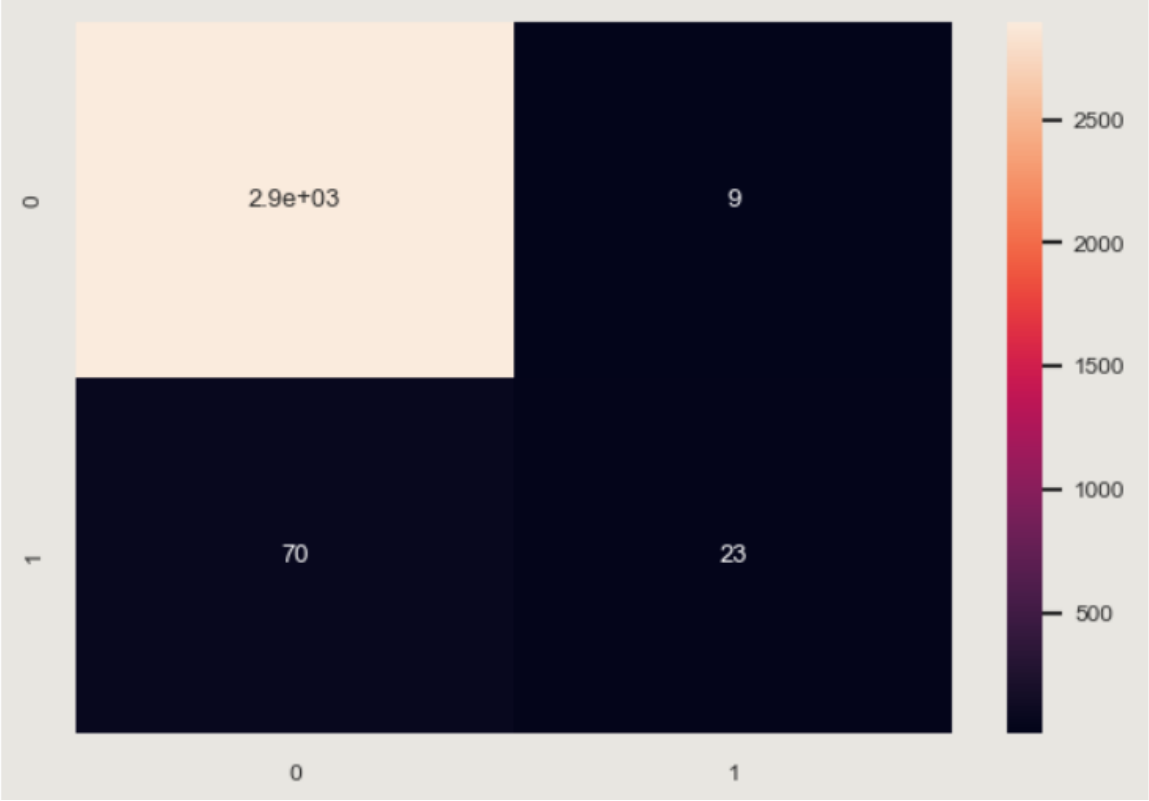

Fig 5.9 LR without the SMOTE

Results With SMOTE:

```
Test Result:
================================================
Accuracy Score: 89.33%

_________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.987902   0.188732  0.893333     0.588317      0.961529
recall        0.900724   0.676768  0.893333     0.788746      0.893333
f1-score      0.942301   0.295154  0.893333     0.618727      0.920945
support    2901.000000  99.000000  0.893333  3000.000000   3000.000000

_________________________________________________
Confusion Matrix:
 [[2613  288]
 [  32   67]]
```

Table No. 5.4 LR With SMOTE

By comparing both 5.3 and the 5.4 table we can see that our Performance Indicator Recall has been improved from 24% to the significant 67% which is quite remarkable compared to the previous algorithm Random Forest. We always known that LR is a good ML approach for the Binary Classifications. Now the accuracy at 89% which is quite good.

Fig 5.10 LR With SMOTE.

As we can Fig 5.10

- False Positive is at 290
- False Negative is at 32

3. Decision Tree

Results without SMOTE:

```
Test Result:
================================================
Accuracy Score: 97.60%

_________________________________________________
Classification Report:
                     0          1  accuracy   macro avg  weighted avg
precision     0.987616   0.612903     0.976     0.80026         0.976
recall        0.987616   0.612903     0.976     0.80026         0.976
f1-score      0.987616   0.612903     0.976     0.80026         0.976
support    2907.000000  93.000000     0.976  3000.00000      3000.000

_________________________________________________
Confusion Matrix:
 [[2871   36]
 [  36   57]]
```

Table No. 5.5 DT Without the SMOTE

As we can see, the test results' accuracy was 97.60%, which would seem to be the best value, but it is not since Decision Tree is a poor learner and it is practically impossible to reach such an accuracy rate for a dataset of imbalanced natural data that is used as a raw input without sampling. This leads us to believe that it is overfitting**.**

As we can Fig 5.11 in the next page

- False Positive is at 36
- False Negative is at 36

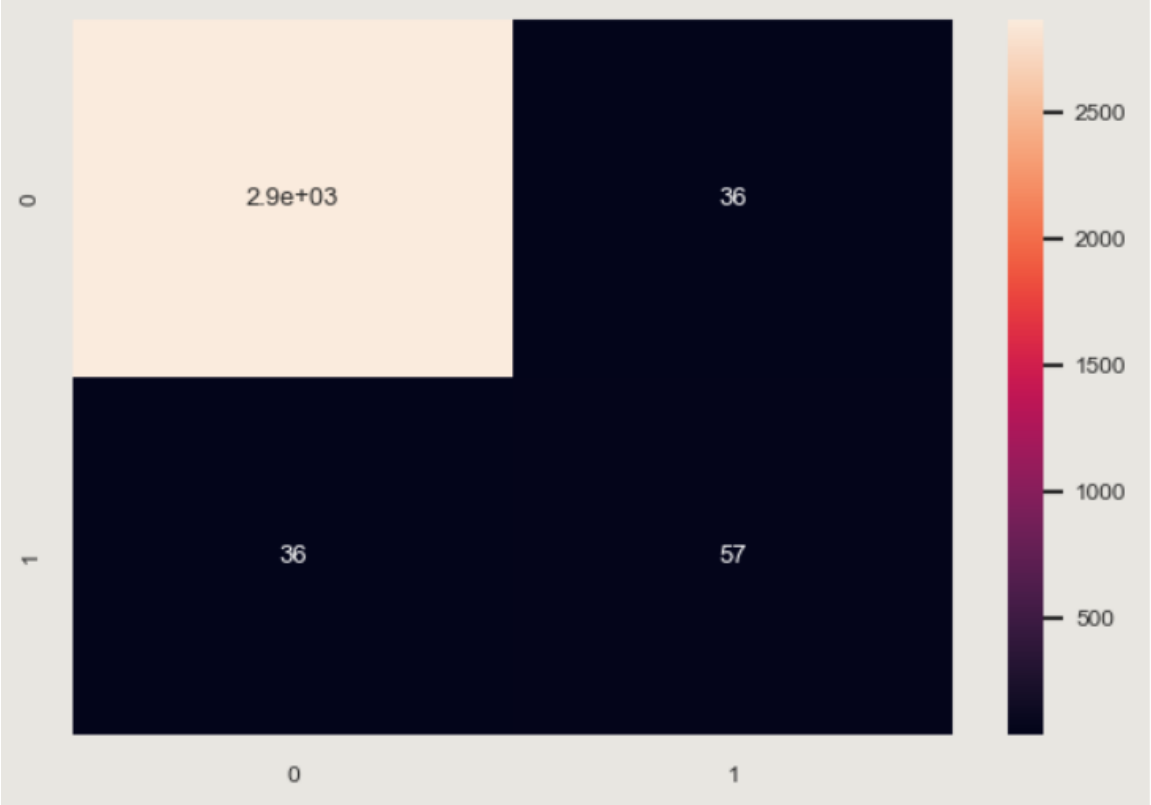

Fig 5.11 DT Without the SMOTE

Results with SMOTE:

```
Test Result:
================================================
Accuracy Score: 95.87%

________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.989425   0.423313  0.958667     0.706369      0.970744
recall        0.967597   0.696970  0.958667     0.832284      0.958667
f1-score      0.978390   0.526718  0.958667     0.752554      0.963485
support    2901.000000  99.000000  0.958667  3000.000000   3000.000000

________________________________________________
Confusion Matrix:
 [[2807   94]
 [  30   69]]
```

Table No. 5.6 DT with SMOTE

When we examine table numbers 5.5 and 5.6, we can see that there has been slight fall in our main performance indicator, recall, which was 61% for the 1s before the SMOTE and is now 52% after the SMOTE. There has been a minor improvement over the prior one, h owever it is not a huge one. But still accuracy is still nearly unchanged.

Here the Visualisation of the Results

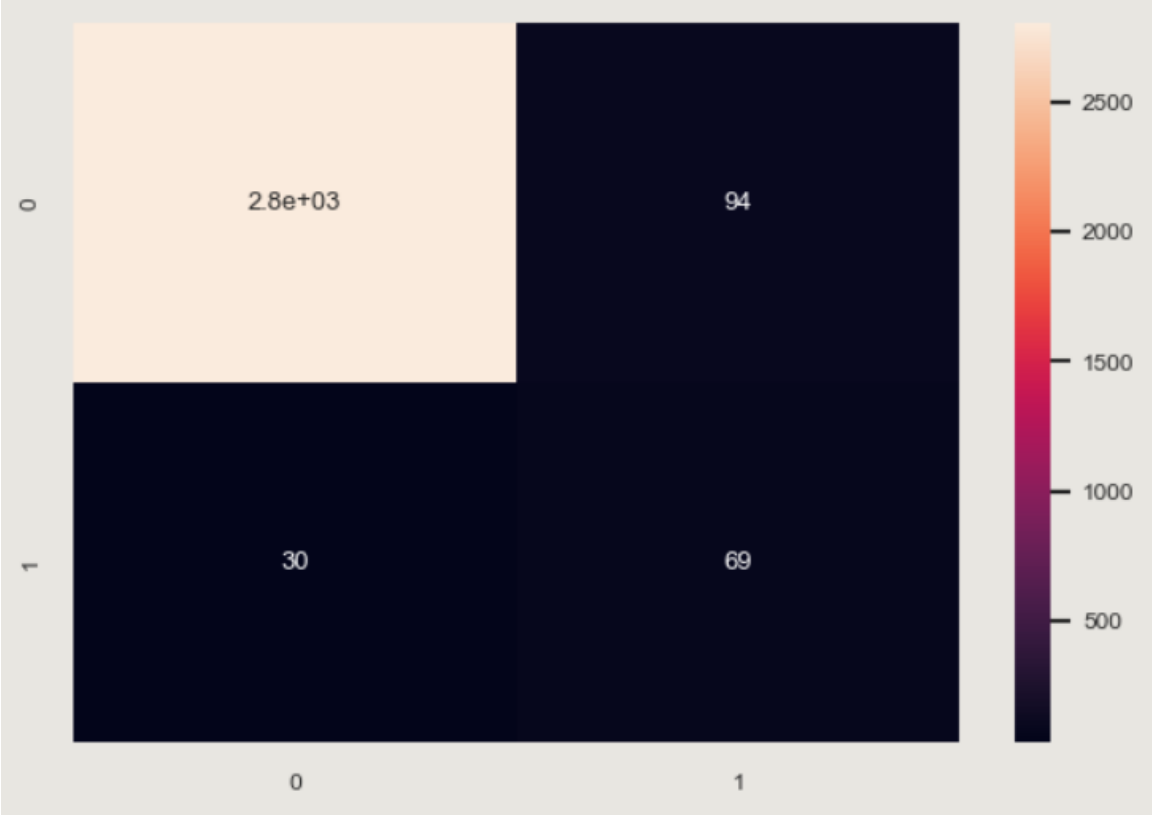

Fig 5.12 DT With SMOTE

As we can Fig 5.12

- False Positive is at 94
- False Negative is at 30

4. SVM

Results Without SMOTE:

```
Test Result:
================================================
Accuracy Score: 97.70%

________________________________________________
Classification Report:
                     0           1  accuracy    macro avg  weighted avg
precision     0.977456    0.928571     0.977     0.953014      0.975941
recall        0.999312    0.279570     0.977     0.639441      0.977000
f1-score      0.988263    0.429752     0.977     0.709008      0.970949
support    2907.000000   93.000000     0.977  3000.000000   3000.000000

________________________________________________
Confusion Matrix:
 [[2905    2]
 [  67   26]]
```

Table No. 5.7 SVM Without SMOTE

From above table we can see that the accuracy is at 97% which is high even for an algorithm like SVM which is good performing algorithm in classification problems. And the recall is at 27% for the 1's of the data which is considerably bad though**.** We must not ignore the fact that we fed the data pre-sampled data. Hence the results were overfitting as it seems.

And the graph below displays the FP and FN values

- False Positive is at 2
- False Negative is at 67

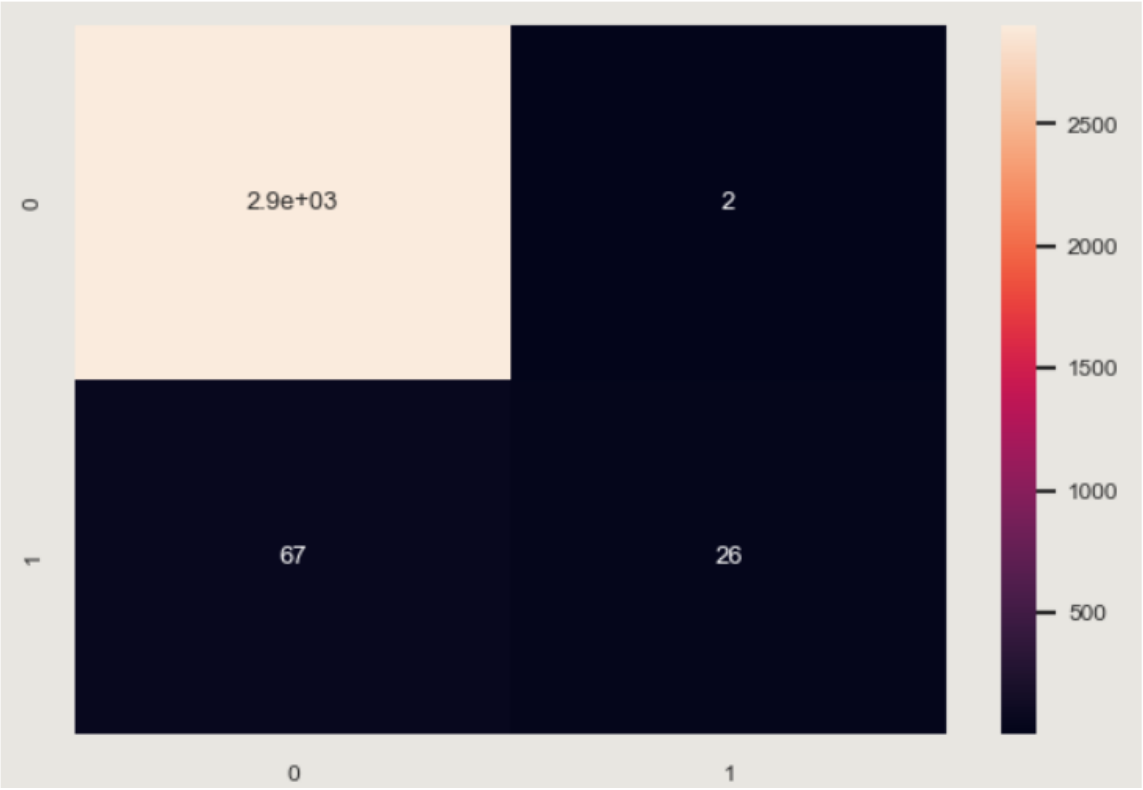

Fig 5.13 SVM Without SMOTE

Results With SMOTE:

```
Test Result:
================================================
Accuracy Score: 86.33%

_________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.989391   0.158242  0.863333     0.573816      0.961963
recall        0.867977   0.727273  0.863333     0.797625      0.863333
f1-score      0.924715   0.259928  0.863333     0.592322      0.902777
support    2901.000000  99.000000  0.863333  3000.000000   3000.000000

_________________________________________________
Confusion Matrix:
 [[2518  383]
 [  27   72]]
```

Table No. 5.8 SVM With SMOTE

Our Performance Indicator Recall has increased from 27% to the noteworthy 72%, which is fairly exceptional when compared to the prior algorithms, with the exception of LR, as seen by comparing the 5.7 and 5.8 tables. SVM is a solid machine learning strategy for binary classifications, as we have always known. Instead of the overfitted 97%, the accuracy is now at 86%, which is respectable.

Fig 5.14 SVM With SMOTE

As we can Fig 5.12

- False Positive is at 380
- False Negative is at 27

5. Naïve Bayes

Results Without SMOTE:

```
Test Result:
================================================
Accuracy Score: 96.30%

________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.974864   0.339286     0.963     0.657075      0.955161
recall        0.987272   0.204301     0.963     0.595787      0.963000
f1-score      0.981029   0.255034     0.963     0.618031      0.958523
support    2907.000000  93.000000     0.963  3000.000000   3000.000000

________________________________________________
Confusion Matrix:
 [[2870   37]
 [  74   19]]
```

Table No. 5.9 Naïve Bayes Without SMOTE

From the above table 5.9 we can conclude that the Naïve bayes is overfitting model without the implementation of SMOTE, here accuracy at 96% which is not suitable for a weak classifier such as Naïve Bayes and the recall value is also hit the bottom as it shown the lowest value up until now 20%.

And as we can Fig 5.12

- False Positive is at 37
- False Negative is at 74

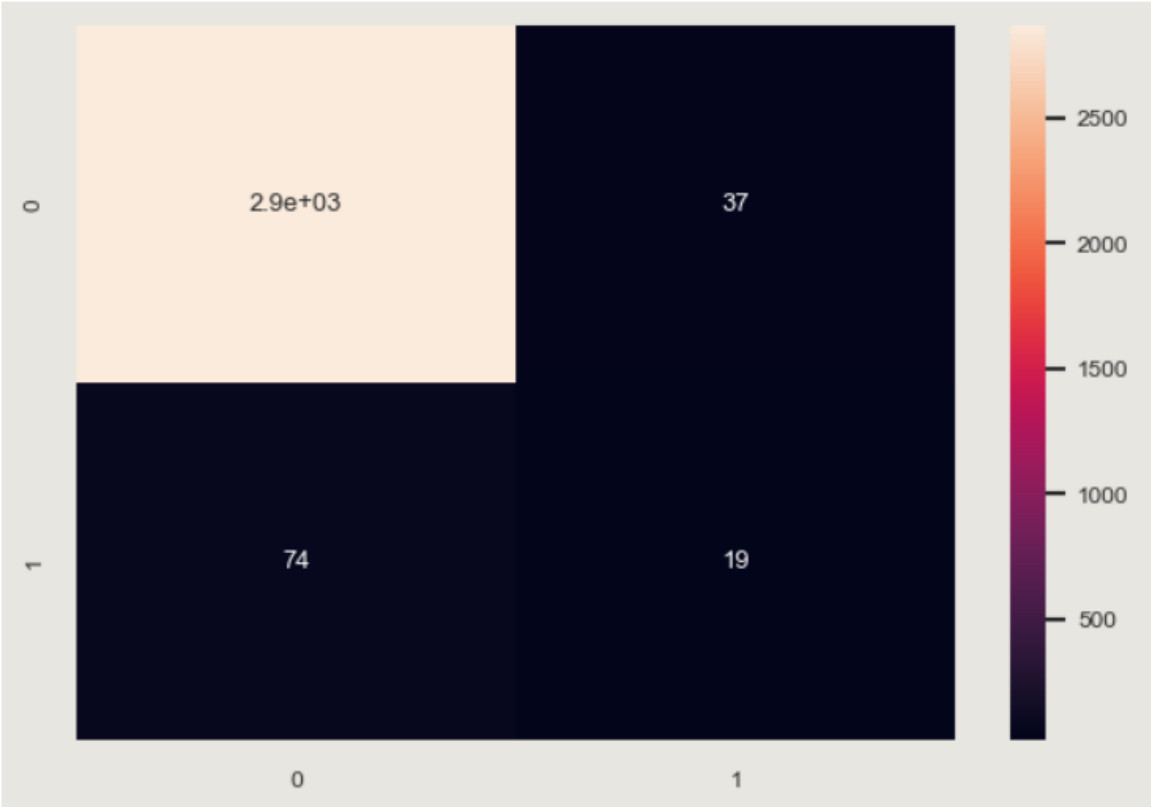

Fig 5.15 Naïve Bayes Without SMOTE

Results With SMOTE:

```
Test Result:
================================================
Accuracy Score: 91.03%

________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.988493   0.222222  0.910333     0.605358      0.963206
recall        0.917959   0.686869  0.910333     0.802414      0.910333
f1-score      0.951921   0.335802  0.910333     0.643862      0.931589
support    2901.000000  99.000000  0.910333  3000.000000   3000.000000

________________________________________________
Confusion Matrix:
 [[2663  238]
 [  31   68]]
```

Table No. 5.10 Naïve Bayes With SMOTE

When compared to some of the earlier algorithms, except for LR and SVM, our Performance Indicator Recall has increased from 20% to the barely 33%, which is rather non exceptional. This can be seen by comparing the 5.9 and 5.10 tables. The accuracy is currently at 91%, which is respectable, as opposed to the overfitted 97%.

And the Picture below represents the Type I and Type II errors

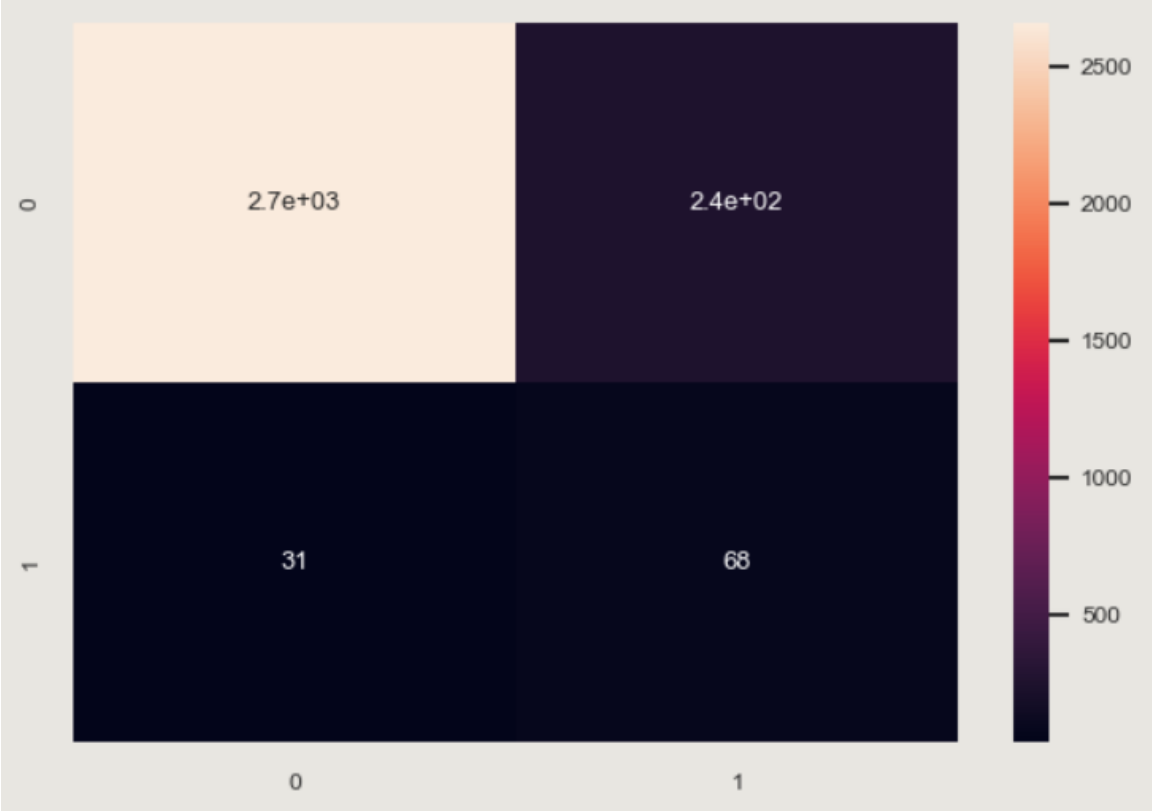

Fig 5.16 Naïve Bayes With SMOTE

- False Positive is at 240
- False Negative is at 31

6. XGBoost

Results Without SMOTE:

```
Test Result:
================================================
Accuracy Score: 97.23%

_________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.972557   0.916667  0.972333     0.944612      0.970824
recall        0.999656   0.118280  0.972333     0.558968      0.972333
f1-score      0.985920   0.209524  0.972333     0.597722      0.961852
support    2907.000000  93.000000  0.972333  3000.000000   3000.000000

_________________________________________________
Confusion Matrix:
 [[2906    1]
 [  82   11]]
```

Table No. 5.11 XGBoost Without SMOTE

We can see from table 5.11 above that the accuracy is at 97%, which is high even for a classification problem-solving method like XGBoost. Additionally, the recall for the data's 1s is at 11%, which is a very poor level. We cannot disregard the fact that we used pre-sampled data to feed the data. As a result, it appears that the results were overfitting.

Here below figure 5.16 is the snap of the results of the confusion matrix and it shows the FP and FN values

- False Positive is at 1
- False Negative is at 82

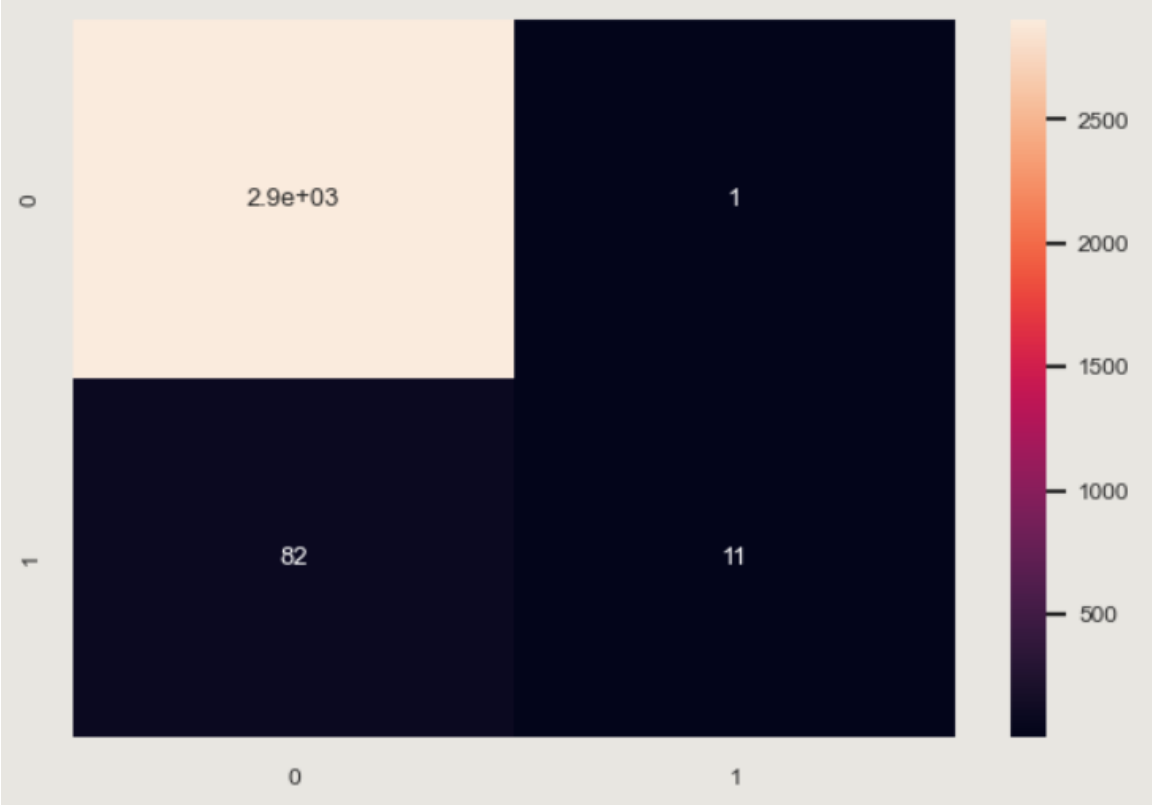

Fig 5.17 XGBoost Without SMOTE

Results With SMOTE:

```
Test Result:
================================================
Accuracy Score: 89.03%

________________________________________________
Classification Report:
                     0          1  accuracy    macro avg  weighted avg
precision     0.980568   0.145062  0.890333     0.562815      0.952996
recall        0.904516   0.474747  0.890333     0.689632      0.890333
f1-score      0.941008   0.222222  0.890333     0.581615      0.917288
support    2901.000000  99.000000  0.890333  3000.000000   3000.000000

________________________________________________
Confusion Matrix:
 [[2624  277]
 [  52   47]]
```

Table No. 5.12 XGBoost With SMOTE

We can say by observing the above table that XGBoost has results and it highly nears to the real-world accomplishment. The Accuracy here is the 89% which is a moderate value and by comparing the tables 5.11 and 5.12, the recall value also jumped from lowly 11% to the barely 22% which is not a remarkable here in this case.

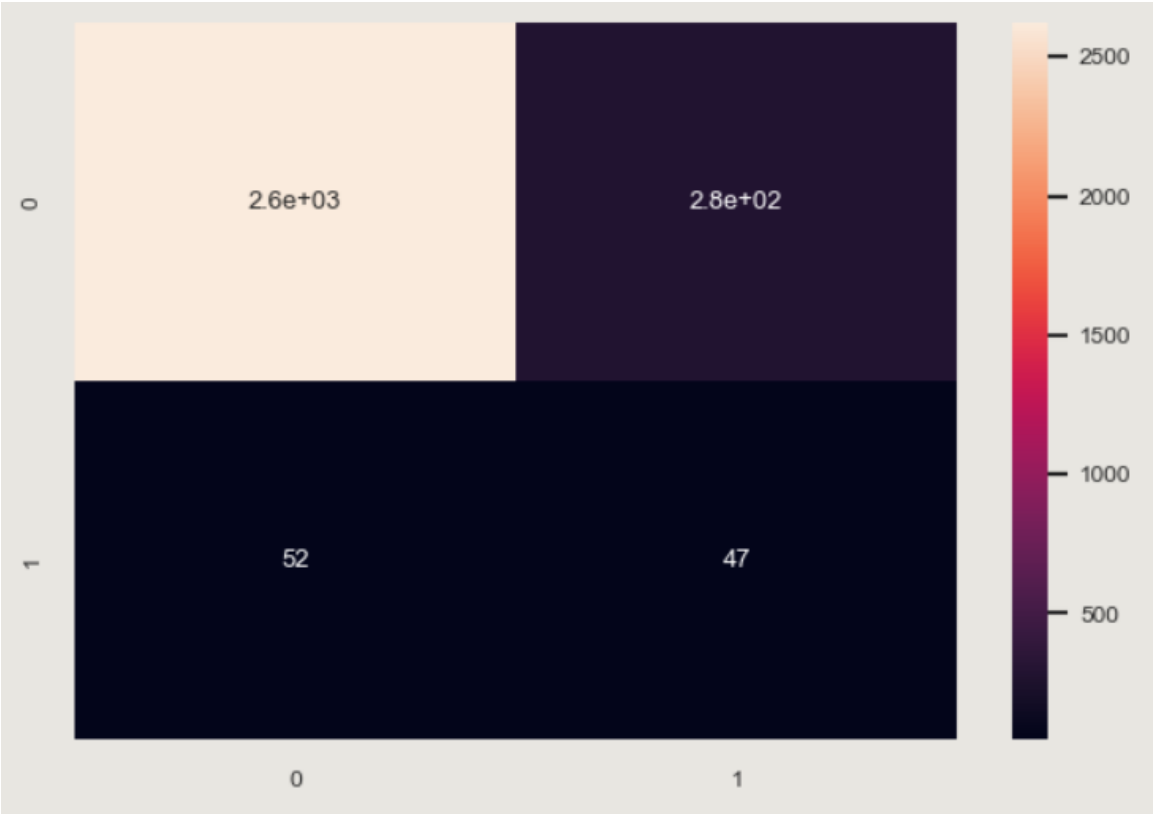

Fig 5.18 XGBoost With SMOTE

The graphical representation above is showing the entities of the confusion matrix

- False Positive is at 280
- False Negative is at 52

CHAPTER 6
CONCLUSION & FUTURE SCOPE

The study can be formulated as follows based on the findings. Almost all classification algorithms, including SVM, XGBoost, DT, Random Forest, Naive Bayes Classifier, and Logistic Regression models, can have their recall values improved by SMOTE. The recall values for SVM had significantly risen from 27.9 to 72.7. The proposed SMOTE approach can be used for imbalanced data classification, according to the results of experimental work done on the chosen data set. Compared to Random Forest and Decision Tree other algorithms such as SVM and LR perform on average better, according to the experimental results that have already been evaluated. It was also utilized in numerous works on imbalanced classification, particularly those that focused on conventional methods. As a result, the comparison of recall and accuracy is the focus of this analysis.

This exact approach can be boosted using the Adaptive Boosting technique which enhances the classifier for the better efficiency and it is quite useful in the problem of the imbalanced natured data sets. And the Performance indicator can be added along with the recall attribute such as MCC and AUC when using a very large amount of the dataset.

REFERENCES

[1] Ainul Yaqin*, Majid Rahardi, Ferian Fauzi Abdulloh. “Accuracy Enhancement of Prediction Method using SMOTE for Early Prediction Student's Graduation in XYZ University ” (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 13, No. 6, 2022.

[2] L. J. Muhammad, Ebrahem A. Algehyne, Sani Sharif Usman, Abdulkadir Ahmad, Chinmay Chakraborty, I. A. Mohammed “Supervised Machine Learning Models for Prediction of COVID-19 Infection using Epidemiology Dataset ” SN Computer Science (2021) 2:11 https://doi.org/10.1007/s42979-020-00394-7.

[3] Stephen Matzka, “Explainable Artificial Intelligence for Predictive Maintenance Applications”, 2020 Third International Conference on Artificial Intelligence for Industries (AI4I).

[4] Yongyi Ran Xin Zhou, Pengfeng Lin , Yonggang Wen , Ruilong Deng, “A Survey of Predictive Maintenance: Systems, Purposes and Approaches”, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, NOV. 2019

[5] Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, Zhuang Wang “Log-based Predictive Maintenance Proceedings of the ACM SIGKDD”, International Conference on Knowledge Discoveryand DataMining, AUG.2020

[6] Susto, Schirru, Pampuri, McLoone, Beghi “Machine Learning for Predictive Maintenance: A Multiple Classifiers Approach”, IEEE Transactions on Industrial Informatics,2019

[7] Kementerian Pendidikan dan Kebudayaan RI, “Peraturan Menteri Standar Nasional Pendidikan Tinggi,” Ban-PT, 2015. [Online]. Available: https://www.banpt.or.id/?page_id=80.

[8] A. Maulana, "Prediction of student graduation accuracy using decision tree with application of genetic algorithms," IOP Conf. Ser. Mater. Sci. Eng., vol. 1073, no. 1, p. 012055, Feb. 2021, DOI: 10.1088/1757- 899X/1073/1/012055.

[9] A. Yaqin, A. D. Laksito, and S. Fatonah, "Evaluation of Backpropagation Neural Network Models for Early Prediction of Student's Graduation in XYZ University," Int. J. Adv. Sci. Eng. Inf. Technol., vol. 11, no. 2, pp. 610–617, 2021, DOI: 10.18517/IJASEIT.11.2.11152.

[10] M. Anwar, "Prediction of the graduation rate of engineering education students using Artificial Neural Network Algorithms," Int. J. Res. Couns. Educ., vol. 5, no. 1, pp. 23–31, Jun. 2021, DOI: 10.24036/00411ZA0002.

[11] M. Wati, W. H. Rahmah, N. Novirasari, Haviluddin, E. Budiman, and Islamiyah, "Analysis K-Means Clustering to Predicting Student Graduation," J. Phys. Conf. Ser., vol. 1844, no. 1, p. 012028, Mar. 2021, DOI: 10.1088/1742-6596/1844/1/012028.

[12] Y. Baashar, G. Alkawsi, N. Ali, H. Alhussian, and H. T. Bahbouh, "Predicting student's performance using machine learning methods: A systematic literature review," 2021 Int. Conf. Comput. Inf. Sci., pp. 357–362, Jul. 2021, DOI: 10.1109/ICCOINS49721.2021.9497185.

[13] D. Gonzalez-Cuautle et al., "Synthetic Minority Oversampling Technique for Optimizing Classification Tasks in Botnet and IntrusionDetection-System Datasets," DOI: 10.3390/app10030794.

[14] D. Agustin, D. Zaenal Abidin, and E. Rasywir, "Kajian Komparasi Teknik Klasifikasi Naive Bayes, K-Nearest Neighbor, Dan Jaringan Saraf Tiruan Pada Prediksi Lama Masa Studi Mahasiswa Jurusan Teknik Informatika (Studi Kasus : Stikom Dinamika Bangsa Jambi)."

[15] R. Muliono, J. H. Lubis, and N. Khairina, "Analysis K-Nearest Neighbor Algorithm for Improving Prediction Student Graduation Time," Sink. J. dan Penelit. Tek. Inform., vol. 4, no. 2, pp. 42–46, Mar. 2020, DOI: 10.33395/SINKRON.V4I2.10480

[16] I. Rodrigues, A. Parayil, T. Shetty, and I. Mirza, "Use of Linear Discriminant Analysis (LDA), K Nearest Neighbours (K-NN), Decision Tree (CART), Random Forest (RF), Gaussian Naive Bayes (NB), Support Vector Machines (SVM) to Predict Admission for Post Graduation Courses," SSRN Electron. J., 2020, DOI: 10.2139/ssrn.3683065.

[17] M. Koziarski, "CSMOUTE: Combined Synthetic Oversampling and Undersampling Technique for Imbalanced Data Classification."

[18] S. Feng et al., "COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction," Inf. Softw. Technol., vol. 129, pp. 950–5849, Jan. 2021, DOI: 10.1016/j.infsof.2020.106432.

[19] F. Kamalov and D. Denisov, "Gamma distribution-based sampling for imbalanced data," Knowledge-Based Syst., vol. 207, p. 106368, Nov. 2020, DOI: 10.1016/j.knosys.2020.106368.

[20] T. Liu, X. Zhu, W. Pedrycz, and Z. Li, "A design of information granule-based under-sampling method in imbalanced data classification," Soft Comput., vol. 24, no. 22, pp. 17333–17347, 2020, DOI: 10.1007/s00500-020-05023-2.

[21] S.-C. Wang, "Artificial Neural Network," in Interdisciplinary Computing in Java Programming, Boston, MA: Springer US, 2003, pp. 81–100.

[22] B. S. Santoso, J. P. Tanjung, U. P. Indonesia, B. Gandum, and A. N. Network, "Classification of Wheat Seeds Using Neural Network Backpropagation," JITE (J. Informatics Telecommun. Eng.), vol. 4, no. January 2021.

[23] O. Ahmed and A. Brifcani, "Gene Expression Classification Based on Deep Learning," 4th Sci. Int. Conf. Najaf, SICN 2019, pp. 145–149, 2019, doi: 10.1109/SICN47020.2019.9019357.

[24] M. A. Mercioni and S. Holban, "Soft-Clipping Swish: A Novel Activation Function for Deep Learning," pp. 225–230, Jun. 2021, DOI: 10.1109/SACI51354.2021.9465622.

[25] R. Harikumar and K. P. Sunil, K-NN Classifier and K-Means Clustering for Robust Classification of Epilepsy ... - Harikumar Rajaguru, Sunil Kumar Prabhakar - Google Books. Hamburg: Anchor Academic Publishing, 2017.

[26] L. Cui, Y. Zhang, R. Zhang, and Q. H. Liu, "A Modified Efficient K-NN Method for Antenna Optimization and Design," IEEE Trans. Antennas Propag., vol. 68, no. 10, pp. 6858–6866, 2020, DOI: 10.1109/TAP.2020.3001743.

[27] J. Smith, Support Vector Machine: Examples With Matlab. CreateSpace Independent Publishing Platform, 2017.

[28] D. J. Armaghani, P. G. Asteris, B. Askarian, M. Hasanipanah, R. Tarinejad, and V. Van Huynh, "Examining hybrid and single SVM models with different kernels to predict rock brittleness," Sustain., vol. 12, no. 6, pp. 1–17, 2020, DOI: 10.3390/su12062229

[29] M. N. Jebur, B. Pradhan, and M. S. Tehrany, "Optimization of landslide conditioning factors using very high-resolution airborne laser scanning (LiDAR) data at catchment scale," Remote Sens. Environ., vol. 152, pp. 150–165, 2014, DOI: 10.1016/j.rse.2014.05.013.

[30] J. Xu, Y. Zhang, and D. Miao, "Three-way confusion matrix for classification: A measure driven view," Inf. Sci. (Ny)., vol. 507, pp. 772–794, Jan. 2020, DOI: 10.1016/J.INS.2019.06.064.

[31] I. Diakonikolas, D. M. Kane, V. Kontonis, and N. Zarifis, "Algorithms and SQ Lower Bounds for PAC Learning One-Hidden-Layer ReLU Networks," Proceedings of Machine Learning Research, vol. 125. PMLR, pp. 1514–1539, 15-Jul-2020.

[32] B. Yekkehkhany, A. Safari, S. Homayouni, and M. Hasanlou, "A comparison study of different kernel functions for SVM-based classification of multi-temporal polarimetry SAR data," Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch., vol. 40, no. 2W3, pp. 281–285, 2014, DOI: 10.5194/ISPRSARCHIVES-XL-2-W3- 281-2014.

[33] H. Shaheen, S. Agarwal, and P. Ranjan, "MinMaxScaler Binary PSO for Feature Selection," Adv. Intell. Syst. Comput., vol. 1045, pp. 705–716, 2020, DOI: 10.1007/978-981-15-0029-9_55.

[34] T. T. Dien, S. H. Luu, N. Thanh-Hai, and N. Thai-Nghe, "Deep learning with data transformation and factor analysis for student performance prediction," Int. J. Adv. Comput. Sci. Appl., vol. 11, no. 8, pp. 711–721, 2020, DOI: 10.14569/IJACSA.2020.0110886.

PLAGIARISM REPORT

Chapter No.	Chapter Name	Number of words	Plagiarism
1	Abstract	249	0%
2	Introduction	471	0%
3	Literature Survey	511	8%
4	Software & Hardware specifications	683	3%
5	Proposed System Design	1279	18%
6	Implementation & Testing	2532	15%
7	Conclusion & Future Scope	191	2%
8	References	1041	25%