ALGORITHMIC ANALYSIS OF STUDENT PERFORMANCE PREDICTION MODELS

A Capstone Phase-II project report submitted in
partial fulfilment of requirement for the award
of degree

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE & ENGINEERING
By

CHERUVUPALLI PRANATHI - (2203A52242)

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide **Mr. Dr. E.L.N. KiranKumar**, **Assoc. Prof. CS and AI** as well as Head of the CSE Department **Dr.M.Sheshikala**, **Associate Professor** for guiding us from the beginning through the end of the Capstone Phase-II project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project co-ordinators **Mr.Sallauddin Md, Asst. Prof, Y.Chanti Asst. Prof**. for their encouragement and support.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

CONTENTS

11 REFERENCES

LIST OF ACRONYMS

- MLP - Multi-Layer Perceptron
- LR - Logistic Regression
- SVM - Support Vector Machine
- LGBM - Light Gradient Boosting Machine
- DT - Decision Tree
- NB - Naive Bayes

•XGB – XGBoost

1.ABSTRACT

The dataset called Higher Education Students Performance Evaluation is designed for students in the Faculty of Engineering and Faculty of Educational Sciences. This data was collected in 2019 with the aim to forecast end-of-term performances among students by means of machine learning techniques. The dataset contains 31 variables that cover a range of demographic, socioeconomic, educational and behavioral characteristics such as student age, gender, high-school type, scholarship status, parental education level attained by parents or guardians with whom they live most often during the school year , study habits (e.g., hours per day spent studying), and academic achievements so far (e.g., GPA). The response variable is OUTPUT Grade which represents final grades achieved by students falling into one of seven categories: F – Fail; D -Poor; C -Satisfactory; B – Good; A – Very Good; AA – Excellent. Therefore this data set can be used by teachers together with various machine learning algorithms not only to identify factors influencing success in higher education but also discover how best they can assist those learners who may be struggling most based on their unique needs.

KEYWORDS

Index Terms—Role, collaborative leadership, fine of higher education

2. INTRODUCTION:

Decision-making, policy formulation, and intervention strategies to improve educational outcomes for all students. In order to gain knowledge about what affects students' academic performance in modern schools — teachers, administrators, and other interested parties must know the factors that influence it. The Higher Education Students Performance Evaluation dataset contains a variety of attributes relating to socio-economic status as well as personal development; this includes information like age groupings, gender identification etc. These datasets were collected among individuals enrolled at two faculties during 2019 – faculty for engineering sciences and faculty of educational science. It can be used effectively when applying machine learning methods which are good tools for predicting end-of-term performance levels among learners.This data set contains 32 different variables which cover many different aspects of students' lives such as their age group , whether they went through High school or not , scholarship awarded , parents' education level achieved by them , how often do they study , what is their routine like . In addition to these there are also other variables that indicate behavior problems at school like drug usage and bullying incidents with others indicating exceptional performance- what grade did the student score in their best subject ever taken ? How many credits does he have? Etc...With its wide range of variables reflecting various components influencing academic outcomes attained by learners, this record provides a lot more than just basic insights into relationship between different factors and student achievement. For instance "OUTPUT Grade" variable classifies final grades into seven classes starting from Fail up until AA which gives detailed understanding about learners' performances. By studying such data sets coupled with artificial intelligence techniques researchers will be able to identify hidden patterns within learner's records helpful for teachers/principals when designing curriculum models aimed at enhancing success rates among them within schools

3. PROBLEM STATEMENT

Design an AI-driven system to evaluate higher education students' performance utilizing machine learning algorithms. The system should analyze academic data to provide insights into student progress, predict future performance, and recommend personalized interventions for improvement.

4. MOTIVATION AND SCOPE OF WORK

The motivation behind this project lies in the need for an efficient and objective method to assess higher education students' performance, leveraging the capabilities of artificial intelligence and machine learning. By employing advanced algorithms, we aim to provide valuable insights into student progress, identify areas of improvement, and personalize educational interventions. The scope of work involves developing a comprehensive system that can analyze diverse academic data, predict student outcomes, and offer actionable recommendations to enhance learning outcomes and academic success.

5. LITERATIVE REVIEW

[1] A Abu Saa, M Al-Emran, K Shaalan focused on the literature review examines the application of qualitative research methods to the study of entrepreneurship.

[2] K Struyven, F Dochy, S Janssens considered The literature survey examines the role of qualitative studies technique in entrepreneurship studies.

[3]LD Parker proposed The literature review examines the impact of corporate governance on firm performance in emerging markets.

[4]HU Rahiman, R Kodikal developed The literature survey investigates the relationship among company social obligation (CSR) and financial performance inside the context of rising markets.

[5]Daniel Sullivan,Richard Lakeman,Debbie Massey,Dima Nasrawi,Marion Tower,Megan Lee proposed about The literature survey investigates the impact of formative evaluation practices on student getting to know results in better training.

[6] A Romhild, A Hollederer considered about he literature¨ survey investigates the effectiveness of mindfulness-based interventions in decreasing signs and symptoms of tension and melancholy amongst college students.

[7] W Karunarathne, A Calma proposed about The literature survey examines the influence of cooperative gaining knowledge of on student motivation and educational achievement in secondary education.

[8]Marie Hutchinson,Rosanne Coutts,Debbie Massey,Dima Nasrawi,Jann Fielden,Megan Lee,Richard Lakeman considered about The literature survey explores the effectiveness of project-primarily based learning in selling vital wondering capabilities among university college students.

[9]X Jin, Q Jiang, W Xiong, Y Feng,W Zhao considered totally about The literature survey examines the effectiveness of peer tutoring programs in enhancing educational performance and mastering effects amongst students. [10]JH Nieminen focused on the literature survey investigates the effectiveness of mindfulness-based interventions in reducing stress and enhancing well-being amongst university students. [11]Mohamad Sudi,Ivon Arisanti,Siti Aisyah Hanim,Ridwan Sya'rani,Kusuma Agdhi Rahwana did reaserch to prove that The literature survey investigates the effectiveness of educational interventions in selling sustainable development recognition among university students.

[12] D Boud, M Bearman proposed about the literature survey investigates the effectiveness of peer tutoring in improving instructional achievement and scholar engagement in secondary training.

[13] Joseph Crawford,Kelly-Ann Allen,Taren Sanders,Roy Baumeister,Philip Parker,Cassandra Saunders focused on The literature survey examines the effect of flipped gaining knowledge of on pupil engagement and educational performance in better schooling.

6. DATASET

	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG
1	AGE	GENDER	HS_TYPE	SCHOLARS	WORK	ACTIVITY	PARTNER	SALARY	TRANSPO	LIVING	MOTHER_E	FATHER_E	#_SIBLING	KIDS	MOTHER_J	FATHER_J	STUDY_HR	READ_FRE	READ_FRE	ATTEND_D	IMPACT	ATTEND	PREP_STU	PREP_EXA	NOTES	LISTENS	LIKES_DIS	CLASSROO	CUML_GPA	EXP_GPA	COURSE ID	GRADE
2	2	2	3	3	1	2	2	1	1	1	1	2	3	1	2	5	3	2	2	1	1	1	1	1	3	2	1	2	1	1	1	1
3	2	2	3	3	1	2	2	1	1	1	2	3	2	1	2	1	2	2	2	1	1	1	1	1	3	2	3	2	2	3	1	1
4	2	2	2	3	2	2	2	2	4	2	2	2	2	1	2	1	2	1	2	1	1	1	1	1	2	2	1	1	2	2	1	1
5	1	1	1	3	1	2	1	2	1	2	1	2	5	1	2	1	3	1	2	1	1	1	1	2	3	2	2	1	3	2	1	1
6	2	2	1	3	2	2	1	3	1	4	3	3	2	1	2	4	2	1	1	1	1	1	2	1	2	2	2	1	2	2	1	1
7	2	2	2	3	2	2	2	2	1	1	3	3	2	1	2	3	1	1	2	1	1	1	1	1	1	2	1	2	4	4	1	2
8	1	2	2	4	2	2	2	1	1	3	1	3	1	1	2	4	2	2	2	2	1	2	1	1	3	3	3	3	4	4	1	5
9	1	1	2	3	1	1	1	2	2	3	4	3	1	1	4	3	1	2	2	1	1	1	3	1	3	2	2	1	1	1	1	2
10	2	1	3	3	2	1	1	1	1	3	2	4	2	1	2	4	1	2	2	1	1	1	1	1	3	2	2	2	4	3	1	5
11	2	1	2	3	2	2	1	3	4	2	1	2	3	1	2	3	2	2	2	1	1	2	1	1	2	2	2	2	1	2	1	0
12	1	1	1	3	2	2	2	3	2	3	3	4	2	1	3	2	1	1	1	1	1	2	1	1	2	2	2	2	1	1	1	2
13	1	1	1	4	1	1	2	4	2	3	5	5	1	1	3	2	3	3	3	1	3	1	3	2	3	1	3	3	4	3	1	0
14	1	1	1	4	2	2	2	1	1	1	3	5	4	2	2	2	3	2	2	1	1	1	1	1	2	2	2	3	4	2	1	0
15	2	1	2	5	2	2	2	1	1	1	2	2	2	1	2	3	1	2	1	1	1	2	1	1	3	2	3	3	4	2	1	1
16	3	2	2	4	1	1	2	3	4	2	3	1	2	1	4	1	2	2	2	1	1	1	1	1	2	3	2	1	4	4	1	2
17	2	2	2	3	2	2	2	1	1	2	4	4	2	1	3	1	2	2	2	2	3	2	1	1	3	2	2	3	2	2	1	2
18	1	1	2	5	2	1	2	1	1	1	2	2	4	1	2	3	2	2	2	1	1	2	1	1	3	2	3	3	4	3	1	1
19	2	2	2	3	2	2	2	1	1	1	2	2	2	1	2	1	2	2	2	2	1	2	1	1	2	2	2	2	2	2	1	2
20	1	1	2	4	2	2	2	3	1	1	2	2	5	1	2	5	5	3	2	2	1	2	1	1	3	1	3	3	3	3	1	2
21	1	2	1	3	2	2	1	2	2	2	3	3	3	1	2	4	4	2	2	1	2	1	1	1	3	2	2	3	2	3	1	3
22	1	2	2	5	1	2	1	1	4	2	3	3	3	2	2	3	4	2	2	2	1	1	1	2	3	1	2	3	4	4	1	1
23	1	2	2	5	2	2	1	1	4	2	2	2	4	1	2	3	3	2	2	1	1	1	1	1	3	1	3	3	3	3	1	1
24	2	2	2	3	1	2	1	1	1	1	1	1	4	2	2	4	3	3	3	1	1	1	1	2	3	1	2	3	3	3	1	3
25	3	2	2	2	1	1	2	5	1	1	1	4	3	1	2	1	3	2	3	1	1	2	1	1	3	2	3	3	3	3	1	1
26	2	2	2	3	2	2	2	2	1	1	3	1	3	1	2	4	1	2	2	1	1	1	1	1	2	1	3	2	4	4	1	2
27	2	2	2	3	2	2	1	1	1	2	1	4	3	1	2	4	2	2	2	1	1	1	1	1	2	1	3	2	1	2	1	3
28	2	2	2	3	2	1	1	1	1	1	4	4	4	1	2	4	2	3	2	1	1	1	1	1	3	3	3	2	2	1	1	1
29	1	2	1	3	1	2	2	1	1	1	1	1	4	1	2	3	3	2	2	1	1	2	1	1	3	1	2	1	2	1	1	1
30	3	2	2	3	2	2	1	1	4	2	2	2	4	1	2	1	2	2	2	1	1	1	1	1	3	2	3	3	5	4	1	3
31	2	2	3	4	2	2	2	1	4	2	3	3	2	1	4	1	2	2	2	1	1	1	1	1	3	2	2	2	4	3	1	5
32	2	2	2	5	1	1	1	1	1	2	1	1	5	1	2	4	2	2	2	1	1	1	2	1	2	3	3	3	5	4	1	5

7. PROPOSED METHODOLOGY

A.Overview

Machine learning methods are being used to estimate the success rate of higher education students by collecting data from educational institutions and engineering colleges Logistic regression models,lightgbm,XGB boost,Naive Bayes,Multi layer perceptron,decision tree and Support Vector Machines (SVM) are utilized to predict student performance, with logistic regression estimating the probability of outcomes and SVM being a classification technique Predictive models in

education focus on evaluating high school student performance using various materials and machine learning techniques, providing valuable insights for educators and policymakers

7.1 DATA PRE-PROCESSING

In the dataset the target or the analysing variable is Grade in that there are 0-7 values repeatedly so i came to known that the dataset is a classification.it is also known as multi classification dataset.With original dataset the accuracy is not proper so i changes to a classification dataset by changing 0-3 are 0 and 3-7 are 1.In dataset there is one string with the column name student Id i dropped it.later i filled the null values using fillna().

Implementation

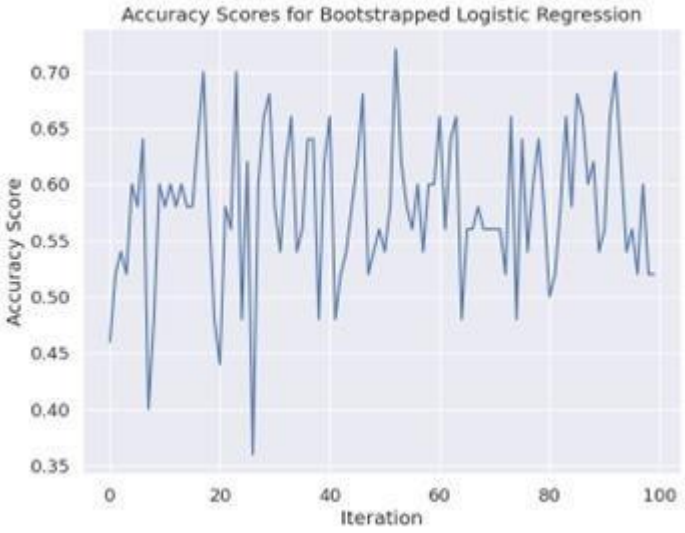
logistic regression:

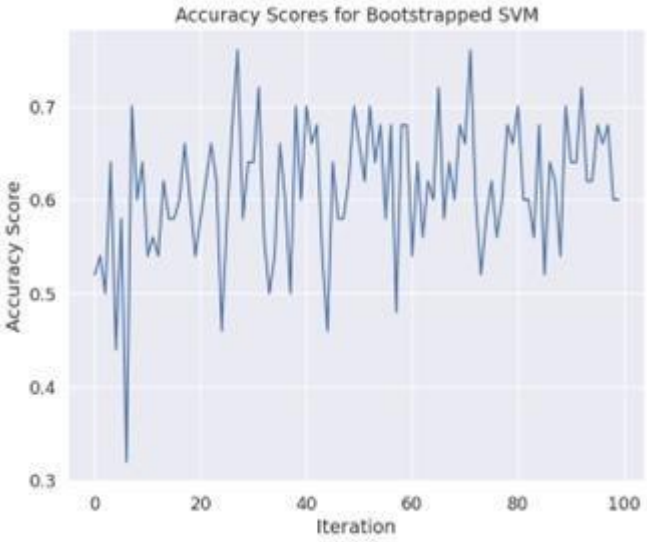
Logistic regression is a mathematical version for binary type duties, wherein the intention is to expect certainly one of two feasible consequences (e.G., 0 or 1, yes or no, real or false) primarily based on one or greater enter traits on. Unlike linear regression, which predicts a continuous outcome, logistic regression predicts the possibility of two consequences. A logistic characteristic (also referred to as a sigmoid characteristic) is used to version this probability. Here is an evidence of logistic regression with its formula. Measurement: A logistic regression model uses the logistic characteristic to estimate the chance (P) of a couple of events (e.G., the probability of a tremendous final results) the usage of the logistic function:

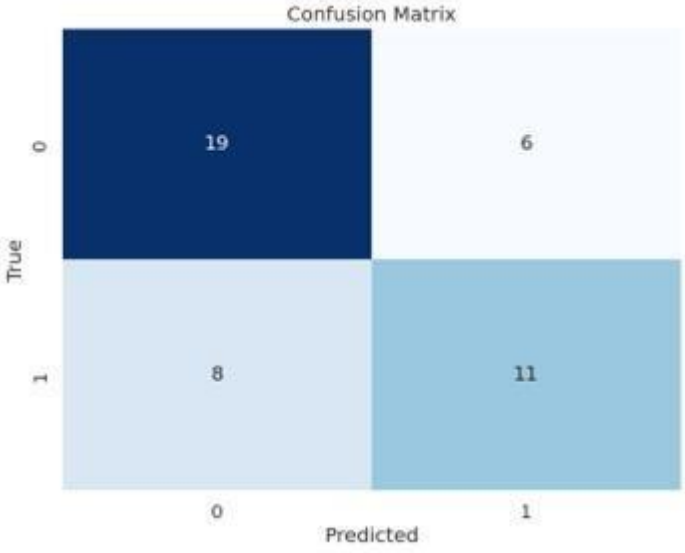
z = w0w1x1w2x2−−−−−−−−−−−−−−−−−−wnxn

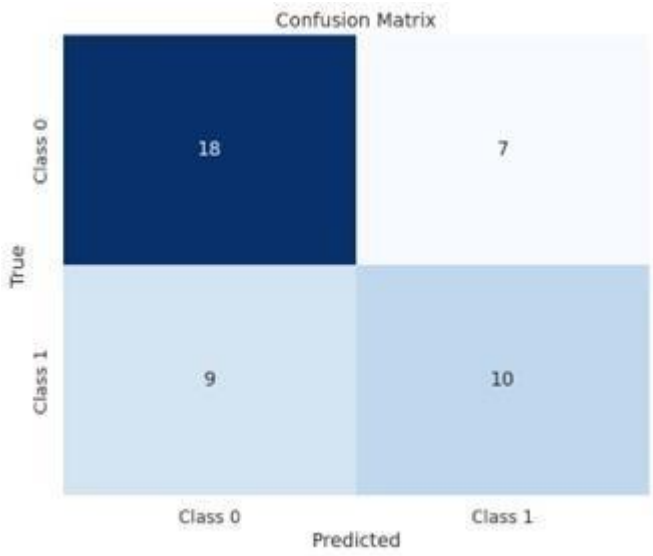
(1) yp = 1/1e−z (2) z = −Y log(Y P)−(1−Y)INTERPRETATION(1−Y P)

(3)


Accuracy Scores for Bootstrapped Logistic Regression
Accuracy Score
Iteration
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
20
40
60
80
100


Accuracy Scores for Bootstrapped SVM
Accuracy Score
Iteration
0.7
0.6
0.5
0.4
0.3
0
20
40
60
80
100


Confusion Matrix
True
Predicted
19
6
8
11
0
1
0
1



7.2 LONG SHORT-TERM MEMORY:

The proposed model for Higher Education Students Performance Evaluation utilizes clustering techniques followed by classification and regression methods to analyze academic data. Initially, the input data, comprising student performance metrics, is subjected to clustering using diverse algorithms including K-Means, Hierarchical, DBSCAN, Agglomerative, KMedoids, and Gaussian Mixture Models (GMM). These clustering methods partition the data into meaningful clusters based on similarity. Subsequently, each cluster is analyzed separately using classification techniques such as Logistic Regression, Support Vector Machines, Decision Trees, Naive Bayes, multi-layer perceptrons, convolutional neural networks, XGBoost, and LightGBM, to predict students' academic outcomes. Additionally, regression models including Decision Tree Regression, Support Vector Regression (SVR), Random Forest Regression, K-Nearest Neighbor's (KNN) Regression, ElasticNet Regression, and Neural Networks (Multi-layer Perceptron) are employed to forecast future performance trends and provide insights into academic progress. This comprehensive approach leverages the power of artificial intelligence and machine learning to enhance student performance evaluation in higher education settings.

7.3 COMPARED ALGORITHMS

.• Support Vector Machine:

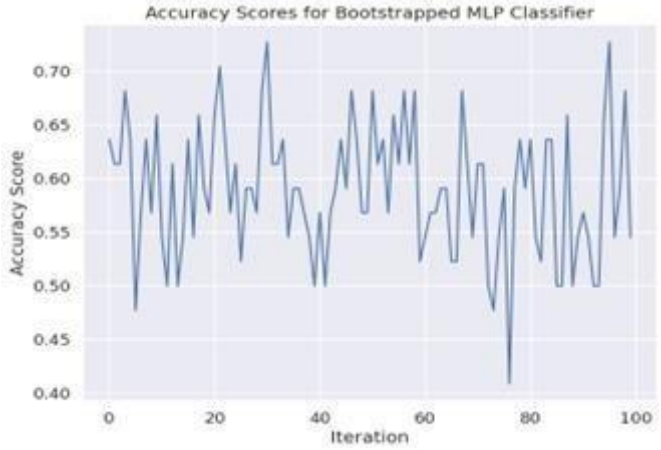
-Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, it's far used for Classification in addition to for Regression issues. However, it's miles specially used for classification issues in machine learning.

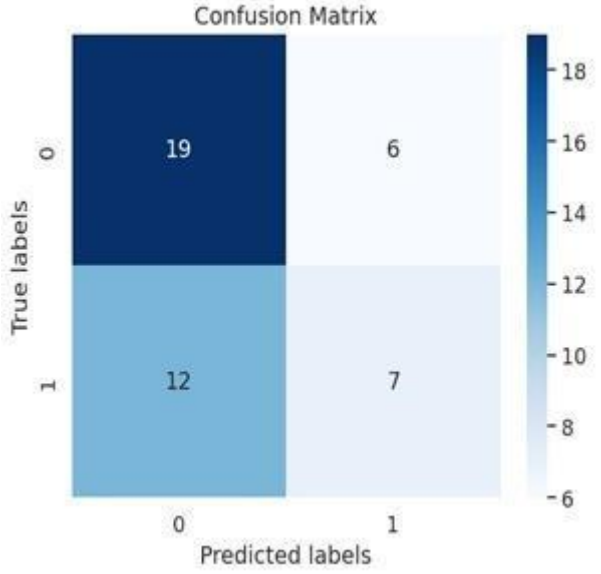
-The intention of the SVM set of rules is to create an most useful line or decision boundary that can separate the n-dimensional area into training so that within the destiny we can easily assign a brand new facts point to the suitable range. This choicest choice constraint is called a hyperplane.

• Multi-layer Perceptron:

-A Multi-Layer Perceptron (MLP) is a fundamental form of artificial neural network such as more than one layers of interconnected nodes or neurons. It generally accommodates an enter layer, one or greater hidden layers, and an output layer. Each neuron within the community gets enter alerts, strategies them thru a non-linear activation function, and transmits an output to neurons within the next layer.

-MLPs are able to learning complex styles and relationships within statistics, making them widely used in various fields consisting of pattern recognition, class, and regression tasks. Through a manner called backpropagation, MLPs regulate the weights of connections among neurons all through schooling, aiming to decrease the difference between expected and actual outputs. -Despite their effectiveness, MLPs have barriers, together with vulnerability to overfitting and problem in handling excessive-dimensional data. Nevertheless, they remain a cornerstone within the subject of deep studying, forming the idea for more stateof-the-art neural network architectures.



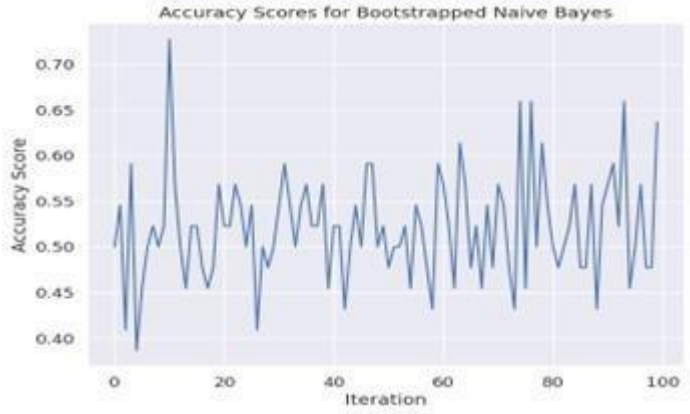


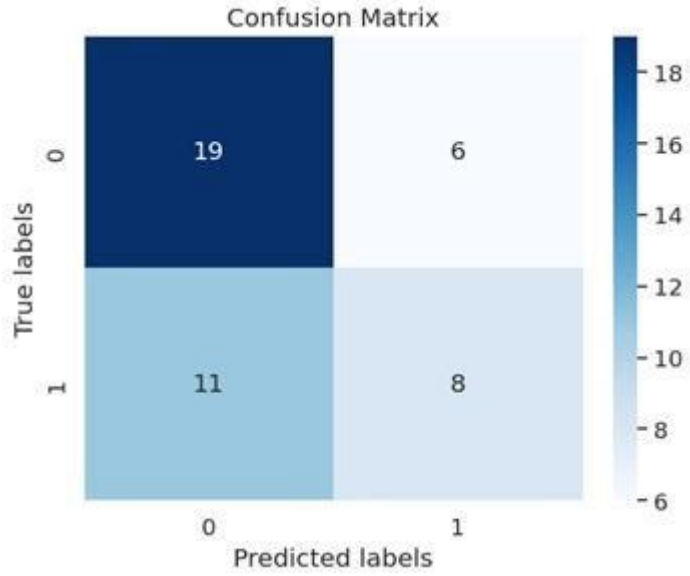
• Naive Bayes:

-Naive Bayes is a famous set of rules in machine gaining knowledge of for type duties. It's based totally on Bayes' theorem, which calculates the possibility of a hypothesis given the proof. Despite its simplicity, Naive Bayes often performs remarkably nicely, mainly in text type and spam filtering.

-The "naive" issue of Naive Bayes comes from its assumption of independence among functions. It assumes that each function contributes independently to the chance of a selected final results, which may not hold actual in real-international scenarios. However, this simplifying assumption allows for immediate schooling and prediction, making Naive Bayes specially beneficial for massive datasets.

-Naive Bayes works with the aid of calculating the possibility of each elegance given the enter features and choosing the class with the very best possibility as the prediction. Despite its "naive" assumptions, Naive Bayes may be extraordinarily effective and is broadly utilized in numerous programs because of its simplicity, efficiency, and frequently aggressive overall performance

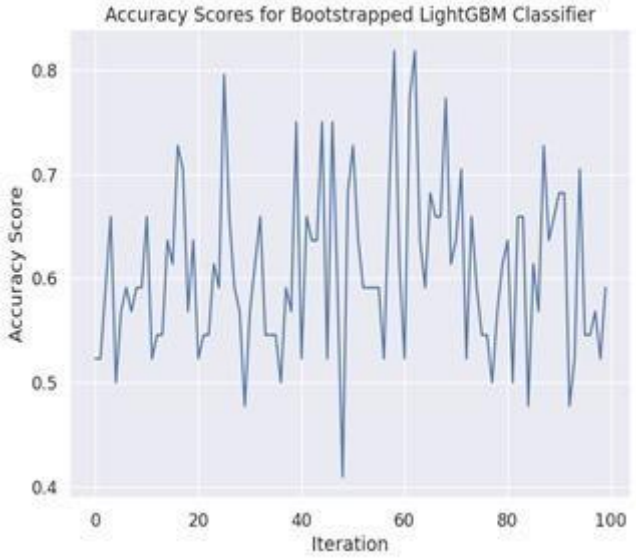


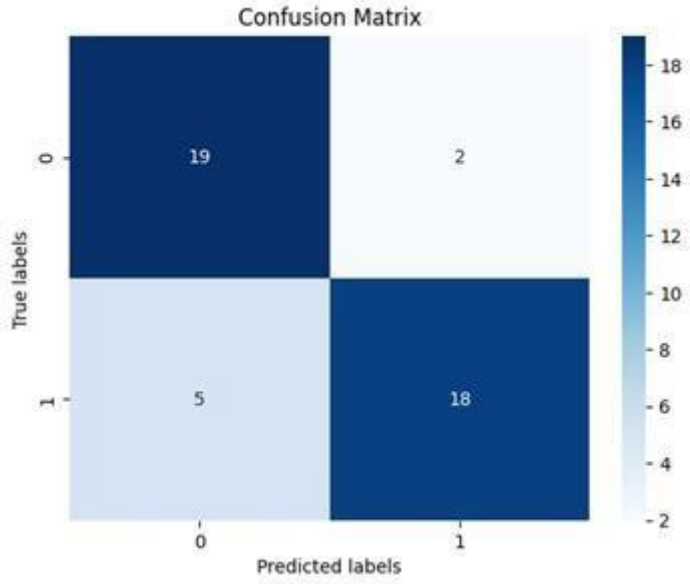


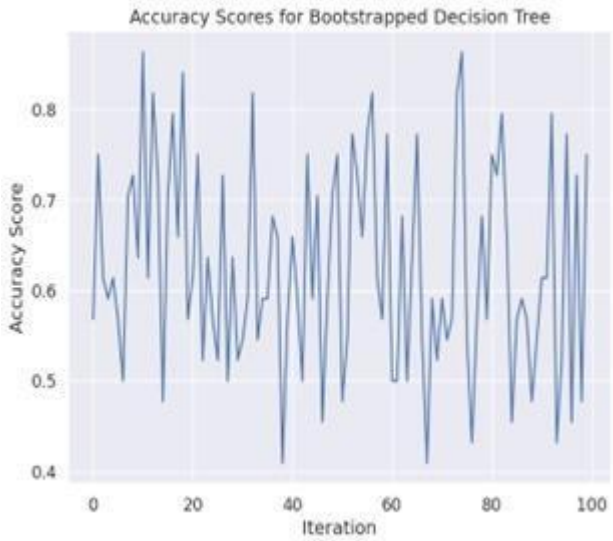
• Lightgbm:

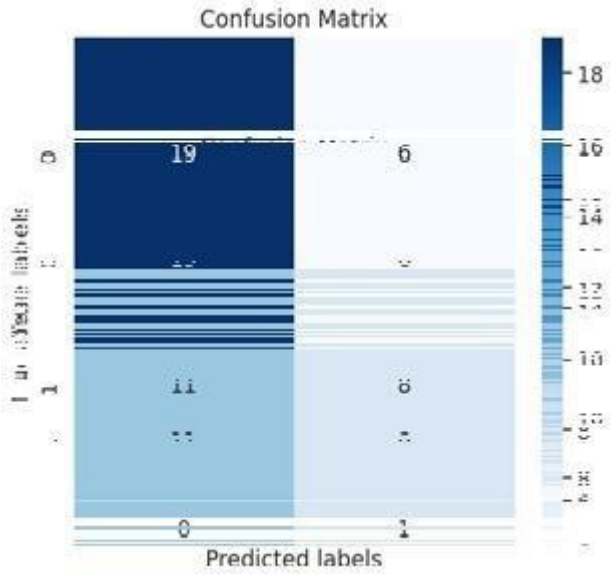
LightGBM, brief for Light Gradient Boosting Machine, is a high-performance gradient boosting framework developed via Microsoft. It's designed for performance and speed in

handling large datasets and is widely used in device studying competitions and enterprise programs. LightGBM makes use of a tree-based mastering set of rules, using a singular technique referred to as Gradientbased totally One-Side Sampling (GOSS) to reduce memory usage and enhance schooling velocity. It additionally utilizes Histogram-primarily based algorithms for quicker computation. Its flexibility, scalability, and capacity to handle specific features make it a famous desire for diverse tasks, along with category, regression, and ranking problems, continuously handing over competitive effects with minimal tuning.







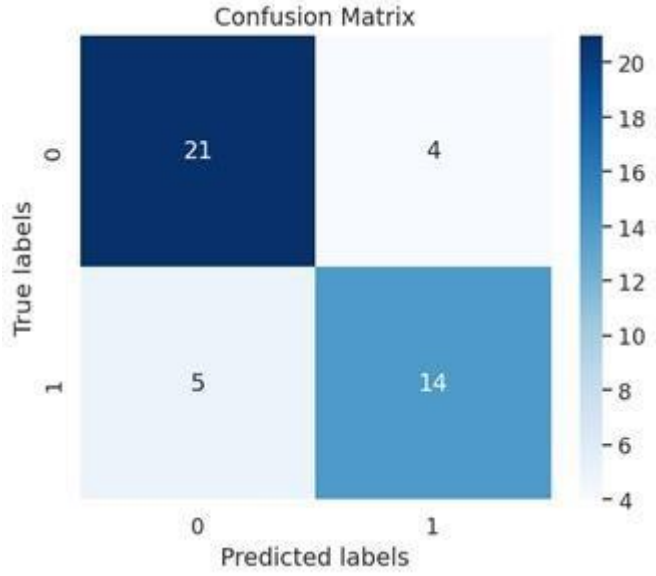


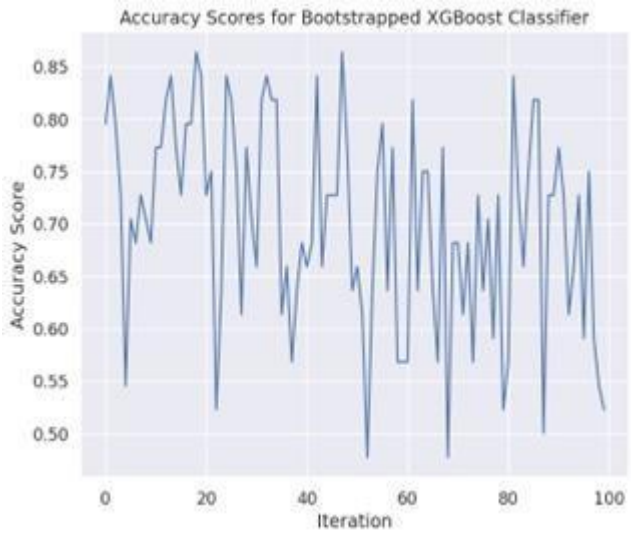
• Decision Tree:

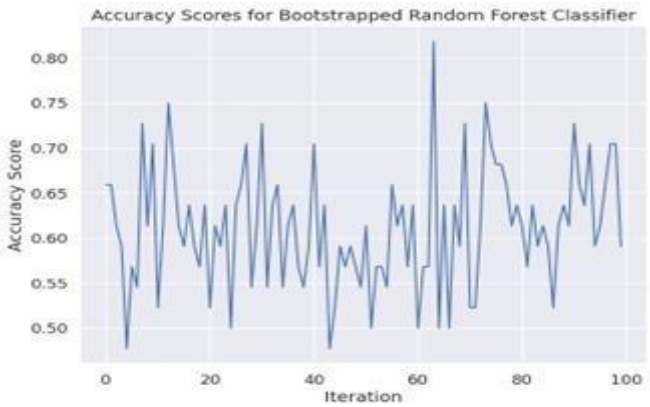
A decision tree is a predictive modeling tool utilized in facts mining and machine getting to know. It organizes records right into a tree-like structure of choices and their viable outcomes. Each internal node represents a ”take a look at” on an characteristic, every branch represents the final results of the test, and every leaf node represents a class label or a choice. Decision timber are famous due to their simplicity and interpretability, making them treasured for knowhow and explaining the decisionmaking technique. They’re employed in various fields, from finance to healthcare, aiding in class, regression, and solving complicated choice-making troubles easily and transparency.

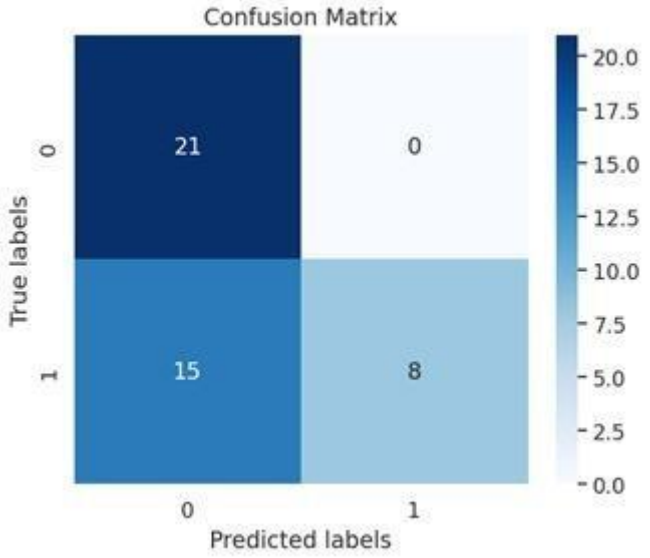
• XG Boost:

XGBoost, short for Extreme Gradient Boosting, is a powerful machine learning set of rules famend for its efficiency and performance in classification and regression responsibilities. It employs a gradient boosting framework, iteratively combining vulnerable beginners, normally decision trees, to shape a robust predictive model. XGBoost contains regularization techniques to save you overfitting and makes use of parallel computing to beautify velocity. Its versatility, scalability, and effectiveness in handling huge datasets make it a popular desire in numerous domain names, from finance to healthcare and past. With its capability to deliver excessive accuracy and interpretability, XGBoost stands as a cornerstone in modern-day device studying workflows.









• Random Forest:

Random Forest is a versatile ensemble studying set of rules widely used for each category and regression duties. It operates by building a mess of decision timber in the course of training and outputs the mode of the training (class) or the imply prediction (regression) of man or woman bushes. Each tree inside the wooded area is trained on a random subset of the schooling information and makes impartial predictions. This approach mitigates overfitting and complements generalization overall performance. Random Forest is famend for its robustness, scalability to big datasets, and capability to address high-dimensional feature areas. It's a popular choice across diverse domain names due to its simplicity, flexibility, and high predictive accuracy.

8. RESULTS & DISCUSSION

Code:

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import classification_report
```

```
import pandas as pd
do=pd.read_csv("/content/student_prediction.csv")
print(do)
```

```
     STUDENTID  AGE  GENDER  HS_TYPE  SCHOLARSHIP  WORK  ACTIVITY  PARTNER  \
0      STUDENT1    2       2        3            3     1         2        2
1      STUDENT2    2       2        3            3     1         2        2
2      STUDENT3    2       2        2            3     2         2        2
3      STUDENT4    1       1        1            3     1         2        1
4      STUDENT5    2       2        1            3     2         2        1
..          ...  ...     ...      ...          ...   ...       ...      ...
140  STUDENT141    2       1        2            3     1         1        2
141  STUDENT142    1       1        2            4     2         2        2
142  STUDENT143    1       1        1            4     2         2        2
143  STUDENT144    2       1        2            4     1         1        1
144  STUDENT145    1       1        1            5     2         2        2

     SALARY  TRANSPORT  ...  PREP_STUDY  PREP_EXAM  NOTES  LISTENS  \
0         1          1  ...           1          1      3        2
1         1          1  ...           1          1      3        2
2         2          4  ...           1          1      2        2
3         2          1  ...           1          2      3        2
4         3          1  ...           2          1      2        2
..      ...        ...  ...         ...        ...    ...      ...
140       1          1  ...           1          1      2        1
141       1          4  ...           1          1      3        2
142       1          1  ...           1          1      3        3
143       5          2  ...           2          1      2        1
144       3          1  ...           2          1      3        2

     LIKES_DISCUSS  CLASSROOM  CUML_GPA  EXP_GPA  COURSE ID  GRADE
0                1          2         1        1          1      1
1                3          2         2        3          1      1
2                1          1         2        2          1      1
3                2          1         3        2          1      1
4                2          1         2        2          1      1
..             ...        ...       ...      ...        ...    ...
140              2          1         3        3          9      5
141              2          1         5        3          9      5
142              2          1         4        3          9      1
143              2          1         5        3          9      4
144              3          1         5        4          9      3

[145 rows x 33 columns]
```

```
do.head()
```

	STUDENTID	AGE	GENDER	HS_TYPE	SCHOLARSHIP	WORK	ACTIVITY	PARTNER	SALARY	TRANSPORT	...	PREP_STUDY	PREP_EXAM	NOTES	LISTENS	LIKES_DI
0	STUDENT1	2	2	3	3	1	2	2	1	1	...	1	1	3	2	
1	STUDENT2	2	2	3	3	1	2	2	1	1	...	1	1	3	2	
2	STUDENT3	2	2	2	3	2	2	2	2	4	...	1	1	2	2	
3	STUDENT4	1	1	1	3	1	2	1	2	1	...	1	2	3	2	
4	STUDENT5	2	2	1	3	2	2	1	3	1	...	2	1	2	2	

5 rows × 33 columns

0s

```
f=do['STUDENTID']
d=do.drop('STUDENTID',axis=1)
```

0s

```
print(d)
```

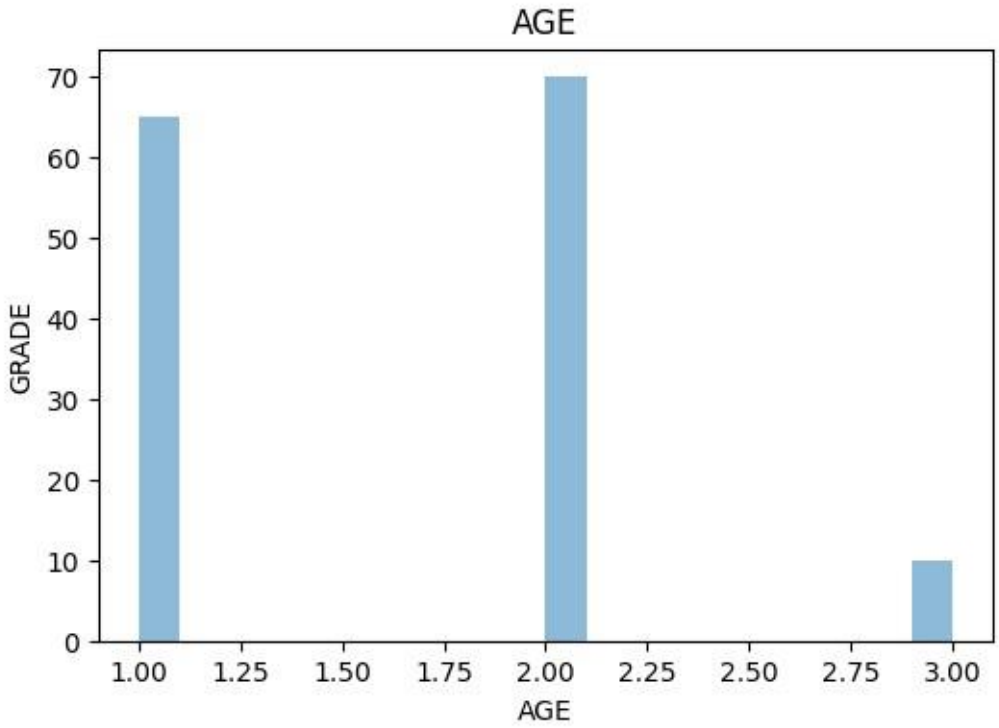
```
     AGE   GENDER   HS_TYPE   SCHOLARSHIP   WORK   ACTIVITY   PARTNER   SALARY   \
0      2        2         3             3      1          2         2        1
1      2        2         3             3      1          2         2        1
2      2        2         2             3      2          2         2        2
3      1        1         1             3      1          2         1        2
4      2        2         1             3      2          2         1        3
..   ...      ...       ...           ...    ...        ...       ...      ...
140    2        1         2             3      1          1         2        1
141    1        1         2             4      2          2         2        1
142    1        1         1             4      2          2         2        1
143    2        1         2             4      1          1         1        5
144    1        1         1             5      2          2         2        3

     TRANSPORT   LIVING   ...   PREP_STUDY   PREP_EXAM   NOTES   LISTENS   \
0             1        1   ...            1           1       3         2
1             1        1   ...            1           1       3         2
2             4        2   ...            1           1       2         2
3             1        2   ...            1           2       3         2
4             1        4   ...            2           1       2         2
..          ...      ...   ...          ...         ...     ...       ...
140           1        2   ...            1           1       2         1
141           4        2   ...            1           1       3         2
142           1        1   ...            1           1       3         3
143           2        3   ...            2           1       2         1
144           1        1   ...            2           1       3         2

     LIKES_DISCUSS   CLASSROOM   CUML_GPA   EXP_GPA   COURSE ID   GRADE
0                1           2          1         1           1       1
1                3           2          2         3           1       1
2                1           1          2         2           1       1
3                2           1          3         2           1       1
4                2           1          2         2           1       1
..             ...         ...        ...       ...         ...     ...
140              2           1          3         3           9       5
141              2           1          5         3           9       5
142              2           1          4         3           9       1
143              2           1          5         3           9       4
144              3           1          5         4           9       3

[145 rows x 32 columns]
```

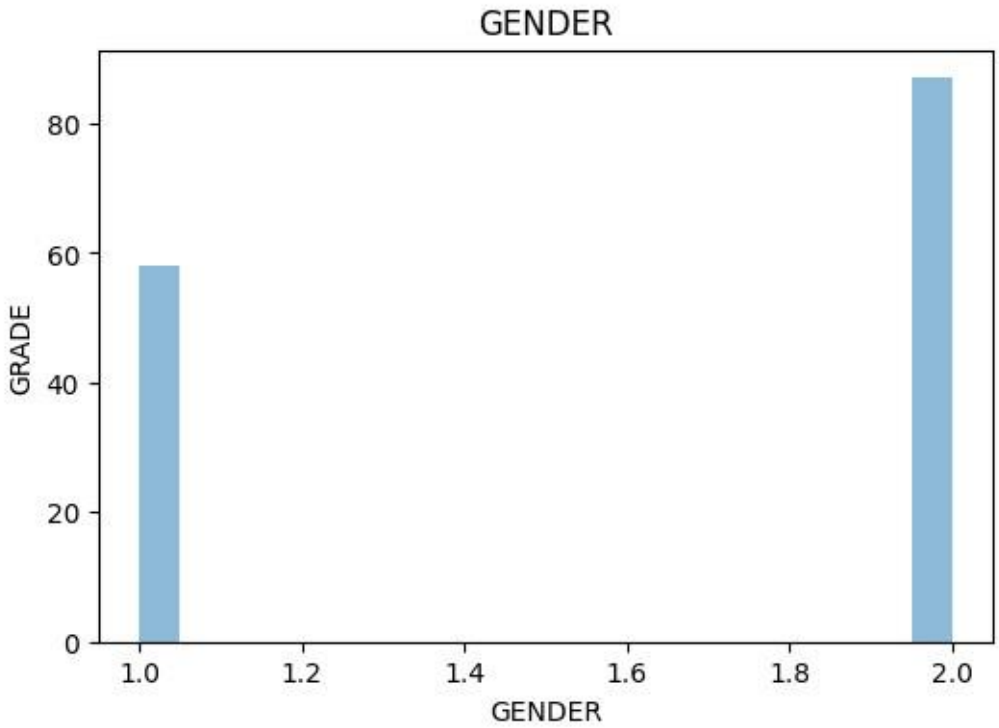
```
[6]  y=d['GRADE']
     x=d.drop('GRADE',axis=1)
```

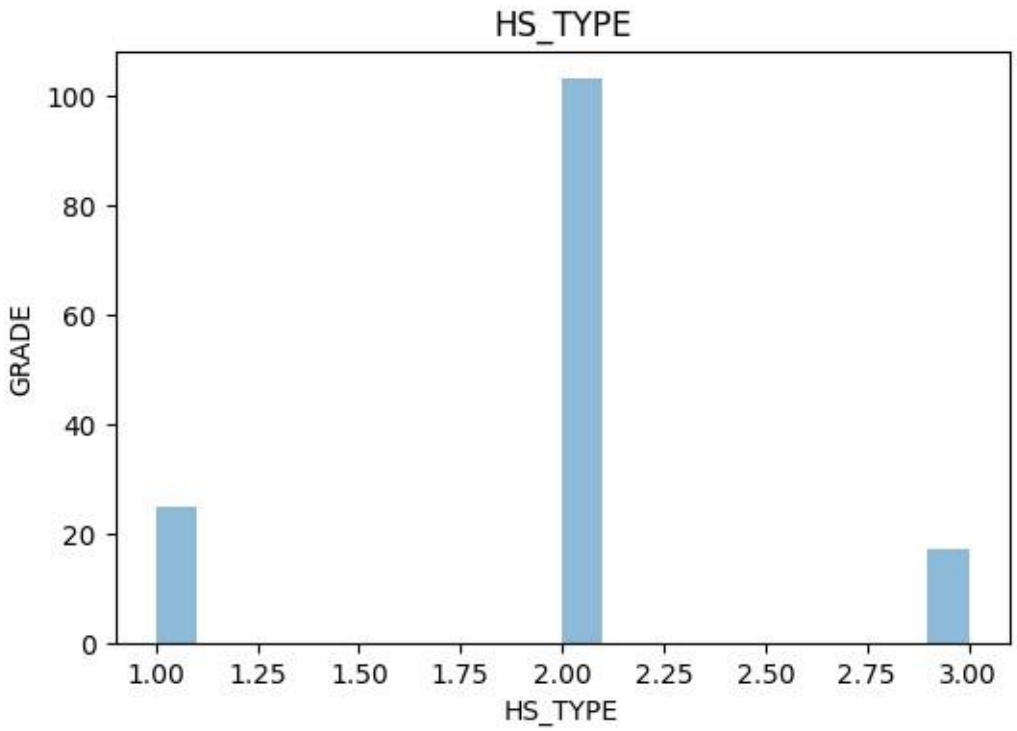
```
for i in range(len(y)):
  if y[i]>=3.5:
    y[i]=1
  else:
    y[i]=0
print(y)
```

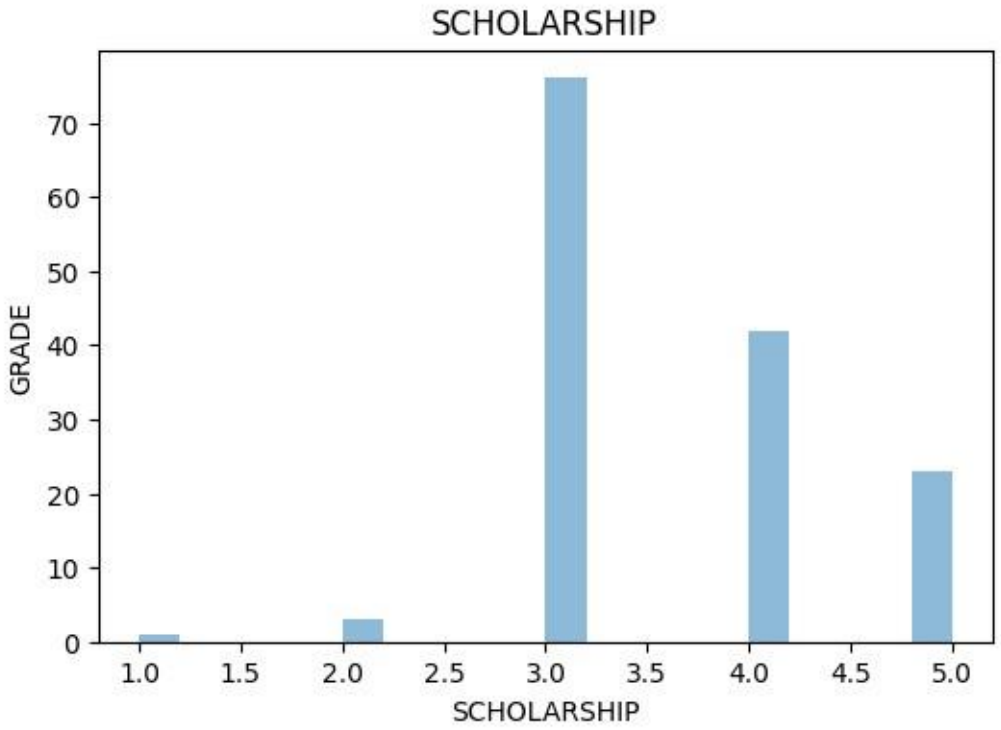
```
0      0
1      0
2      0
3      0
4      0
      ..
140    1
141    1
142    0
143    1
144    0
Name: GRADE, Length: 145, dtype: int64
```

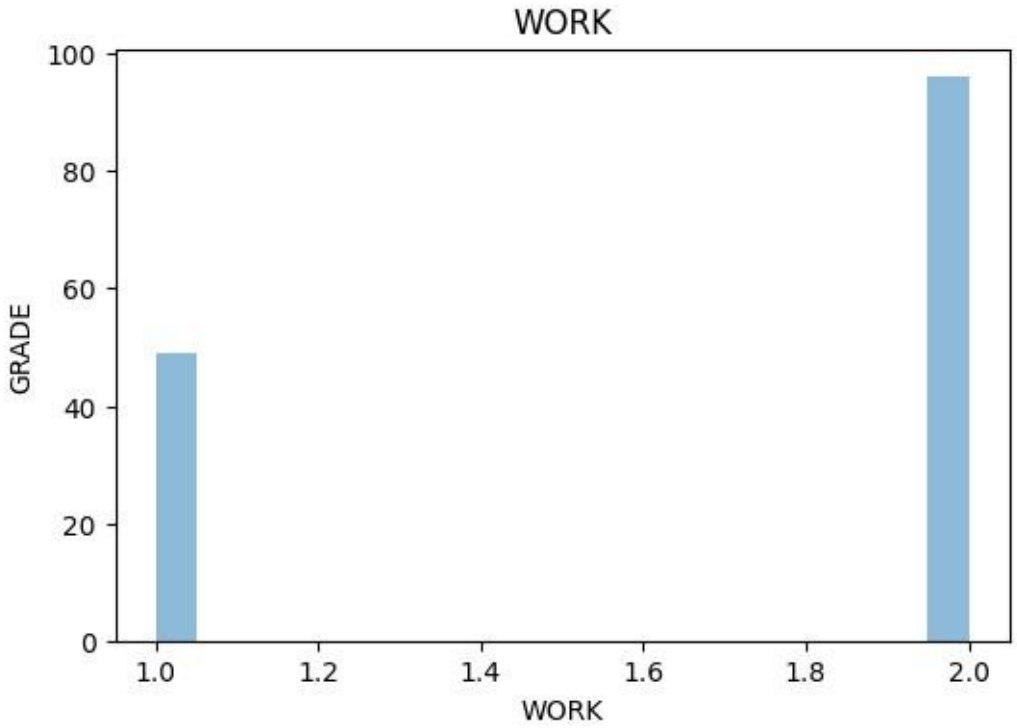
```
import matplotlib.pyplot as plt
for column in d.columns:
    plt.figure(figsize=(6, 4))
    plt.hist(d[column], bins=20, alpha=0.5)
    plt.xlabel(column)
    plt.ylabel("GRADE")
    plt.title(f"{column}")
    plt.show()
plt.figure(figsize=(6, 4))
plt.show()
```

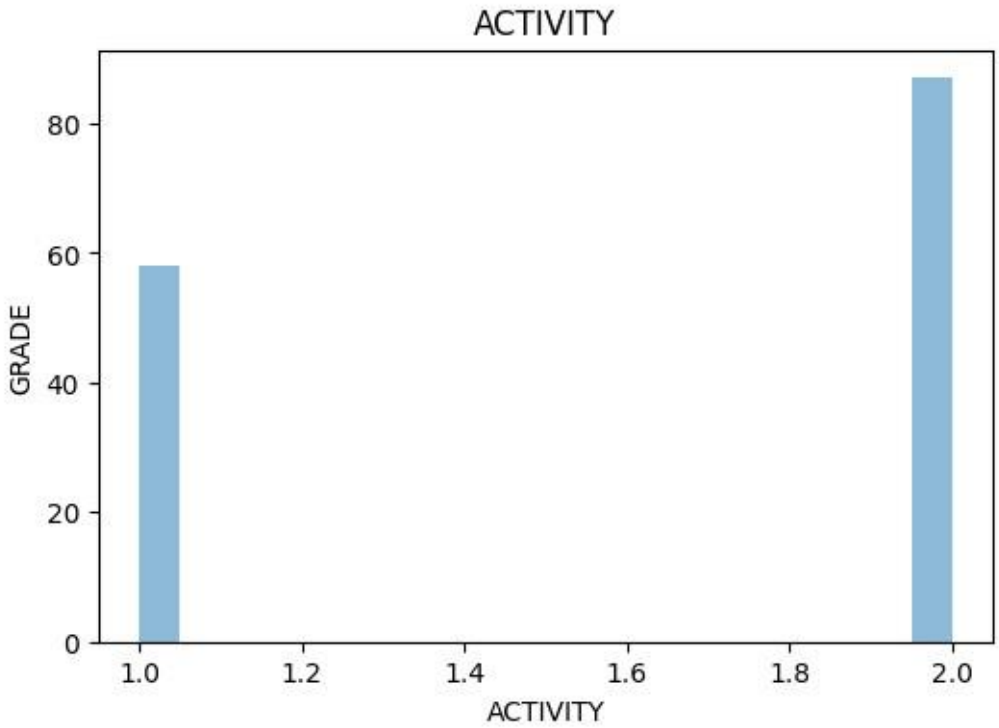

AGE
GRADE
70
60
50
40
30
20
10
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
AGE

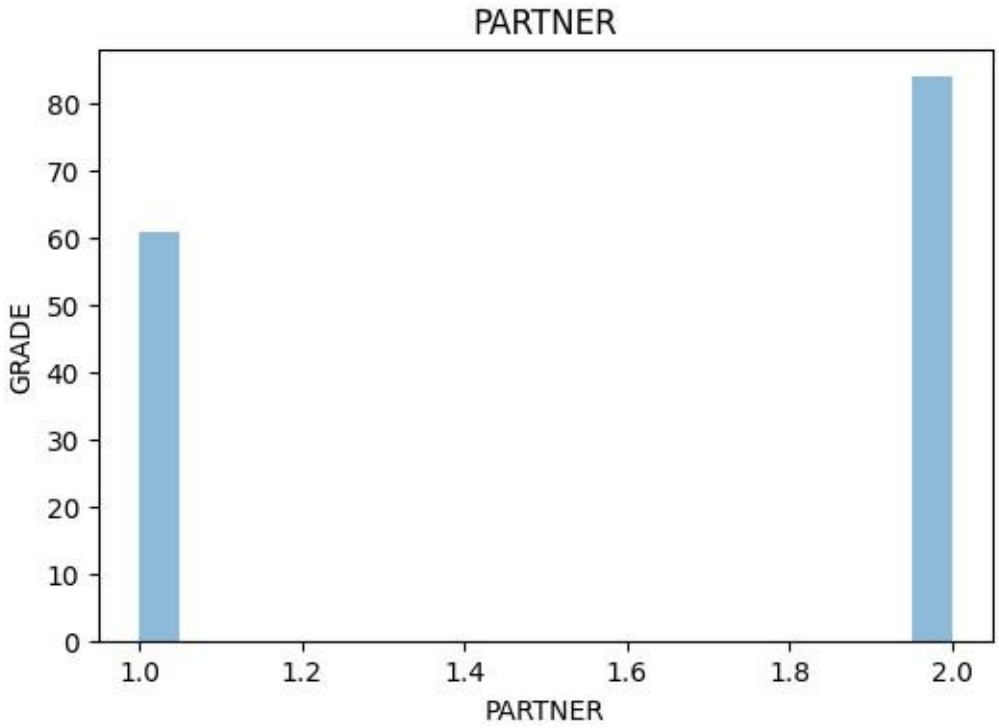

GENDER
GRADE
80
60
40
20
0
1.0
1.2
1.4
1.6
1.8
2.0
GENDER

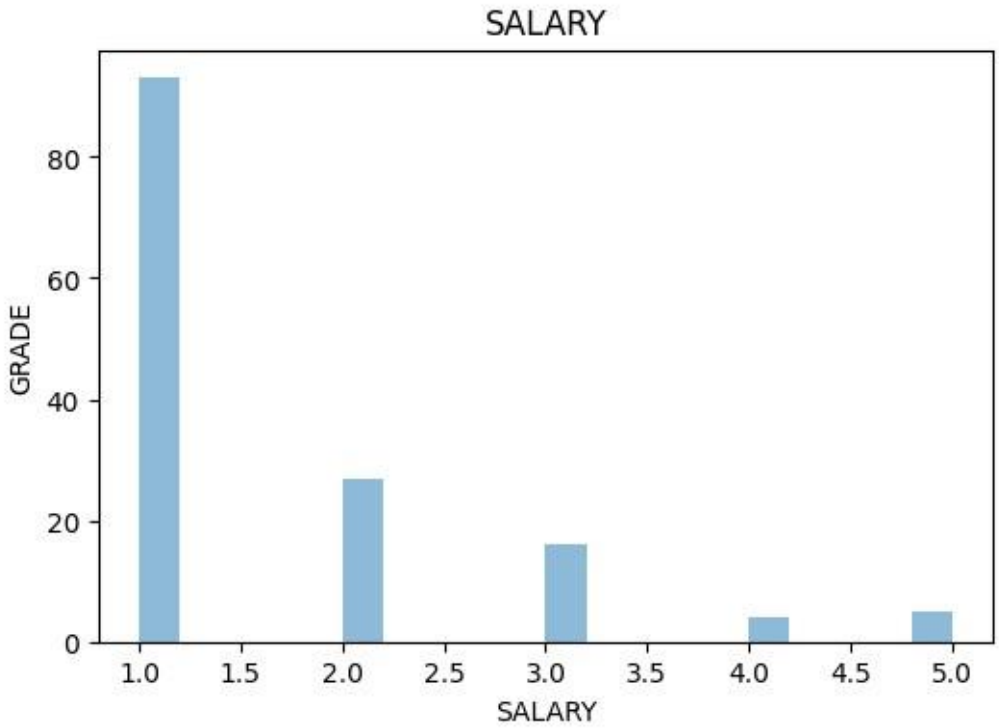

HS_TYPE
GRADE
100
80
60
40
20
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
HS_TYPE

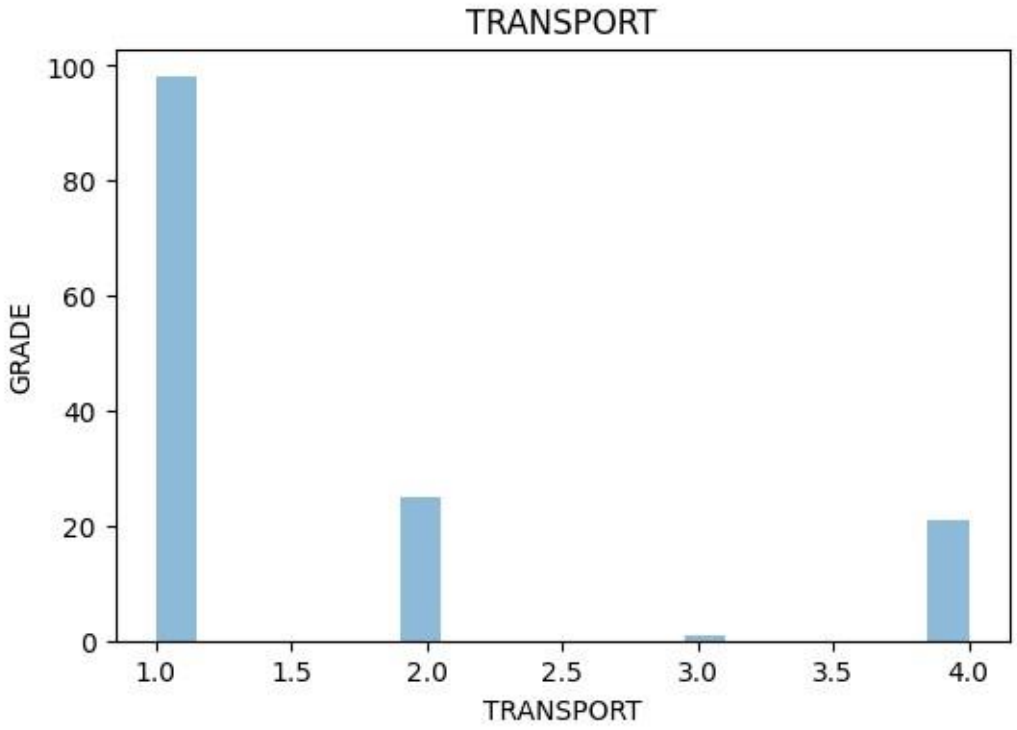

SCHOLARSHIP
GRADE
70
60
50
40
30
20
10
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
SCHOLARSHIP

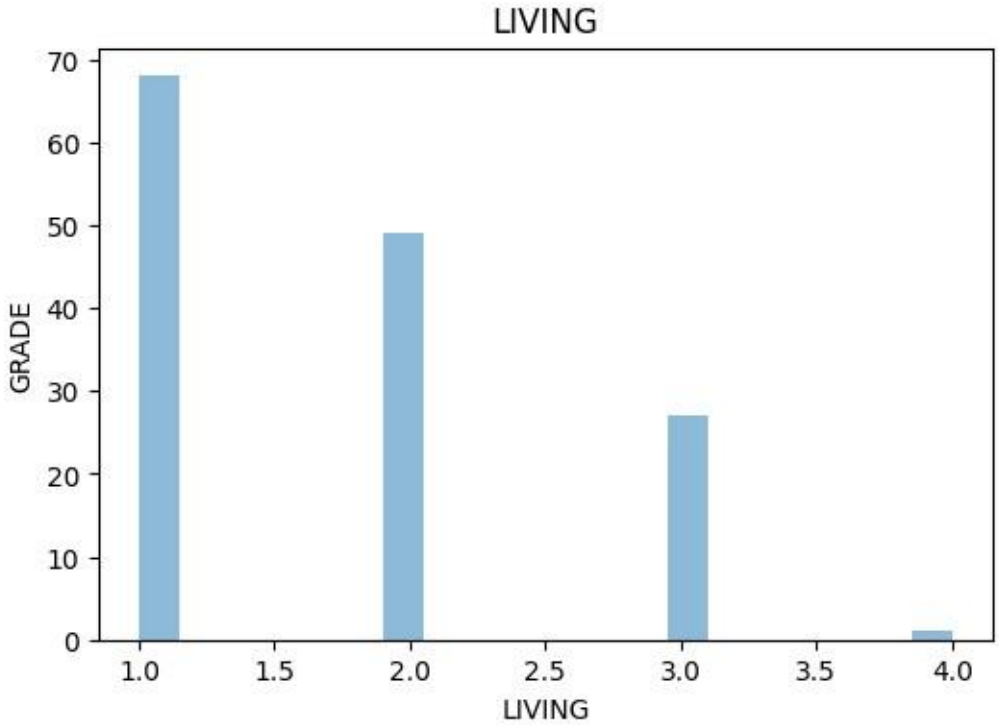

WORK
100
80
60
40
20
0
GRADE
1.0
1.2
1.4
1.6
1.8
2.0
WORK

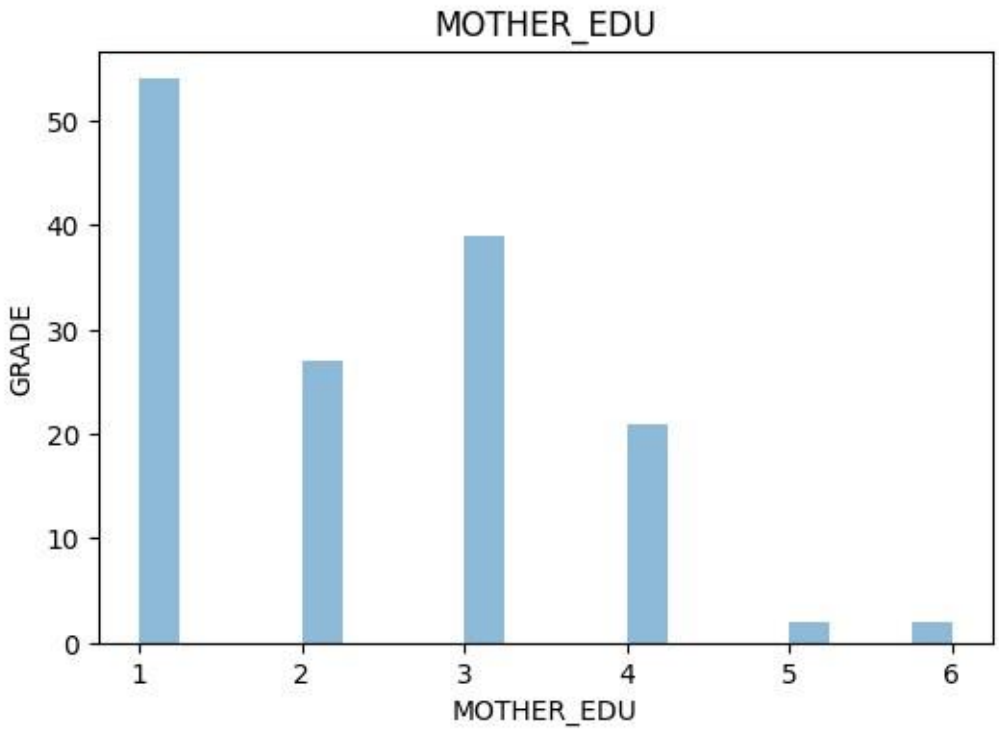

ACTIVITY
80
60
40
20
0
GRADE
1.0
1.2
1.4
1.6
1.8
2.0
ACTIVITY

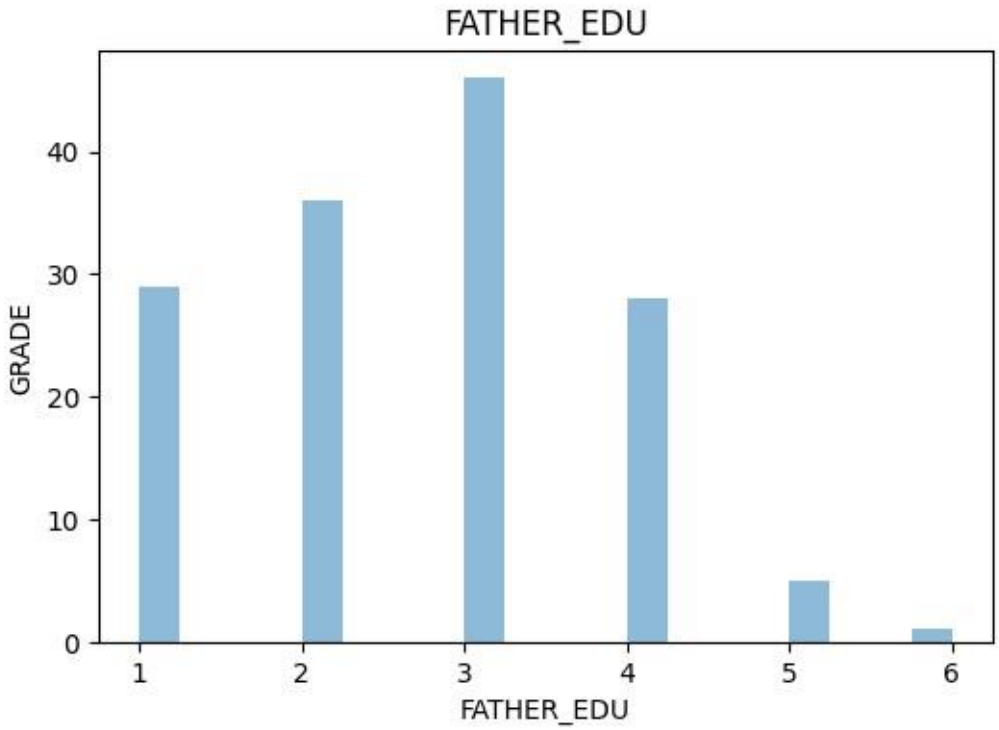

PARTNER
GRADE
0
10
20
30
40
50
60
70
80
1.0
1.2
1.4
1.6
1.8
2.0
PARTNER

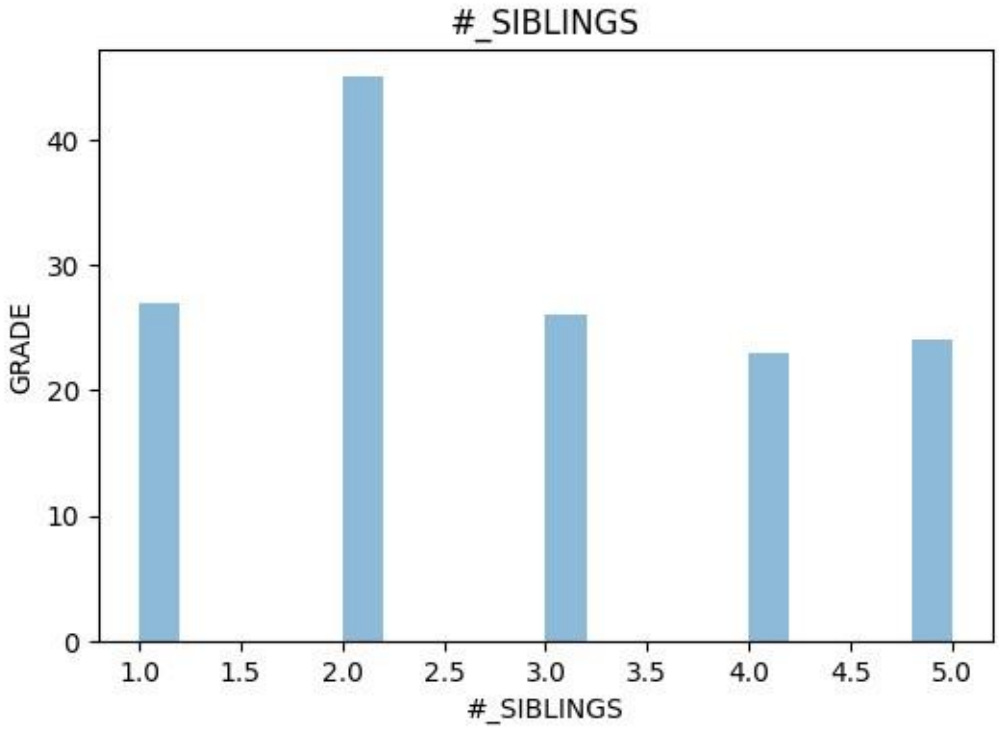

SALARY
GRADE
0
20
40
60
80
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
SALARY

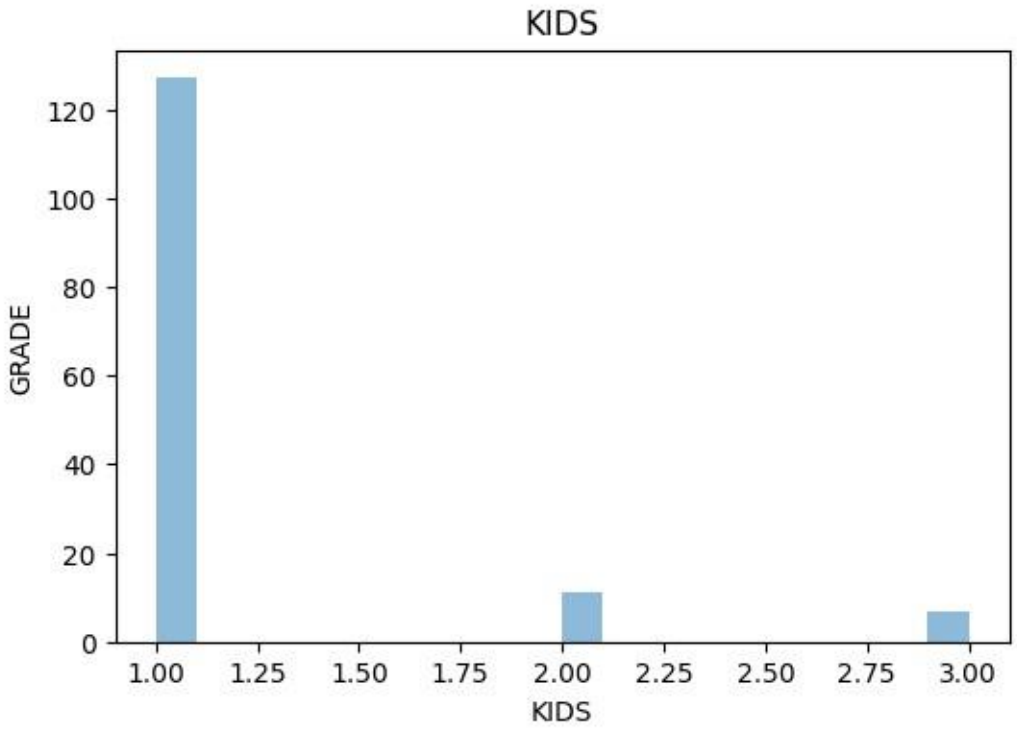

TRANSPORT
GRADE
TRANSPORT
0
20
40
60
80
100
1.0
1.5
2.0
2.5
3.0
3.5
4.0

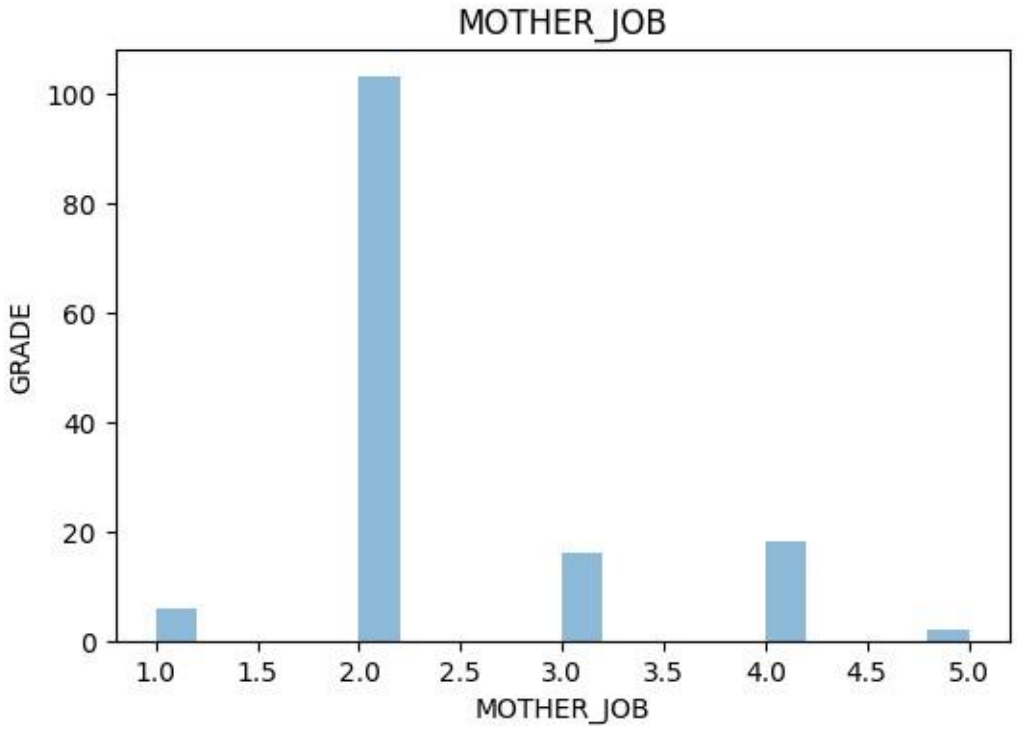

LIVING
GRADE
LIVING
0
10
20
30
40
50
60
70
1.0
1.5
2.0
2.5
3.0
3.5
4.0

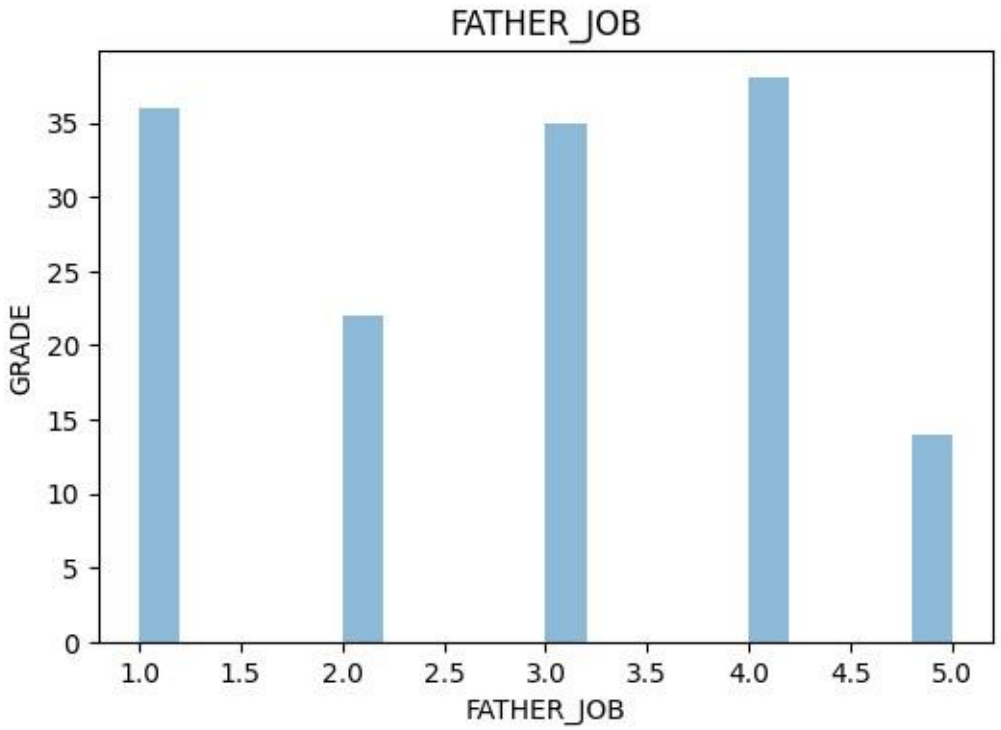

MOTHER_EDU
GRADE
0
10
20
30
40
50
1
2
3
4
5
6
MOTHER_EDU

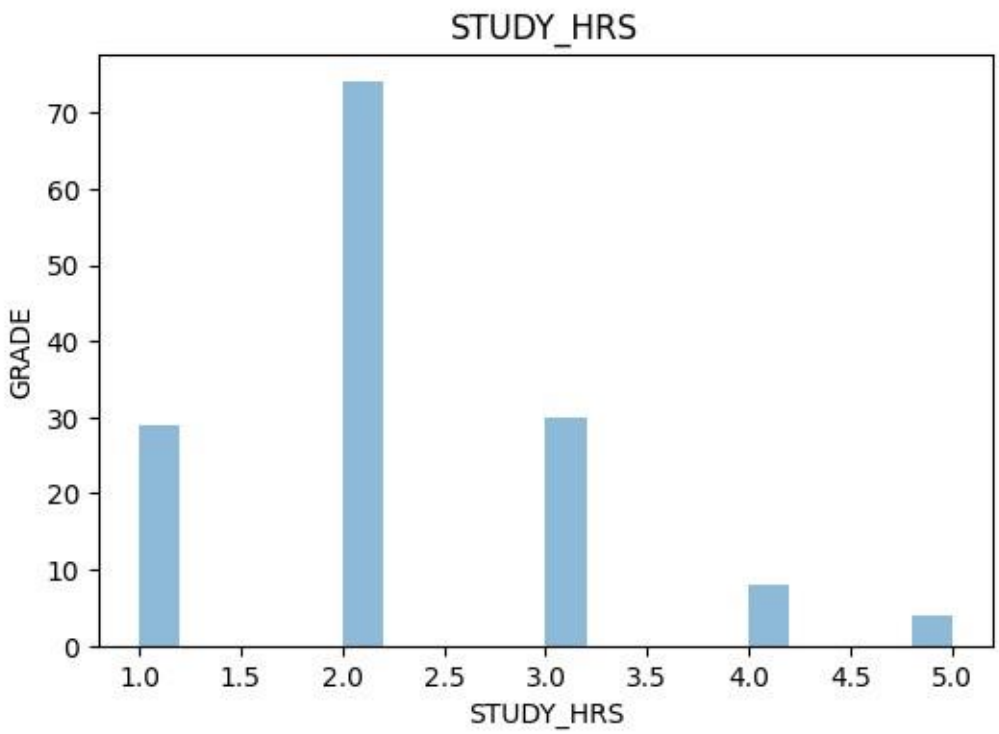

FATHER_EDU
GRADE
0
10
20
30
40
1
2
3
4
5
6
FATHER_EDU

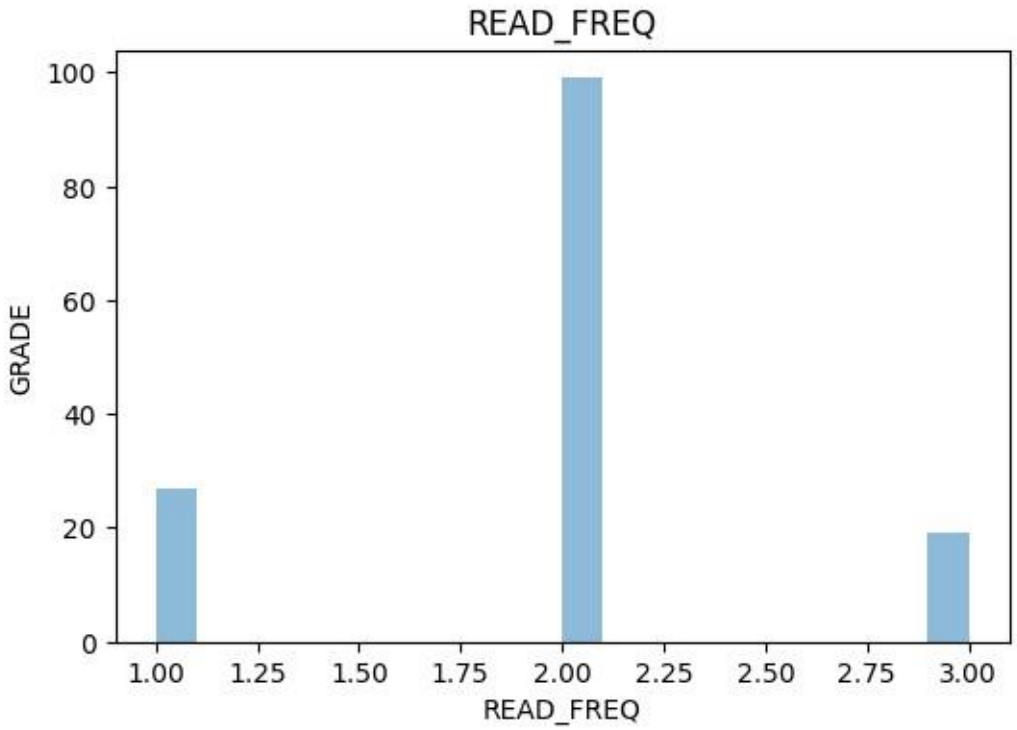

#_SIBLINGS
GRADE
0
10
20
30
40
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
#_SIBLINGS

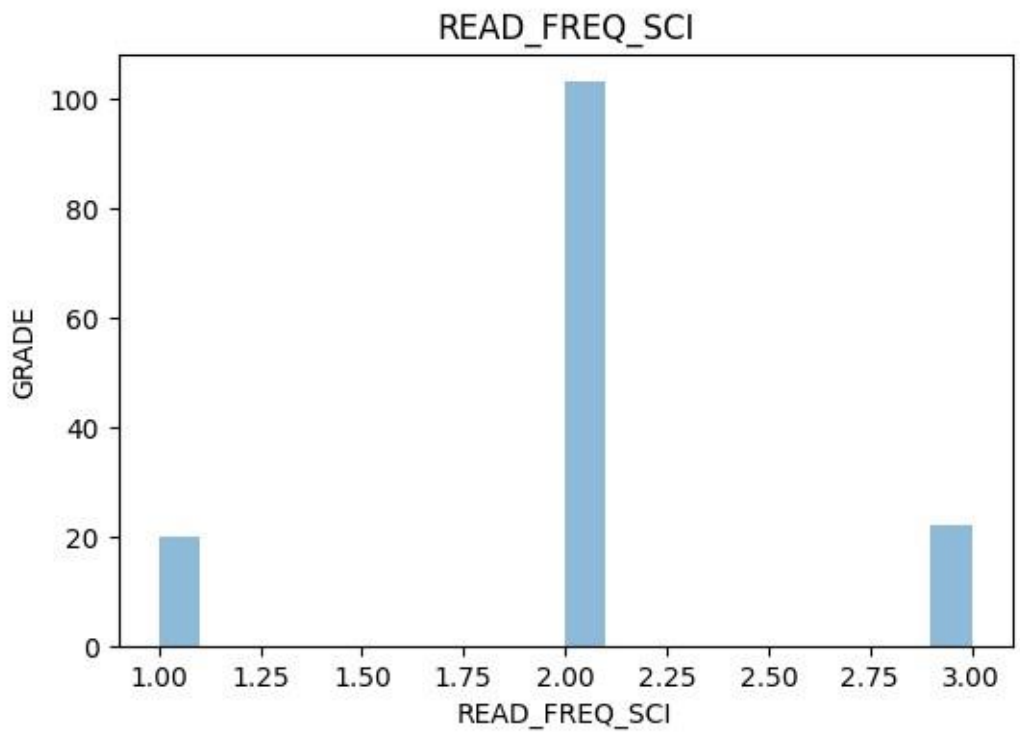

KIDS
GRADE
0
20
40
60
80
100
120
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
KIDS

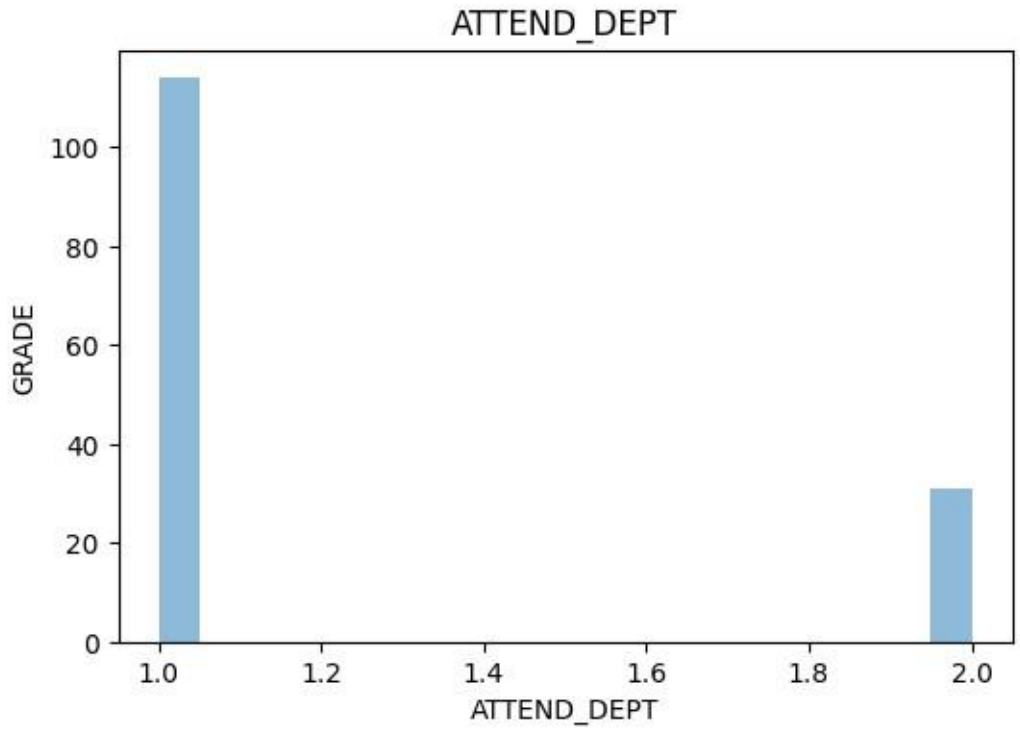

MOTHER_JOB
GRADE
MOTHER_JOB
0
20
40
60
80
100
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

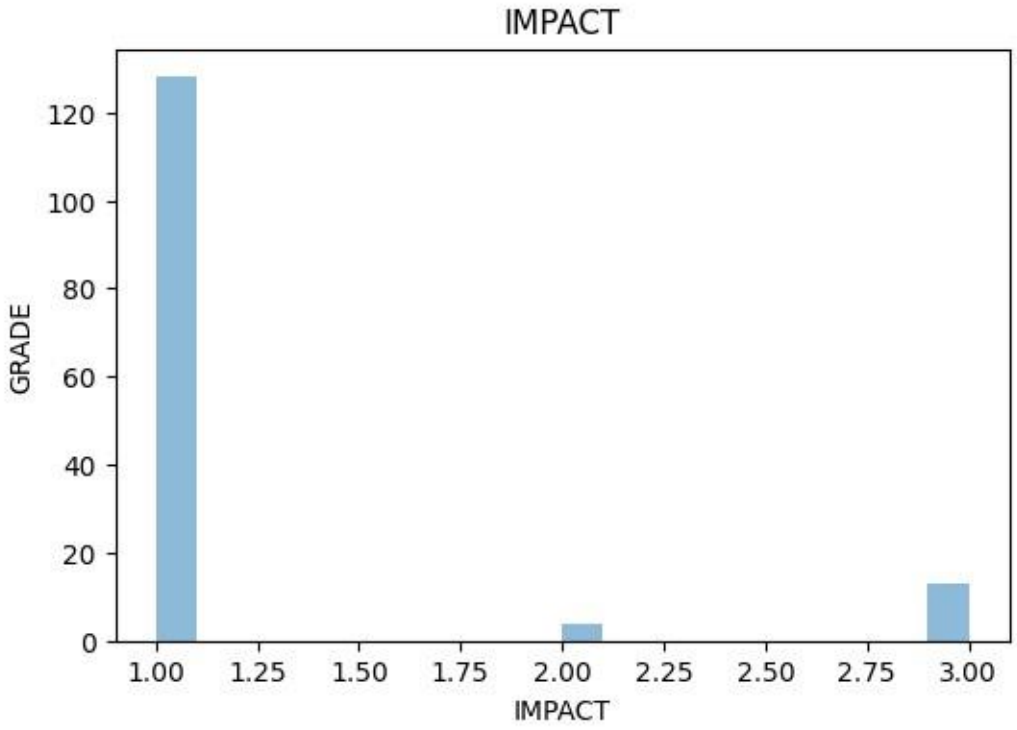

FATHER_JOB
GRADE
FATHER_JOB
0
5
10
15
20
25
30
35
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

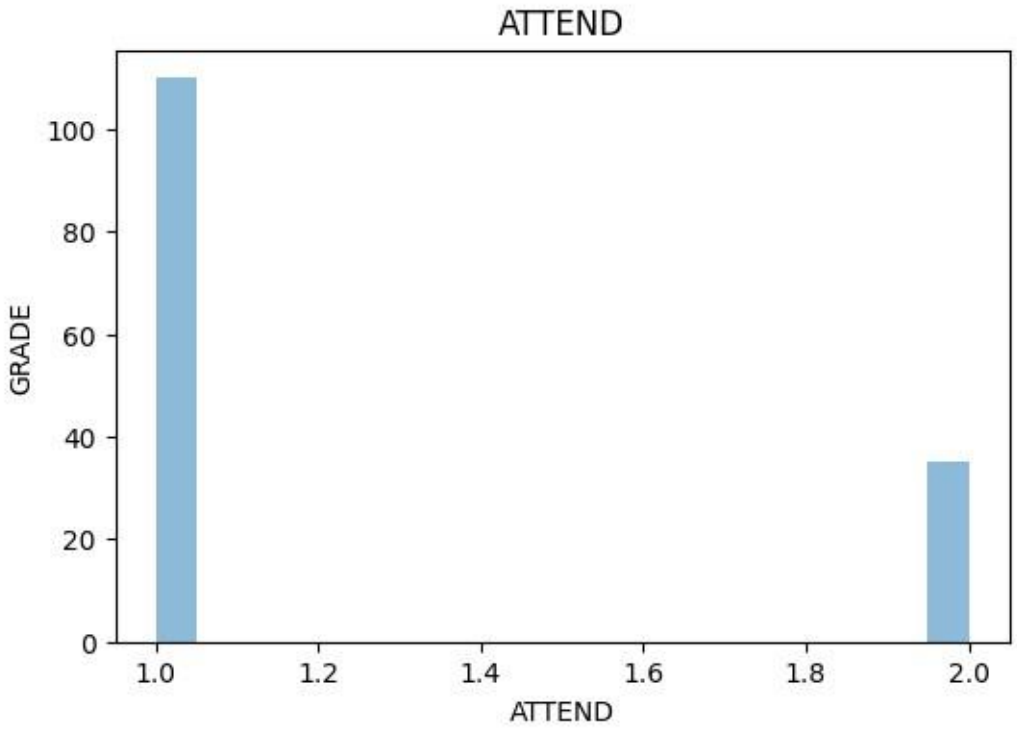

STUDY_HRS
GRADE
0
10
20
30
40
50
60
70
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
STUDY_HRS

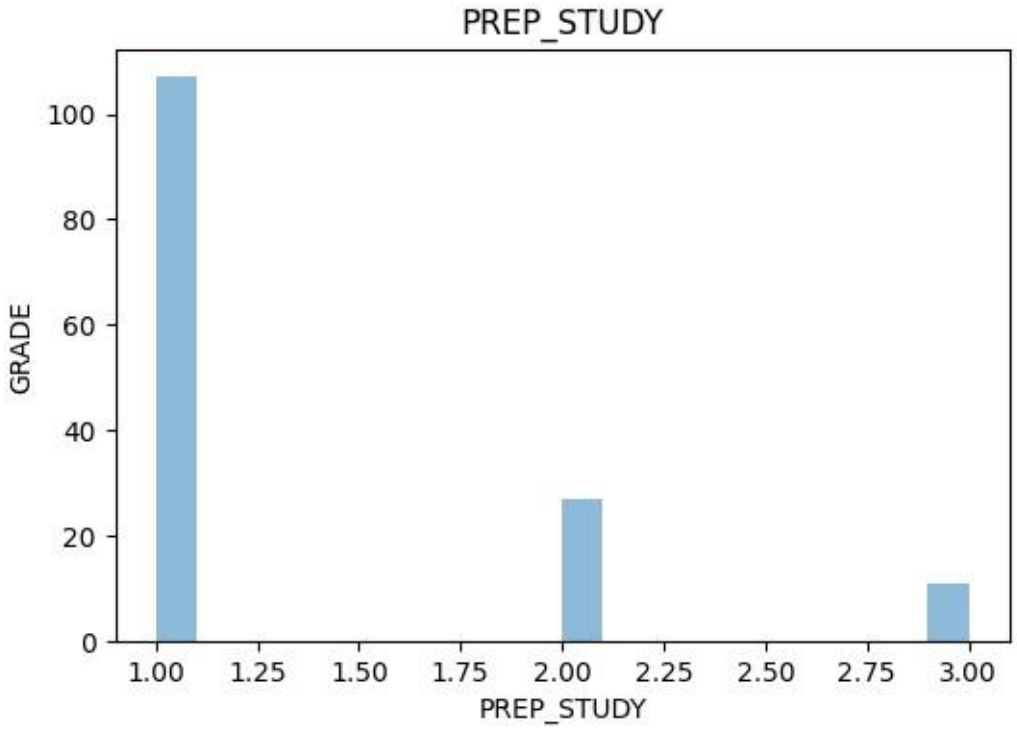

READ_FREQ
GRADE
0
20
40
60
80
100
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
READ_FREQ

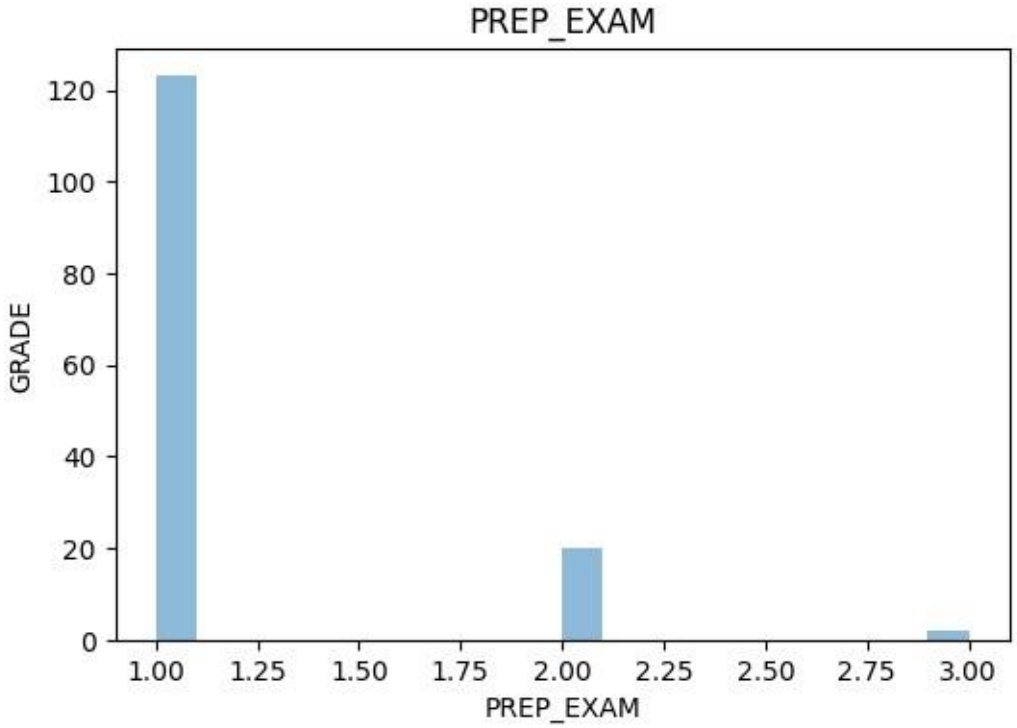

READ_FREQ_SCI
GRADE
0
20
40
60
80
100
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
READ_FREQ_SCI

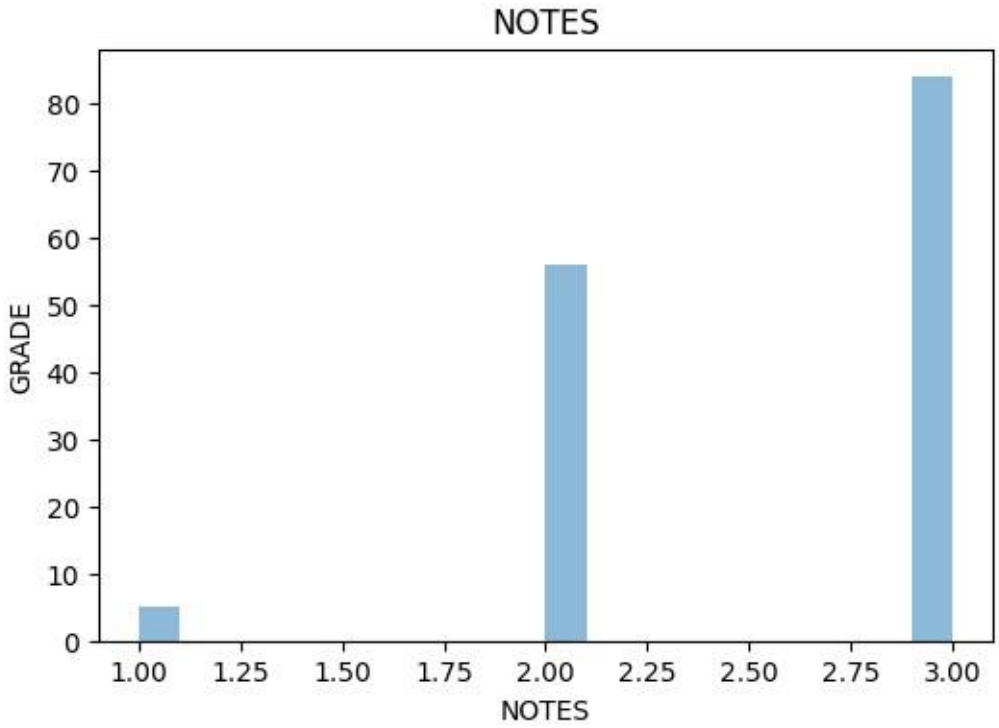

ATTEND_DEPT
GRADE
0
20
40
60
80
100
1.0
1.2
1.4
1.6
1.8
2.0
ATTEND_DEPT

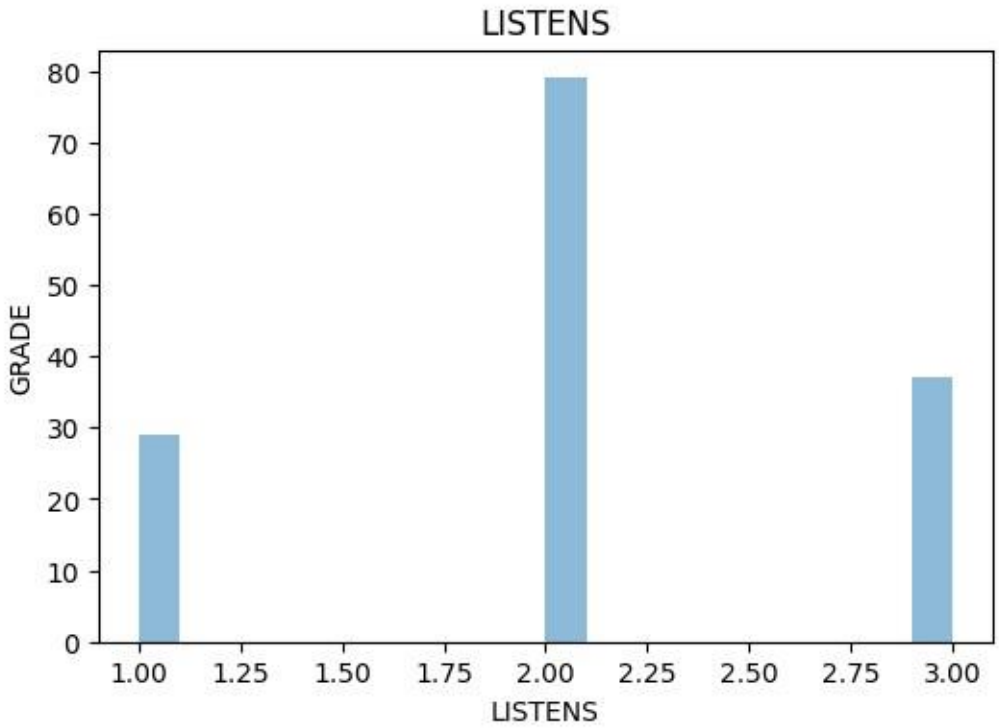

IMPACT
GRADE
120
100
80
60
40
20
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
IMPACT

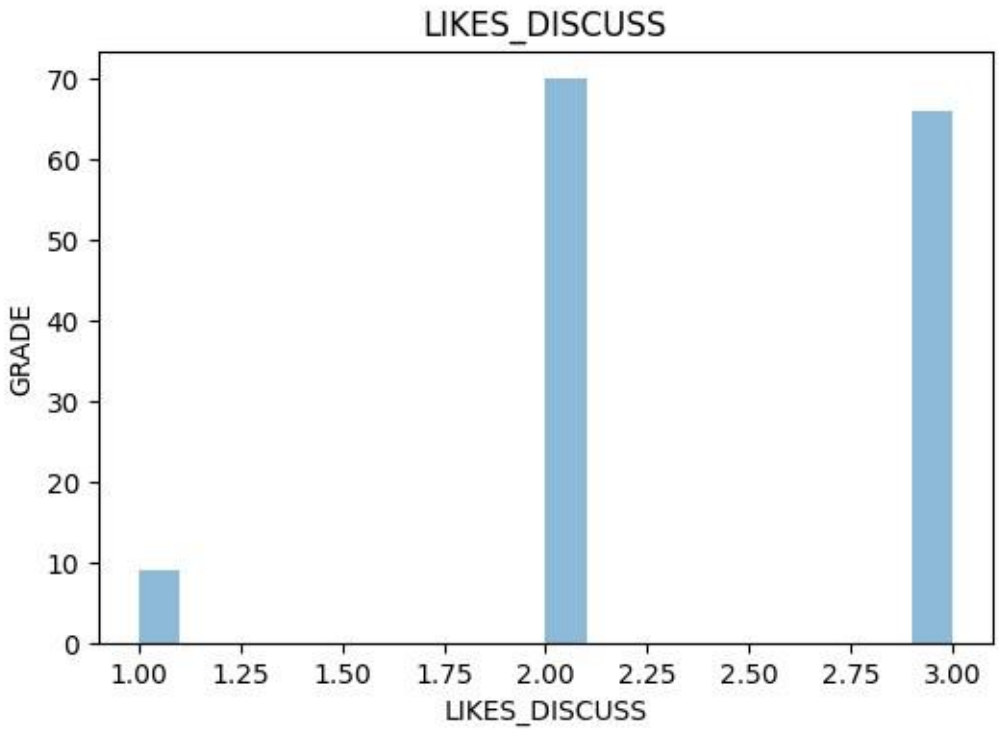

ATTEND
GRADE
100
80
60
40
20
0
1.0
1.2
1.4
1.6
1.8
2.0
ATTEND

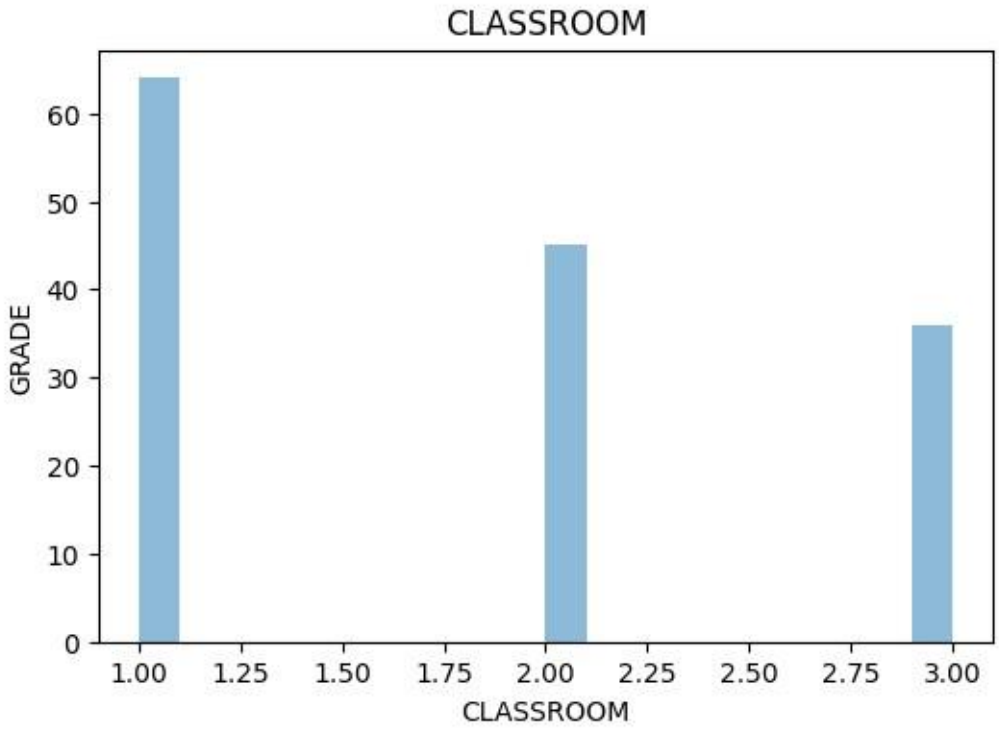

PREP_STUDY
GRADE
0
20
40
60
80
100
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
PREP_STUDY

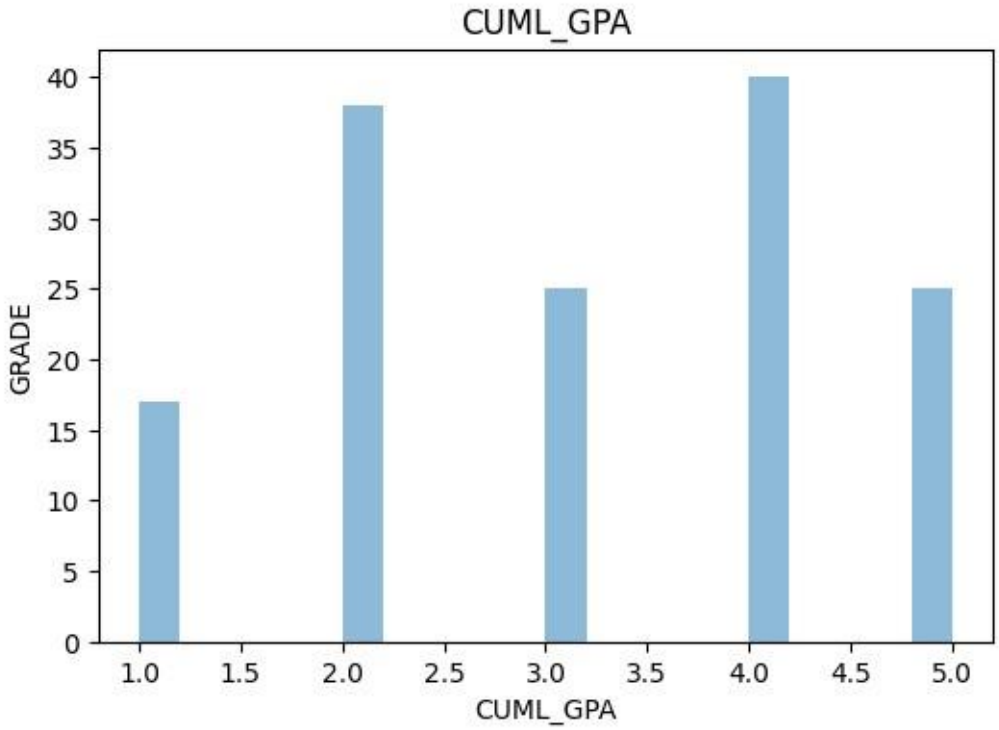

PREP_EXAM
GRADE
0
20
40
60
80
100
120
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
PREP_EXAM

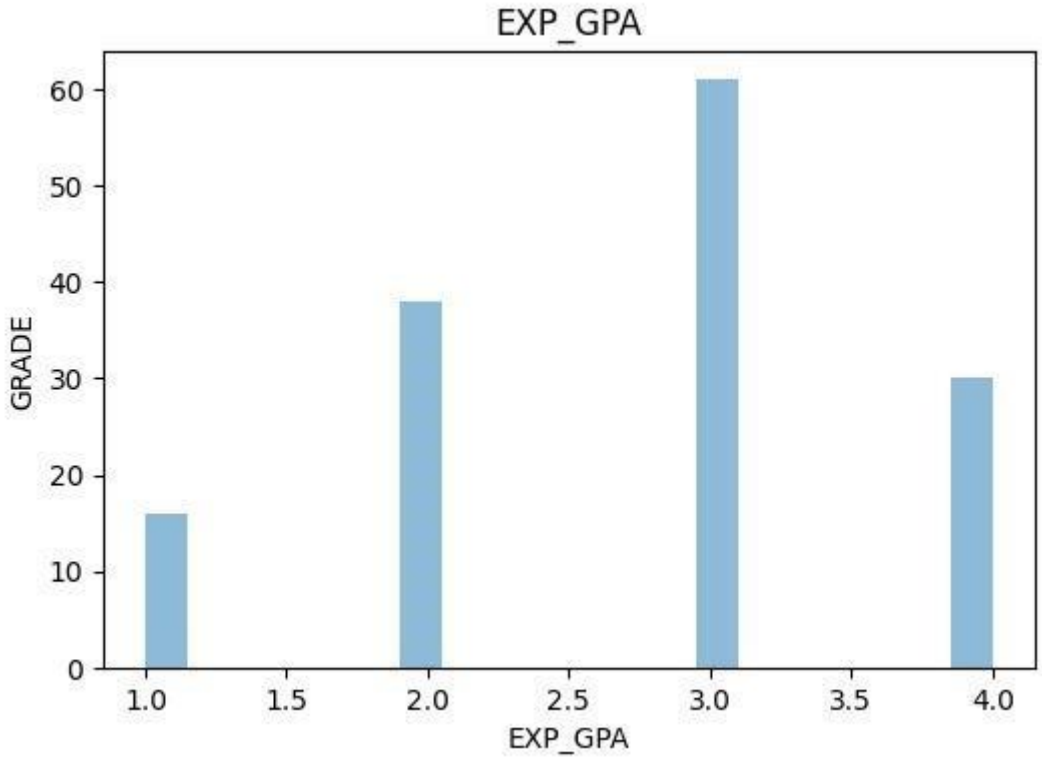

NOTES
GRADE
0
10
20
30
40
50
60
70
80
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
NOTES

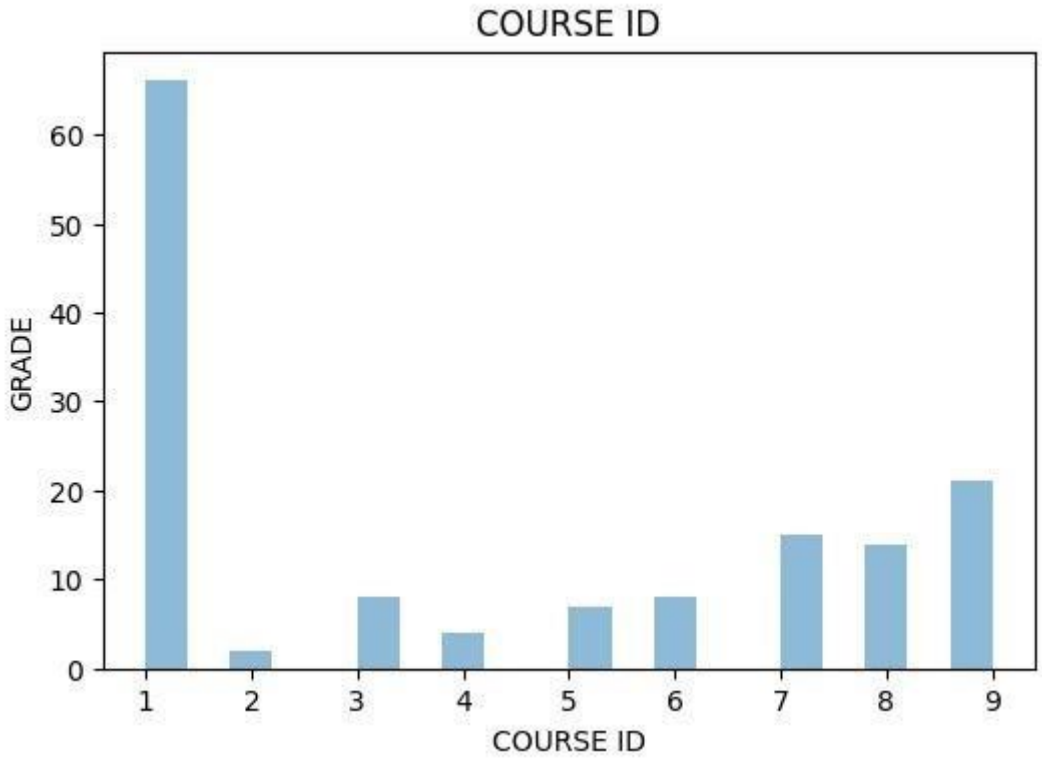

LISTENS
GRADE
0
10
20
30
40
50
60
70
80
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
LISTENS

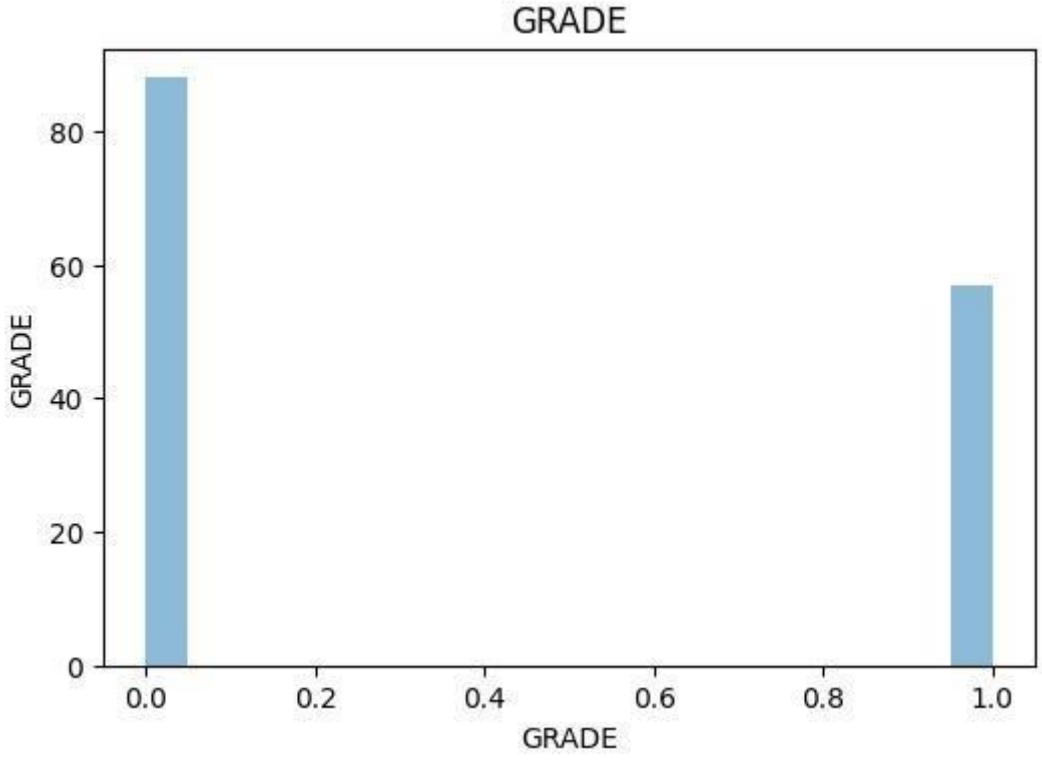

LIKES_DISCUSS
GRADE
70
60
50
40
30
20
10
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
LIKES_DISCUSS


CLASSROOM
GRADE
60
50
40
30
20
10
0
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00
CLASSROOM


CUML_GPA
GRADE
0
5
10
15
20
25
30
35
40
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
CUML_GPA


EXP_GPA
GRADE
0
10
20
30
40
50
60
1.0
1.5
2.0
2.5
3.0
3.5
4.0
EXP_GPA


COURSE ID
GRADE
0
10
20
30
40
50
60
1
2
3
4
5
6
7
8
9
COURSE ID


GRADE
GRADE
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
GRADE

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(X_train.shape)
print(X_test.shape)
print(X_test)
print(y_test)
print(y_train)
```

```
(101, 31)
(44, 31)
     AGE  GENDER  HS_TYPE  SCHOLARSHIP  WORK  ACTIVITY  PARTNER  SALARY  \
22     2       2        2            3     1         2        1       1
86     2       2        2            4     2         2        2       1
75     1       2        2            4     2         1        2       1
56     2       2        2            3     2         1        2       1
127    1       1        2            4     2         2        2       1
91     2       2        2            5     1         1        1       1
15     2       2        2            3     2         2        2       1
130    1       1        2            3     1         1        2       1
61     1       2        3            3     2         1        2       2
48     1       2        2            3     2         1        1       1
55     3       2        2            3     1         2        1       4
134    1       1        2            4     2         1        2       1
124    1       1        2            4     1         1        1       1
103    1       2        1            4     2         2        1       5
92     1       2        2            3     2         2        2       1
142    1       1        1            4     2         2        2       1
85     1       2        2            5     2         2        2       2
131    1       1        1            5     2         1        2       1
33     2       1        2            3     1         2        1       1
63     2       2        2            4     2         2        1       2
17     2       2        2            3     2         2        2       1
43     1       2        2            3     2         2        1       1
62     2       2        2            3     2         2        2       1
36     2       2        3            4     1         2        1       3
26     2       2        2            3     2         1        1       1
67     2       2        3            3     1         1        2       3
93     1       2        2            1     1         2        1       1
139    1       1        2            4     1         2        1       2
54     2       2        2            3     2         2        2       3
73     2       2        2            4     2         2        2       1
70     1       2        2            4     2         1        1       1
27     1       2        1            3     1         2        2       1
120    2       1        1            3     1         1        1       2
144    1       1        1            5     2         2        2       3
141    1       1        2            4     2         2        2       1
38     2       2        2            5     2         2        2       1
6      1       2        2            4     2         2        2       1
60     2       1        2            3     2         2        2       5
```

```
68     2       1       2            4     1       2       2       1
9      2       1       2            3     2       2       1       3
125    1       1       2            5     1       1       2       1
65     1       2       2            3     2       2       1       3
88     1       2       2            4     2       1       1       1
41     3       2       2            3     1       2       2       2

     TRANSPORT  LIVING  ...  ATTEND  PREP_STUDY  PREP_EXAM  NOTES  LISTENS  \
22           1       1  ...       1           1          2      3        1
86           4       2  ...       1           1          1      3        2
75           1       3  ...       1           1          1      2        2
56           1       1  ...       1           1          2      3        2
127          4       3  ...       1           1          1      3        2
91           1       2  ...       1           1          1      2        3
15           1       2  ...       2           1          1      3        2
130          1       1  ...       1           1          2      3        1
61           1       2  ...       2           1          1      3        2
48           1       2  ...       1           1          1      3        2
55           1       2  ...       1           3          3      1        3
134          2       3  ...       1           1          1      2        1
124          1       3  ...       1           2          2      3        3
103          1       2  ...       1           1          1      2        2
92           1       1  ...       1           1          1      3        2
142          1       1  ...       1           1          1      3        3
85           1       1  ...       1           1          1      3        2
131          2       2  ...       1           1          2      3        2
33           1       1  ...       1           1          1      1        3
63           1       1  ...       1           3          1      2        2
17           1       1  ...       2           1          1      2        2
43           1       1  ...       2           1          1      3        1
62           4       2  ...       1           1          1      2        3
36           1       1  ...       2           2          1      2        1
26           1       1  ...       1           1          1      3        3
67           1       2  ...       2           1          1      2        2
93           1       2  ...       1           1          1      3        3
139          2       3  ...       2           2          1      2        3
54           4       2  ...       1           1          2      3        1
73           1       2  ...       1           1          2      3        2
70           1       1  ...       1           1          1      2        2
27           1       1  ...       2           1          1      3        1
120          2       3  ...       2           2          1      3        3
```

144	1	1	...	1	2	1	3	2
141	4	2	...	2	1	1	3	2
38	1	1	...	1	2	1	2	2
6	1	3	...	2	1	1	3	3
60	2	1	...	1	1	1	1	3
68	1	1	...	2	1	1	2	2
9	4	2	...	2	1	1	2	2
125	1	3	...	1	1	2	3	2
65	1	1	...	2	2	1	2	1
88	1	1	...	1	1	2	3	1
41	1	2	...	2	1	1	3	2

	LIKES_DISCUSS	CLASSROOM	CUML_GPA	EXP_GPA	COURSE ID
22	2	3	3	3	1
86	3	2	5	4	5
75	2	2	4	1	3
56	3	3	5	4	1
127	2	1	2	2	9
91	3	2	5	4	6
15	2	3	2	2	1
130	3	2	2	3	9
61	2	2	4	4	1
48	3	2	2	2	1
55	3	3	5	4	1
134	2	1	2	3	9
124	2	1	3	3	9
103	3	2	4	4	7
92	3	3	2	2	6
142	2	1	4	3	9
85	3	3	4	3	5
131	3	1	5	3	9
33	2	2	2	3	1
63	2	1	3	3	1
17	2	2	2	2	1
43	3	3	3	2	1
62	3	2	5	4	1
36	2	1	4	3	1
26	3	2	2	1	1
67	1	2	2	3	2
93	3	2	2	2	6
139	2	1	1	2	9

```
54              3          1          5          3          1
73              3          1          5          3          3
70              3          1          5          4          3
27              2          1          2          1          1
120             3          2          2          2          8
144             3          1          5          4          9
141             2          1          5          3          9
38              2          2          4          3          1
6               3          3          4          4          1
60              3          1          2          1          1
68              3          2          4          3          3
9               2          2          1          2          1
125             3          1          1          3          9
65              2          1          2          2          1
88              2          3          5          3          6
41              2          3          4          3          1

[44 rows x 31 columns]
22     0
86     1
75     1
56     1
127    0
91     1
15     0
130    0
61     1
48     0
55     0
134    0
124    0
103    1
92     1
142    0
85     1
131    1
33     0
63     1
17     0
43     1
62     0
```

```
36      0
26      0
67      0
93      1
139     0
54      0
73      1
70      1
27      0
120     0
144     0
141     1
38      0
6       1
60      0
68      1
9       0
125     0
65      0
88      1
41      0
Name: GRADE, dtype: int64
105     1
71      1
16      0
52      0
106     1
       ..
126     0
136     0
18      0
40      0
89      1
Name: GRADE, Length: 101, dtype: int64
```

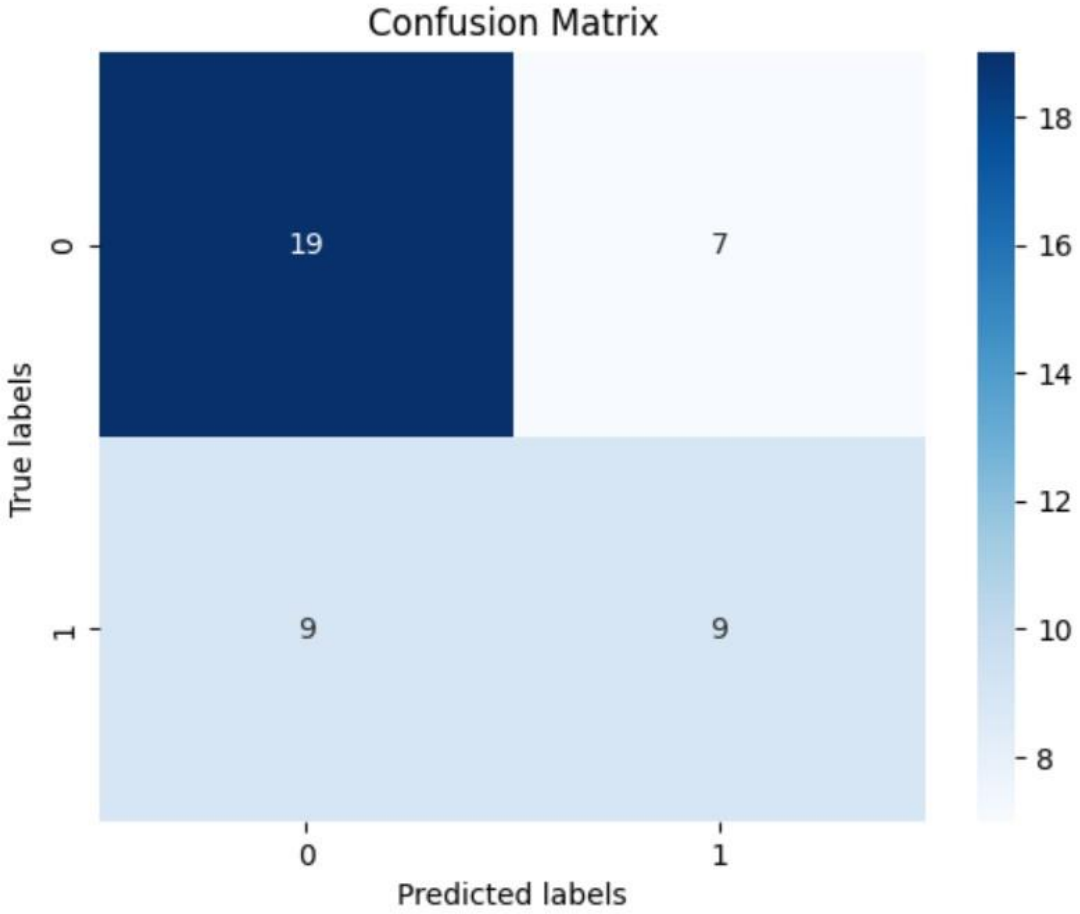
```
[10] # naive bayes #
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.naive_bayes import GaussianNB
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
# Standardizing the features
# Although normalization is not strictly required for Naive Bayes, it can be beneficial, especially for Gaussian Naive Bayes
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Initialize and train the Gaussian Naive Bayes model
gnb = GaussianNB()
gnb.fit(X_train,y_train)
```

```
GaussianNB()
```

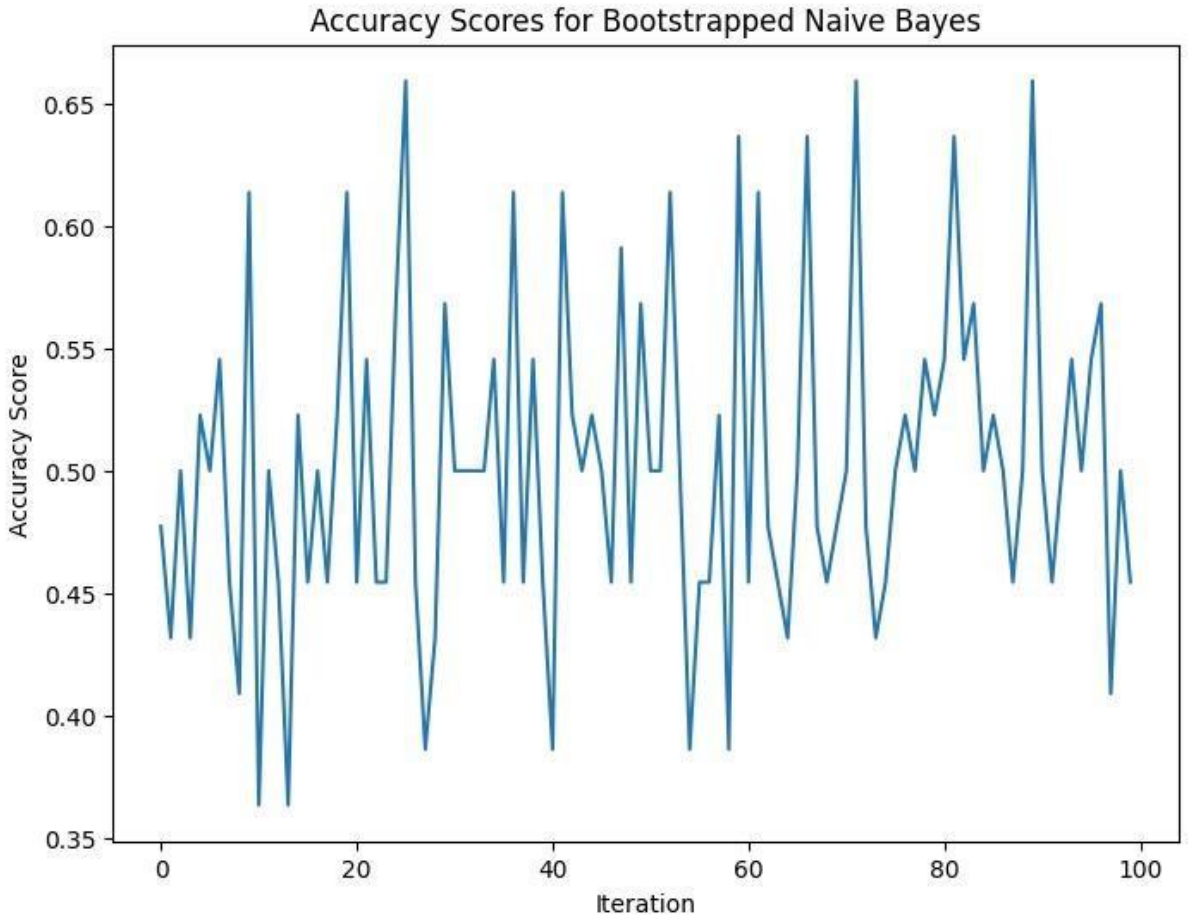
```
# Predicting the test set results
y_pred = gnb.predict(X_test)

# Evaluating the model
accuracy_NB = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_NB)
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
import seaborn as sns
import matplotlib.pyplot as plt

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')

# Add labels and title
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
```

```
Accuracy: 0.6363636363636364
Confusion Matrix:
[[19  7]
 [ 9  9]]
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.68      0.73      0.70        26
           1       0.56      0.50      0.53        18

    accuracy                           0.64        44
   macro avg       0.62      0.62      0.62        44
weighted avg       0.63      0.64      0.63        44
```

```python
from sklearn.naive_bayes import GaussianNB
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt

n_bootstrap_samples = 50
n_iterations = 100

# Create a Naive Bayes model
model = GaussianNB()

accuracy_scores = []

for _ in range(n_iterations):
    # Resample the training data
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=n_bootstrap_samples)

    # Fit the Naive Bayes model to resampled data
    model.fit(X_resampled, y_resampled)

    # Predict using the Naive Bayes model on the test data
    y_pred = model.predict(X_test)
```



```
    # Calculate accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

plt.figure(figsize=(8, 6))
plt.plot(range(n_iterations), accuracy_scores)
plt.xlabel("Iteration")
plt.ylabel("Accuracy Score")
plt.title("Accuracy Scores for Bootstrapped Naive Bayes")
plt.show()
```

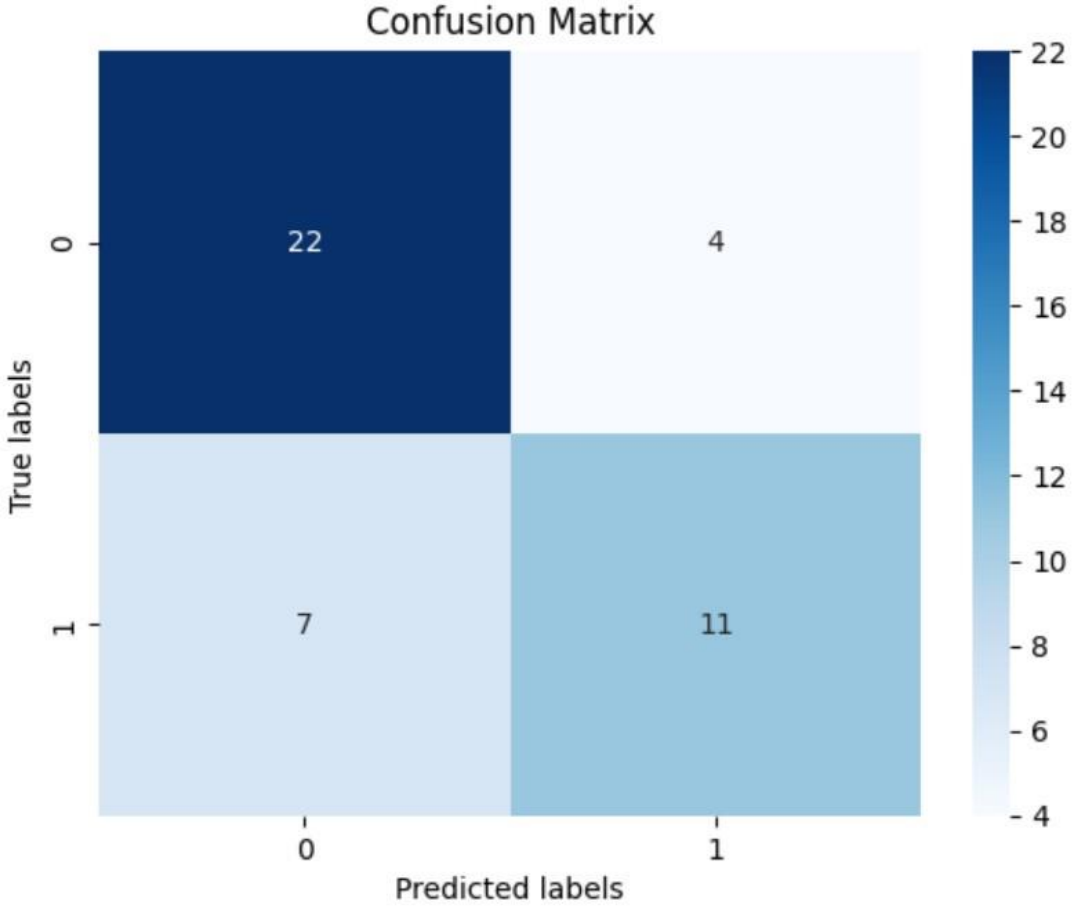
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
[15] scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_train)
     X_test_scaled = scaler.transform(X_test)
     # Initialize the Decision Tree Classifier
     tree_model = DecisionTreeClassifier(random_state=42)

     # Train the model
     tree_model.fit(X_train_scaled, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
# Predict on the test set
y_pred = tree_model.predict(X_test_scaled)

# Evaluation metrics
accuracy_DT= accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_DT)
print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
import seaborn as sns
import matplotlib.pyplot as plt

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')

# Add labels and title
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
```

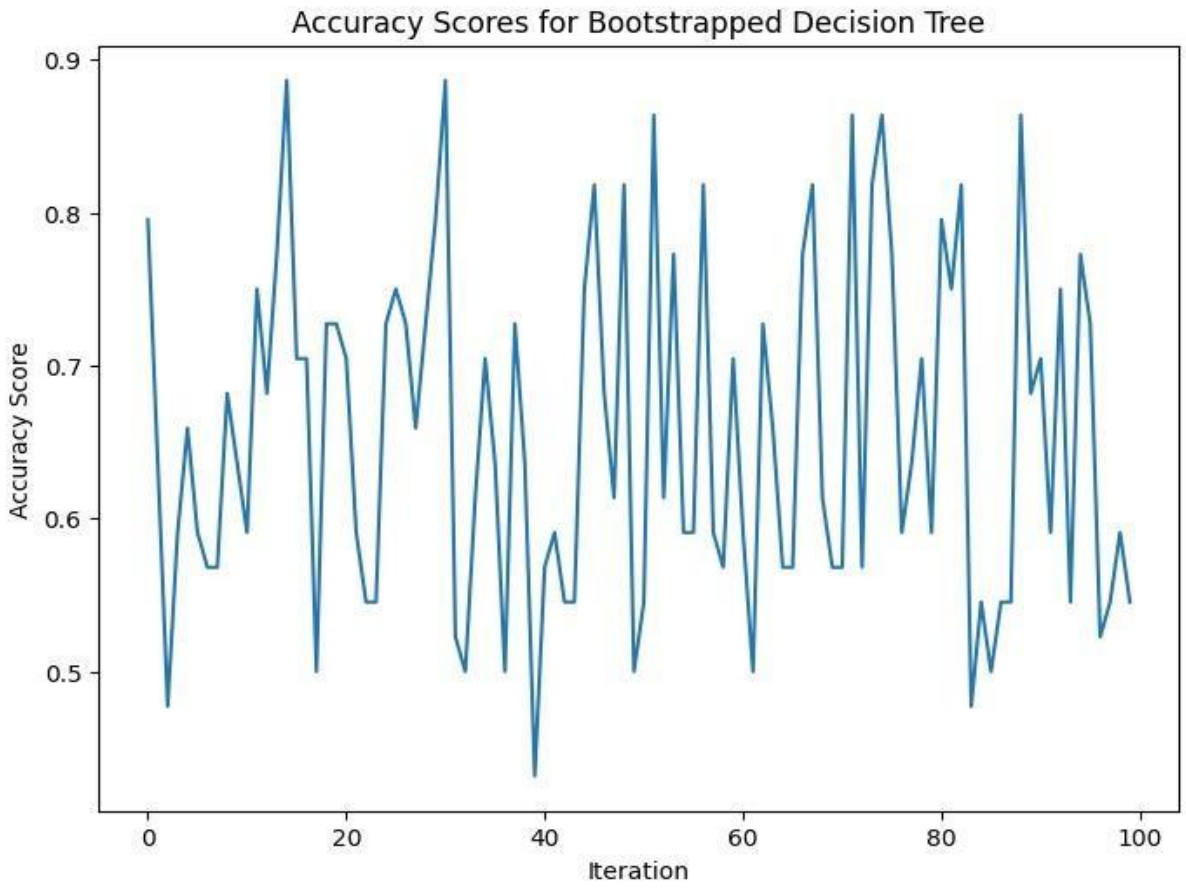
```
Accuracy: 0.75
Accuracy: 0.75
Confusion Matrix:
[[22  4]
 [ 7 11]]
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.85      0.80        26
           1       0.73      0.61      0.67        18

    accuracy                           0.75        44
   macro avg       0.75      0.73      0.73        44
weighted avg       0.75      0.75      0.75        44
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt

n_bootstrap_samples = 50
n_iterations = 100

# Create a Decision Tree model
model = DecisionTreeClassifier()

accuracy_scores = []

for _ in range(n_iterations):
    # Resample the training data
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=n_bootstrap_samples)

    # Fit the Decision Tree model to resampled data
    model.fit(X_resampled, y_resampled)

    # Predict using the Decision Tree model on the test data
    y_pred = model.predict(X_test)
```

```
    # Calculate accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

plt.figure(figsize=(8, 6))
plt.plot(range(n_iterations), accuracy_scores)
plt.xlabel("Iteration")
plt.ylabel("Accuracy Score")
plt.title("Accuracy Scores for Bootstrapped Decision Tree")
plt.show()
```



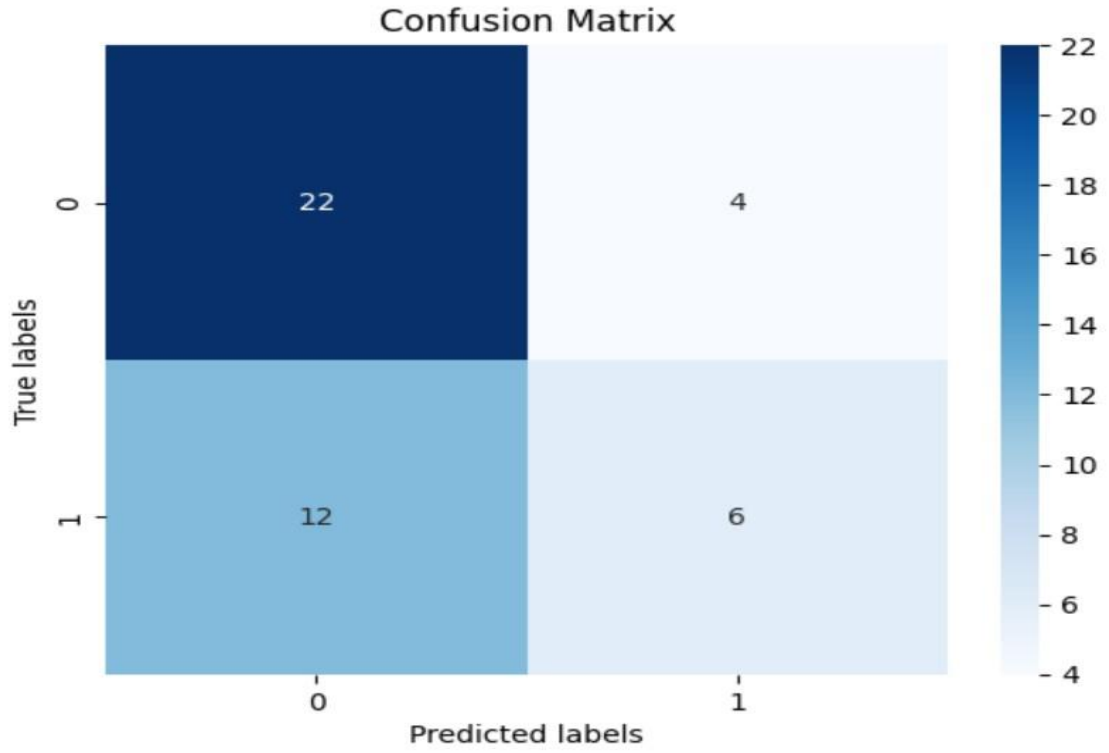

```
[18] #multi-layer perceptrons#
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.neural_network import MLPClassifier
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
# Standardizing the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Initialize the MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(100,), activation='relu', solver='adam', max_iter=300, random_state=42)

# Train the model
mlp.fit(X_train, y_train)
```

```
MLPClassifier
MLPClassifier(max_iter=300, random_state=42)
```

```
y_pred = mlp.predict(X_test)

# Evaluation
accuracy_MLP = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_MLP)
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
import seaborn as sns
import matplotlib.pyplot as plt

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')

# Add labels and title
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
```

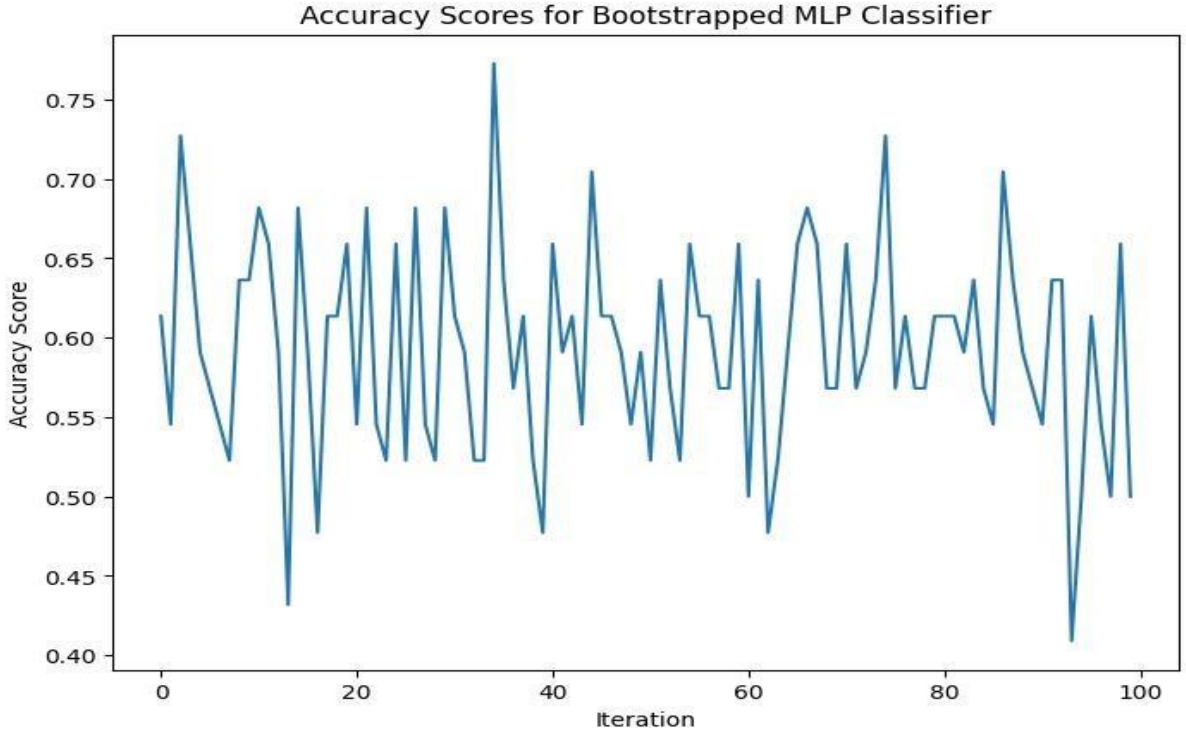
```
Accuracy: 0.6363636363636364
Confusion Matrix:
[[22  4]
 [12  6]]
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.85      0.73        26
           1       0.60      0.33      0.43        18

    accuracy                           0.64        44
   macro avg       0.62      0.59      0.58        44
weighted avg       0.63      0.64      0.61        44
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt

n_bootstrap_samples = 50
n_iterations = 100

# Create an MLP classifier model
model = MLPClassifier()

accuracy_scores = []

for _ in range(n_iterations):
    # Resample the training data
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=n_bootstrap_samples)

    # Fit the MLP classifier model to resampled data
    model.fit(X_resampled, y_resampled)

    # Predict using the MLP classifier model on the test data
    y_pred = model.predict(X_test)
```

```
    # Calculate accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

plt.figure(figsize=(8, 6))
plt.plot(range(n_iterations), accuracy_scores)
plt.xlabel("Iteration")
plt.ylabel("Accuracy Score")
plt.title("Accuracy Scores for Bootstrapped MLP Classifier")
plt.show()
```



```
[25] pip install lightgbm
```

```
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-packages (4.1.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lightgbm) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lightgbm) (1.11.4)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import lightgbm as lgb
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
# Standardizing the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

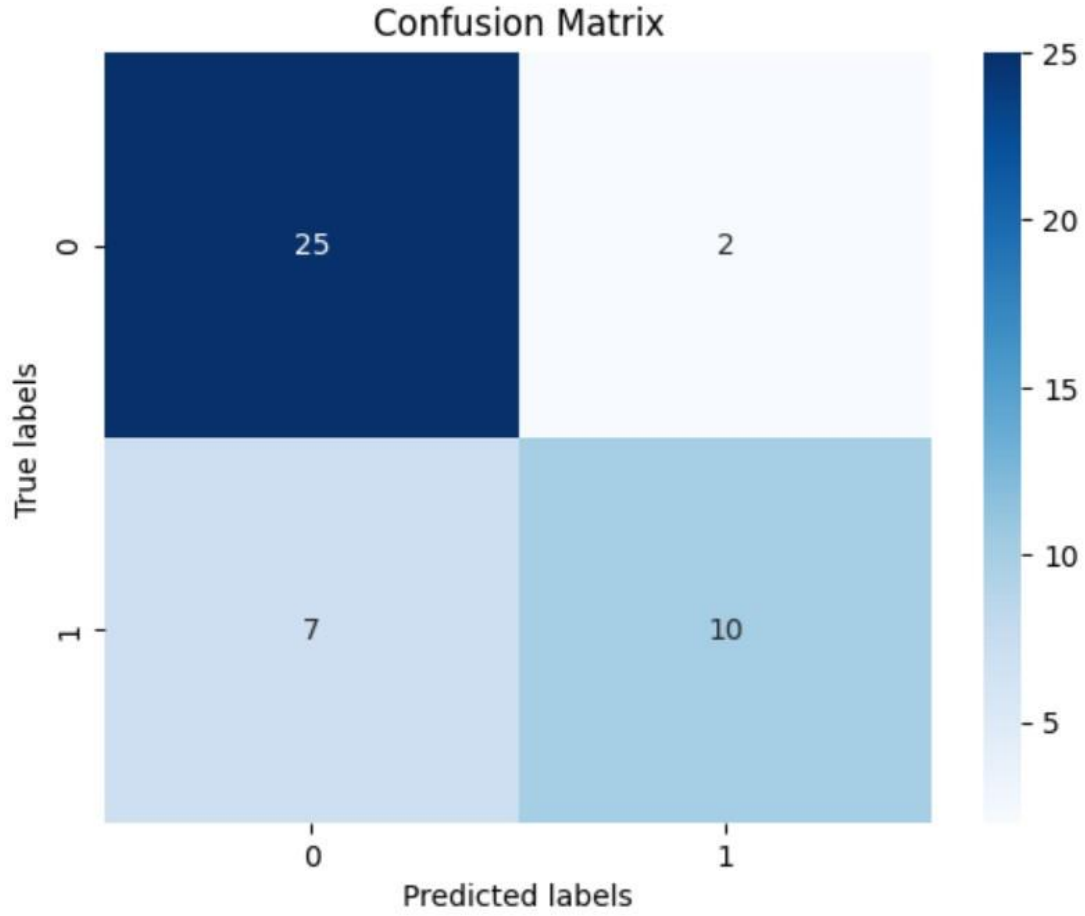
```
[24] # Creating LightGBM dataset from the training data
train_data = lgb.Dataset(X_train, label=y_train)
```

```
# Setting parameters
# Note: These are basic parameters. LightGBM offers a wide range of parameters that can be tuned for improved performance.
params = {
'objective': 'binary',
'metric': 'binary_logloss',
'boosting': 'gbdt',
'learning_rate': 0.05,
'num_leaves': 31,
'max_depth': -1,
}

# Train the model
gbm = lgb.train(params, train_data, num_boost_round=100)

# Predicting on the test set
y_pred = gbm.predict(X_test)
# Converting probabilities to class labels based on a threshold (0.5)
y_pred = [1 if prob > 0.5 else 0 for prob in y_pred]
```

```
[LightGBM] [Info] Number of positive: 38, number of negative: 63
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of testing was 0.000769 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 126
[LightGBM] [Info] Number of data points in the train set: 101, number of used features: 27
```

```
# Evaluating the model
accuracy_LGBM = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_LGBM)
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
import seaborn as sns
import matplotlib.pyplot as plt

# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')

# Add labels and title
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
```

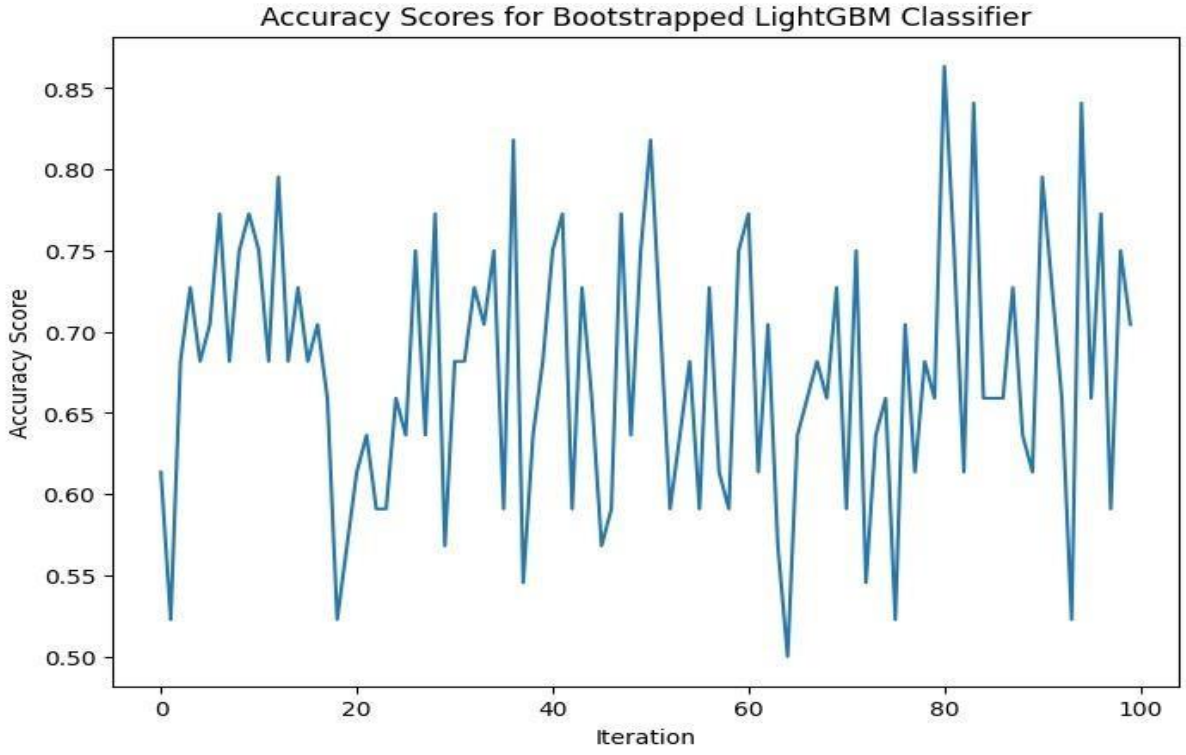
```
Accuracy: 0.7954545454545454
Confusion Matrix:
[[25  2]
 [ 7 10]]
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.93      0.85        27
           1       0.83      0.59      0.69        17

    accuracy                           0.80        44
   macro avg       0.81      0.76      0.77        44
weighted avg       0.80      0.80      0.79        44
```

```
import lightgbm as lgb
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt

n_bootstrap_samples = 50
n_iterations = 100

# Create a LightGBM classifier model
model = lgb.LGBMClassifier()

accuracy_scores = []

for _ in range(n_iterations):
    # Resample the training data
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=n_bootstrap_samples)

    # Fit the LightGBM classifier model to resampled data
    model.fit(X_resampled, y_resampled)

    # Predict using the LightGBM classifier model on the test data
    y_pred = model.predict(X_test)
```

```
    # Calculate accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

plt.figure(figsize=(8, 6))
plt.plot(range(n_iterations), accuracy_scores)
plt.xlabel("Iteration")
plt.ylabel("Accuracy Score")
plt.title("Accuracy Scores for Bootstrapped LightGBM Classifier")
plt.show()
```



```
[27] import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler
     from sklearn.linear_model import LogisticRegression
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
import numpy as np
costs_list = []
num_iter = []
# implementation of logistic regression from scratch without any libraries
class Logisticregression:
    def __init__(self, learning_rate=0.01, num_iterations=1000):
        self.learning_rate = learning_rate
        self.num_iterations = num_iterations
        self.weights = None
        self.bias = None
        self.cost_history = []
    def sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    def cost_function(self, X, y):
        m = X.shape[0]
        h = self.sigmoid(np.dot(X, self.weights) + self.bias)
        cost = (-1 / m) * np.sum(y * np.log(h) + (1 - y) * np.log(1 - h))
        return cost

    def fit(self, X, y):
        m, n = X.shape
        self.weights = np.zeros(n)
        self.bias = 0

        for i in range(self.num_iterations):
            h = self.sigmoid(np.dot(X, self.weights) + self.bias)
            dw = (1 / m) * np.dot(X.T, (h - y))
            db = (1 / m) * np.sum(h - y)
            self.weights -= self.learning_rate * dw
            self.bias -= self.learning_rate * db
            cost = self.cost_function(X, y)
            self.cost_history.append(cost)
            if i % 100 == 0:
                cost = self.cost_function(X, y)
                print(f"Cost after iteration {i}: {cost}")

        print("",self.cost_history)
```

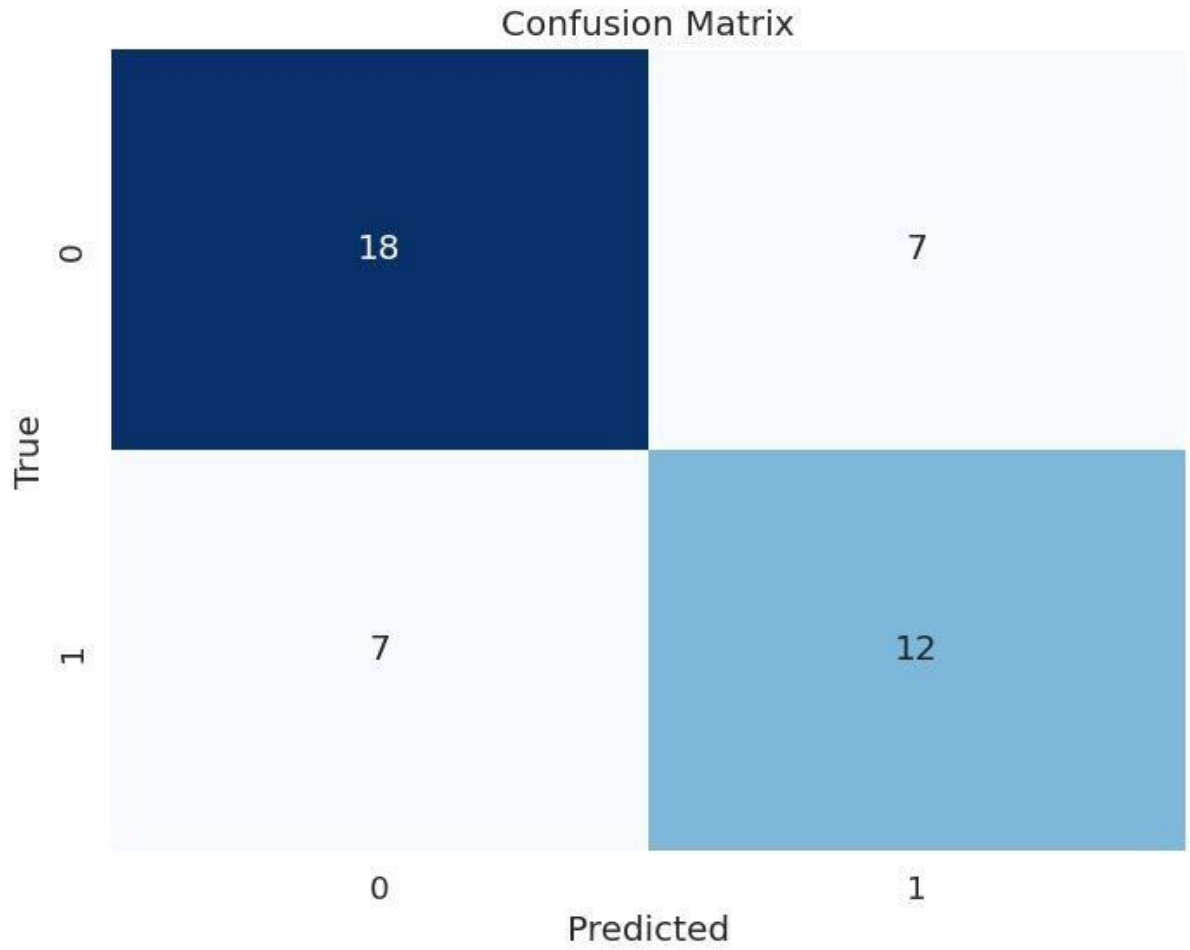
```
    def predict(self, X):
        h = self.sigmoid(np.dot(X, self.weights) + self.bias)
        y_pred = np.where(h > 0.5, 1, 0)
        return y_pred
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler

model = Logisticregression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print(y_pred)
accuracy_p = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_p)
print(classification_report(y_test, y_pred))
```

```
Cost after iteration 0: 0.6916104400662529
Cost after iteration 100: 0.5877576978837151
Cost after iteration 200: 0.5340016259100308
Cost after iteration 300: 0.5001221784891144
Cost after iteration 400: 0.4762444665618011
Cost after iteration 500: 0.4582194797090152
Cost after iteration 600: 0.4439722218534353
Cost after iteration 700: 0.4323360026980761
Cost after iteration 800: 0.4225974391671706
Cost after iteration 900: 0.414291763951613
 [0.6916104400662529, 0.6900878173303963, 0.6885791432794542, 0.687084250632029, 0.6856029739008704, 0.6841351493944086, 0.6826806152172387, 0.6812392112695956
[0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1
 0 1 0 1 0 1 1]
Accuracy: 0.6590909090909091
              precision    recall  f1-score   support

           0       0.71      0.74      0.73        27
           1       0.56      0.53      0.55        17

    accuracy                           0.66        44
   macro avg       0.64      0.64      0.64        44
weighted avg       0.66      0.66      0.66        44
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Assuming you have defined x_train, y_train, x_test, and y_test somewhere in your code
# Split your data into training and testing sets


# Standardize your features if needed
# scaler = StandardScaler()
# x_train = scaler.fit_transform(x_train)
# x_test = scaler.transform(x_test)

# Create and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

```
# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.2)  # Adjust font size for clarity
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# Evaluate the model
accuracy_LR = accuracy_score(y_test, y_pred)
print("Accuracy is:", accuracy_LR)
print(classification_report(y_test, y_pred))
```



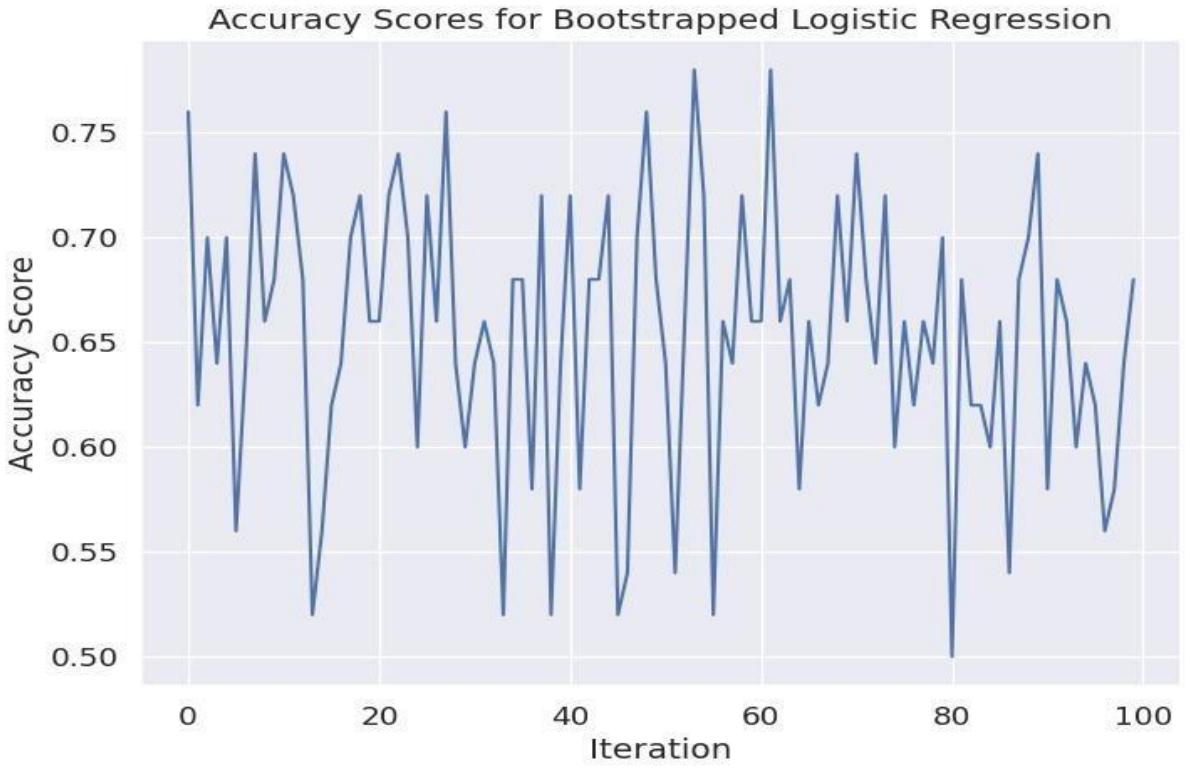
0s

Predicted

```
Accuracy is: 0.6818181818181818
              precision    recall  f1-score   support

           0       0.72      0.72      0.72        25
           1       0.63      0.63      0.63        19

    accuracy                           0.68        44
   macro avg       0.68      0.68      0.68        44
weighted avg       0.68      0.68      0.68        44
```

1s

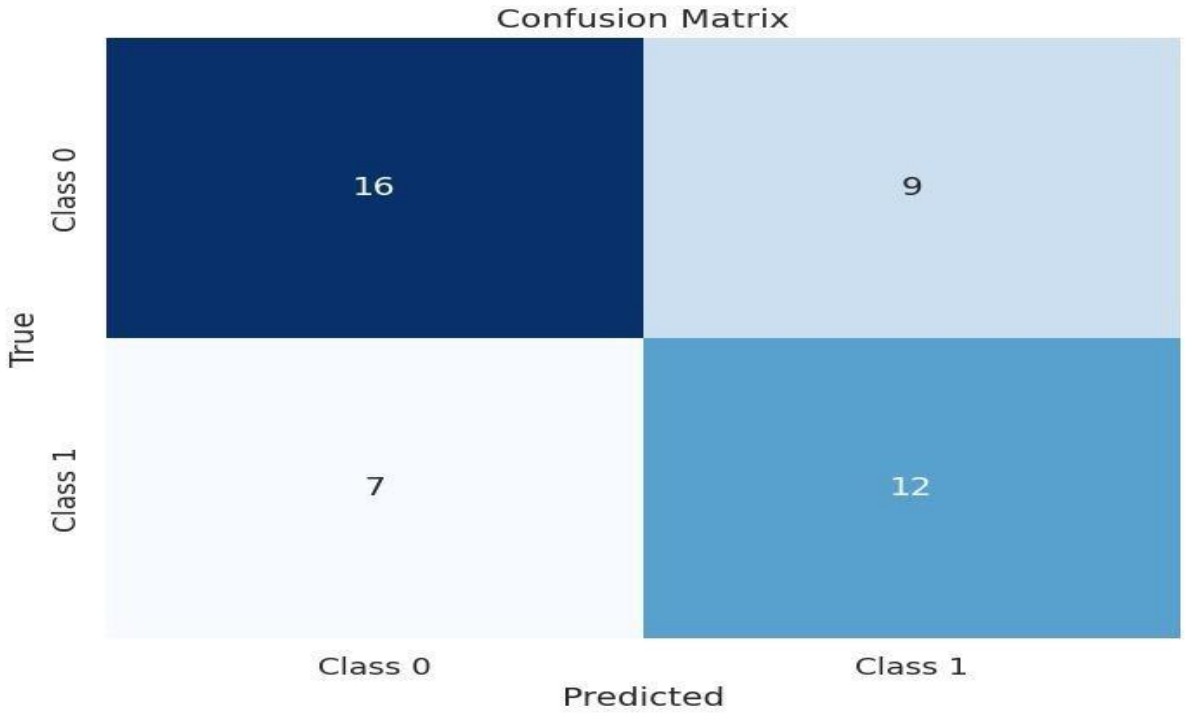
```python
from sklearn.linear_model import LogisticRegression
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import numpy as np
n_bootstrap_samples = 50
n_iterations = 100
model = LogisticRegression()
model.fit(X_test, y_test)

accuracy_scores = []
for _ in range(n_iterations):
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=n_bootstrap_samples)
    y_p = model.predict(X_resampled)
    accuracy = accuracy_score(y_resampled, y_p)
    accuracy_scores.append(accuracy)

import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
plt.plot(range(n_iterations), accuracy_scores)
plt.xlabel("Iteration")
plt.ylabel("Accuracy Score")
plt.title("Accuracy Scores for Bootstrapped Logistic Regression")
plt.show()
```



```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Assuming you have defined x_train, y_train, x_test, and y_test somewhere in your code
# Split your data into training and testing sets
# Create an SVM instance and fit the training data
svm = SVC(kernel='linear', C=1, random_state=42)
svm.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = svm.predict(X_test)

# Calculate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

✓ 1s completed at 12:42 AM

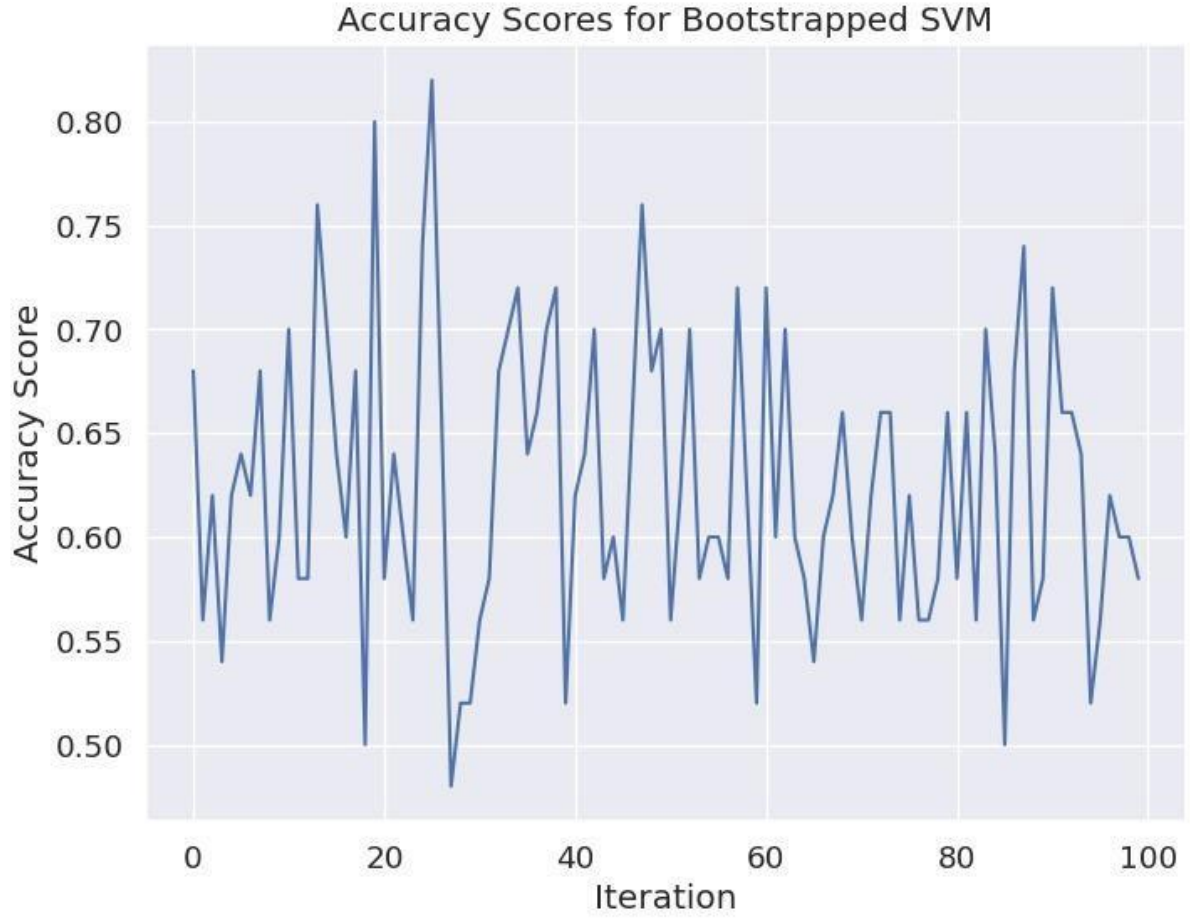
```python
# Create a heatmap for the confusion matrix
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.2)  # Adjust font size for clarity
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
            xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# Evaluate the model
accuracy_SVM = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_SVM)
print(classification_report(y_test, y_pred))
```




```
Accuracy: 0.6363636363636364
              precision    recall  f1-score   support

           0       0.70      0.64      0.67        25
           1       0.57      0.63      0.60        19

    accuracy                           0.64        44
   macro avg       0.63      0.64      0.63        44
weighted avg       0.64      0.64      0.64        44
```

```python
from sklearn.svm import SVC
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt

n_bootstrap_samples = 50
n_iterations = 100

# Create an SVM model
model = SVC()

# Fit the SVM model to your test data (x_test, y_test)
model.fit(X_test, y_test)

accuracy_scores = []

for _ in range(n_iterations):
    # Resample the training data
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=n_bootstrap_samples)

    # Predict using the SVM model
    y_p = model.predict(X_resampled)
```

```
    # Predict using the SVM model
    y_p = model.predict(X_resampled)

    # Calculate accuracy score
    accuracy = accuracy_score(y_resampled, y_p)
    accuracy_scores.append(accuracy)

plt.figure(figsize=(8, 6))
plt.plot(range(n_iterations), accuracy_scores)
plt.xlabel("Iteration")
plt.ylabel("Accuracy Score")
plt.title("Accuracy Scores for Bootstrapped SVM")
plt.show()
```




```
pip install xgboost
```

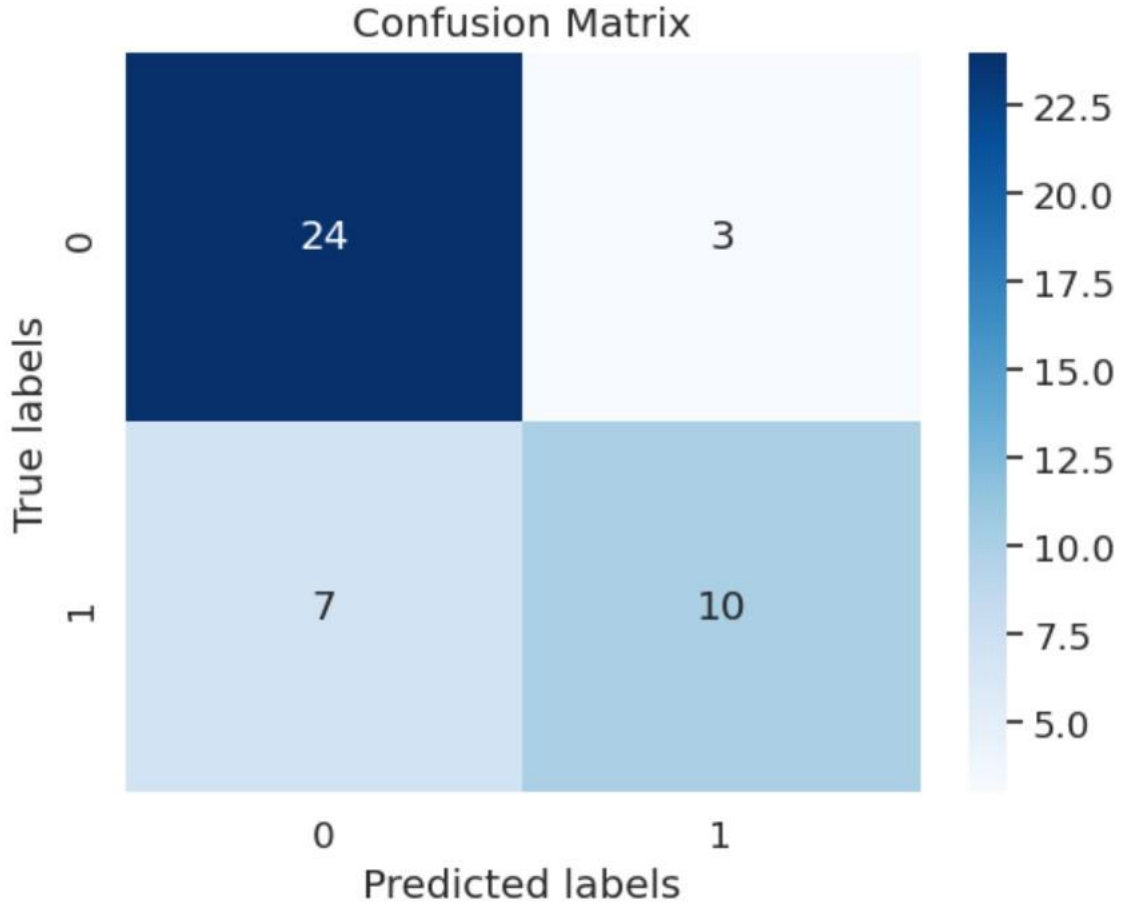
```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.11.4)
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize the XGBClassifier
model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')

# Fit the model to the training data
model.fit(X_train, y_train)
# Predicting the Test set results
y_pred = model.predict(X_test)

# Evaluating the model
accuracy_XGB = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy_XGB)
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
import seaborn as sns
import matplotlib.pyplot as plt
```

0s

```
# Compute confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Create a heatmap
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g')

# Add labels and title
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()

print(f"Classification Report:\n{classification_report(y_test, y_pred)}")
```

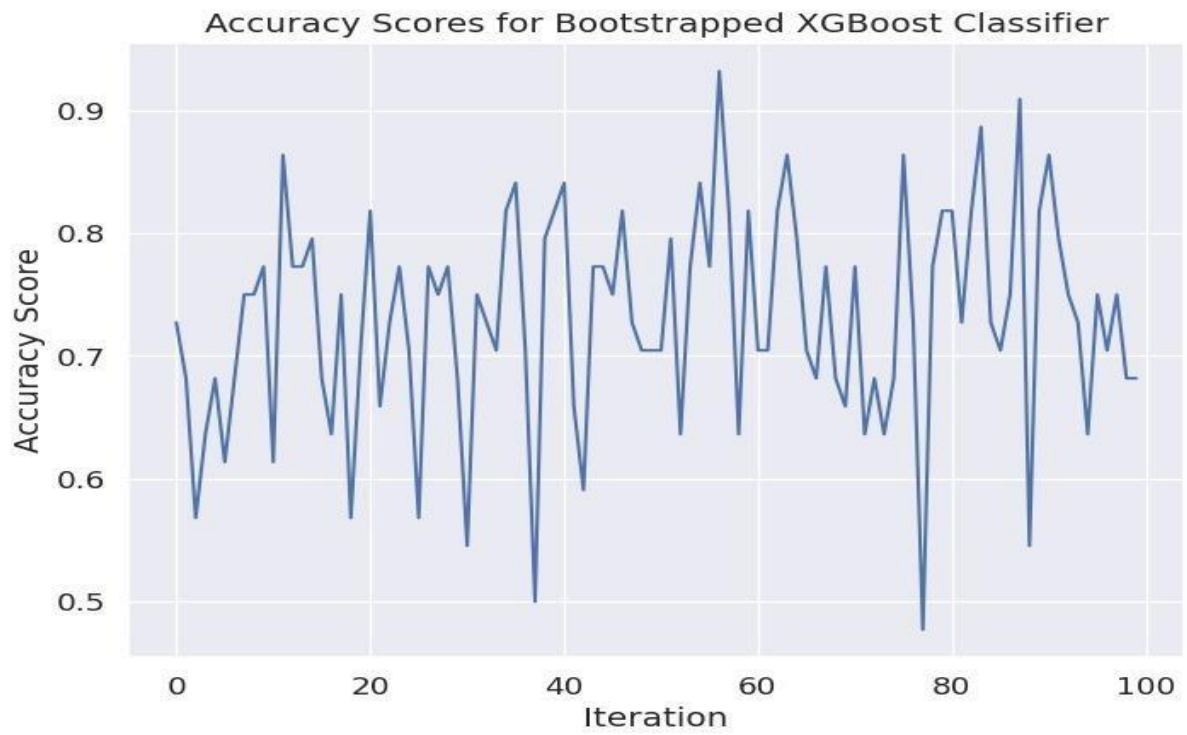
```
Accuracy: 0.7727272727272727
Confusion Matrix:
[[24  3]
 [ 7 10]]
```



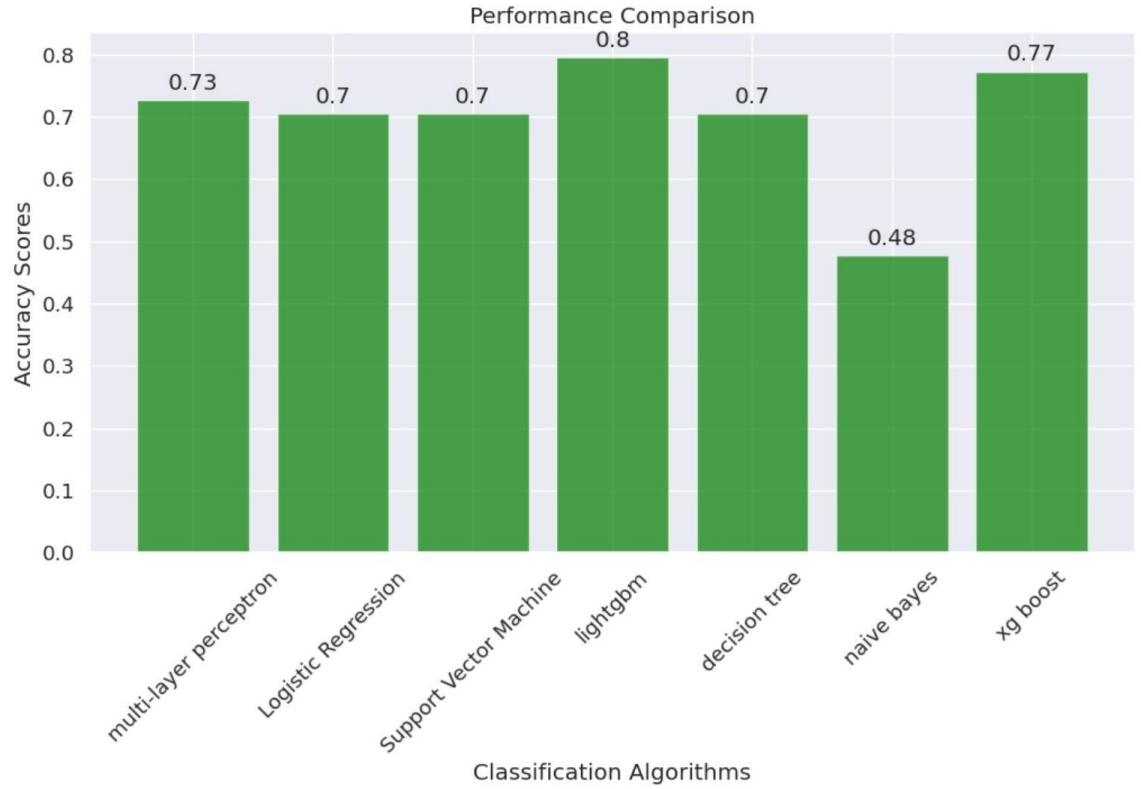

```
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.89      0.83        27
           1       0.77      0.59      0.67        17

    accuracy                           0.77        44
   macro avg       0.77      0.74      0.75        44
weighted avg       0.77      0.77      0.77        44
```

```
import xgboost as xgb
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
import numpy as np
import matplotlib.pyplot as plt

n_bootstrap_samples = 50
n_iterations = 100

# Create an XGBoost classifier model
model = xgb.XGBClassifier()

accuracy_scores = []

for _ in range(n_iterations):
    # Resample the training data
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=n_bootstrap_samples)

    # Fit the XGBoost classifier model to resampled data
    model.fit(X_resampled, y_resampled)

    # Predict using the XGBoost classifier model on the test data
    y_pred = model.predict(X_test)
```

```
    # Calculate accuracy score
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

plt.figure(figsize=(8, 6))
plt.plot(range(n_iterations), accuracy_scores)
plt.xlabel("Iteration")
plt.ylabel("Accuracy Score")
plt.title("Accuracy Scores for Bootstrapped XGBoost Classifier")
plt.show()
```

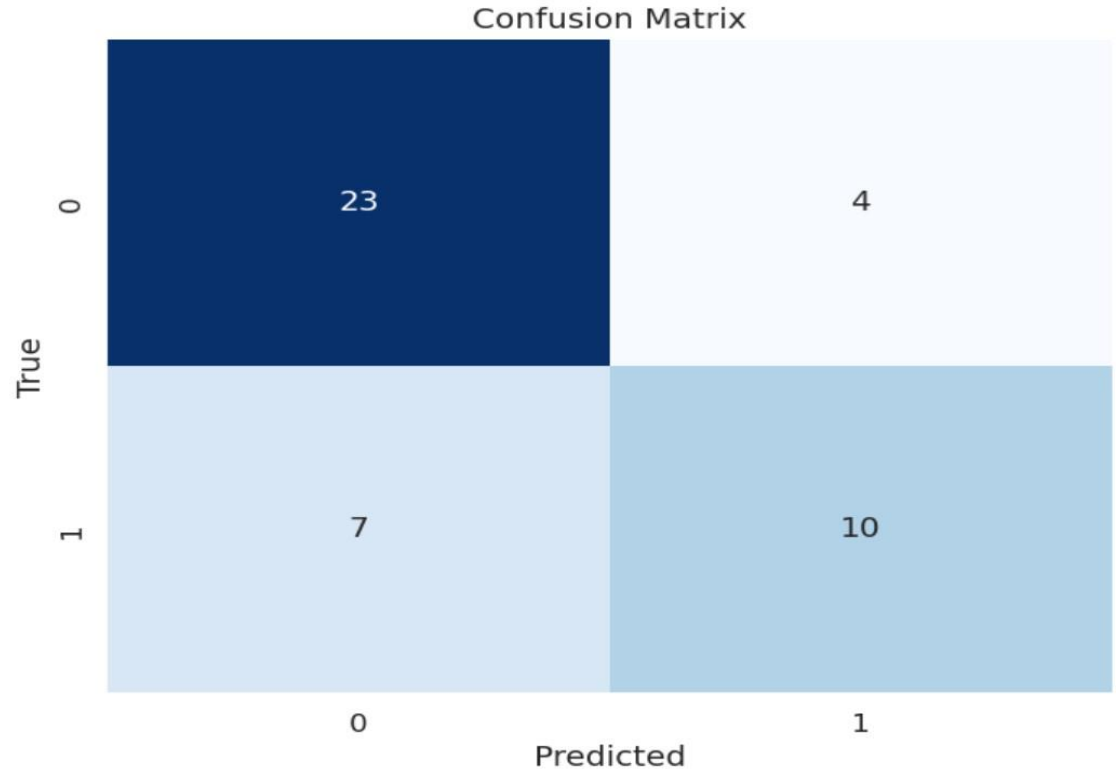


```python
import matplotlib.pyplot as plt

algorithm_names = ['multi-layer perceptron', 'Logistic Regression', 'Support Vector Machine','lightgbm','decision tree','naive bayes','xg boost
accuracy_scores = [accuracy_MLP, accuracy_LR, accuracy_SVM,accuracy_LGBM,accuracy_DT,accuracy_NB,accuracy_XGB]

plt.figure(figsize=(12, 6))

# Bar chart for accuracy scores
plt.bar(algorithm_names, accuracy_scores, color='Green', alpha=0.7)
plt.xlabel('Classification Algorithms')
plt.ylabel('Accuracy Scores')
plt.title('Performance Comparison')

# Adding data labels
for i, v in enumerate(accuracy_scores):
    plt.text(i, v + 0.01, str(round(v, 2)), ha='center', va='bottom')

# Rotating the x-axis labels for better visibility
plt.xticks(rotation=45)

# Displaying the plot
plt.show()
```



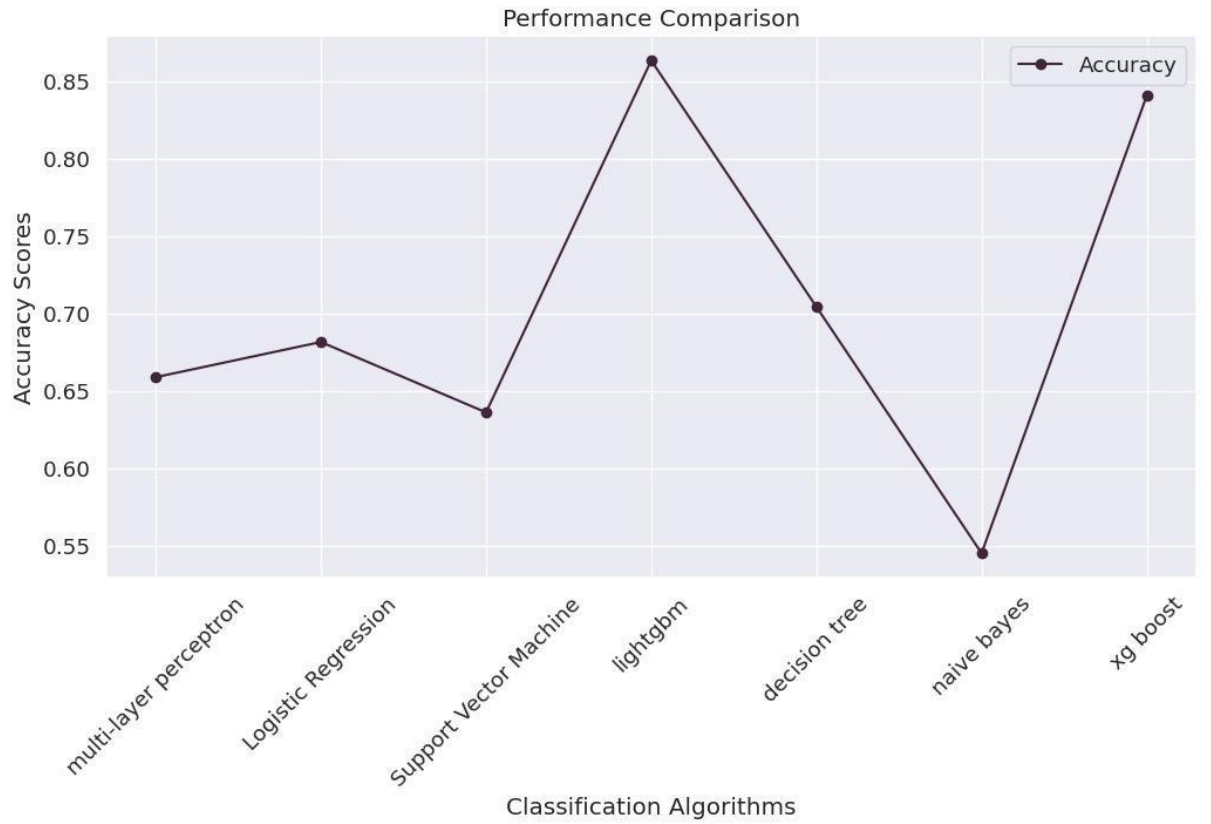
```python
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(y_test, y_pred)
print ("Confusion Matrix : \n", cm)
plt.figure(figsize=(8, 6))
sns.set(font_scale=1.2)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score

# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Calculate and print precision
precision = precision_score(y_test, y_pred, average='weighted',zero_division=0)
print("Precision:", precision)

# Calculate and print recall
recall = recall_score(y_test, y_pred, average='weighted')
print("Recall:", recall)

# Calculate and print F1-score
f1 = f1_score(y_test, y_pred, average='weighted')
print("F1-score:", f1)
```

```
Confusion Matrix :
 [[23  4]
 [ 7 10]]
```




```
Accuracy: 0.75
Precision: 0.7464285714285714
Recall: 0.75
F1-score: 0.7444821731748725
```

```
algorithm_names = ['multi-layer perceptron', 'Logistic Regression', 'Support Vector Machine','lightgbm','decision tree', naive bayes
accuracy_scores = [accuracy_MLP, accuracy_LR, accuracy_SVM,accuracy_LGBM,accuracy_DT,accuracy_NB,accuracy_XGB]

plt.figure(figsize=(12, 6))
# Accuracy scores plot
plt.plot(algorithm_names, accuracy_scores, label = 'Accuracy', marker = 'o', color = '#42253B')
plt.xlabel( 'Classification Algorithms' )
plt.ylabel( 'Accuracy Scores' )
plt.title( 'Performance Comparison' )
# Adding a legend
plt.legend()

# Rotating the x-axis labels for better visibility
plt.xticks( rotation = 45 )

# Displaying the plot

plt.show ()
```



9. CONCLUSION:

Experts aimed to estimate student success charges in higher training the use of machine gaining knowledge of strategies Data amassed from academic establishments and engineering schools turned into classified into grades for analysis Factors including gender, high college type, scholarship kind, extra work, dating popularity, general revenue, and transportation manner have been considered in predicting scholar grades.i used the above mentioned classification methods comparing the all methods. I got the best accuracy and the best confusion matrix and also the f1 score is in light gbm and also in the XG Boost.so i conclude that the these methods are perfect for my dataset.

REFERENCES

[1]Abu Saa, A., Al-Emran, M. and Shaalan, K. Factors Affecting Students' Performance in Higher Education: A Systematic Review of Predictive Data Mining Techniques. Tech Know Learn 24, 567–598 (2019). https://doi.org/10.1007/s10758019-09408-7

[2]Struyven, K., Dochy, F., and Janssens, S. (2005). Students' perceptions about evaluation and assessment in higher education: a review1 . Assessment and Evaluation in Higher Education, 30(4), 325–341.
https://doi.org/10.1080/02602930500099102

[3]Omaya Kuran, Closing the Theory-Practice Gap: Socio-
Economic Approach and Action Research in Management, Systemic Practice and Action Research, 10.1007/s11213-02309666-8, (2024).

[4]Rahiman, H. U., and Kodikal, R. (2024). Revolutionizing education: Artificial intelligence empowered learning in higher education. Cogent Education, 11(1).
https://doi.org/10.1080/2331186X.2023.2293431

[5]Sullivan, D., Lakeman, R., Massey, D., Nasrawi, D., Tower, M., and Lee, M. (2024). Student motivations, perceptions and opinions of participating in student evaluation of teaching surveys: a scoping review. Assessment and Evaluation in Higher Education, 49(2), 178–189.
https://doi.org/10.1080/02602938.2023.2199486

[6]Romhild,¨ A., and Holle derer, A. (2024). Effects of disability-related services, accommodations, and integration on academic success of students with disabilities in higher education. A scoping review. European Journal of Special Needs Education, 39(1), 143–166.
https://doi.org/10.1080/08856257.2023.2195074

[7]Karunarathne, W., and Calma, A. (2024). Assessing creative thinking skills in higher education: deficits and improvements. Studies in Higher Education, 49(1), 157–177.
https://doi.org/10.1080/03075079.2023.2225532

[8]Hutchinson, M., Coutts, R., Massey, D., Nasrawi, D., Fielden, J., Lee, M., and Lakeman, R. (2024). Student evaluation of teaching: reactions of Australian academics to anonymous nonconstructive student commentary. Assessment and Evaluation in Higher Education, 49(2), 154– 164. https://doi.org/10.1080/02602938.2023.2195598

[9]Jin, X., Jiang, Q., Xiong, W., Feng, Y., and Zhao, W. (2024). Effects of student engagement in peer feedback on writing performance in higher education. Interactive Learning Environments, 32(1),
128–143. https://doi.org/10.1080/10494820.2022.2081209

[10]Nieminen, J. H. (2024). Assessment for Inclusion: rethinking inclusive assessment in higher education. Teaching in Higher Education, 29(4), 841–859. https://doi.org/10.1080/13562517.2021.2021395

[11]Sudi, M., Arisanti, I., Hanim, S. A., Sya'rani, R., and Rahwana, K. A. (2024). The Effect of Organizational Culture and Communication Skills on Administrative Performance in Higher Education Institutions in East Java. West Science Interdisciplinary Studies, 2(02), 440–447. https://doi.org/10.58812/wsis.v2i02.673

[12]Boud, D., and Bearman, M. (2024). The assessment challenge of social and collaborative learning in higher education. Educational Philosophy and Theory, 56(5), 459–468. https://doi.org/10.1080/00131857.2022.2114346

[13]Crawford, J., Allen, K. A., Sanders, T., Baumeister, R., Parker, P., Saunders, C., and Tice, D. (2024). Sense of belonging in higher education students: an Australian longitudinal study from 2013 to 2019. Studies in Higher Education, 49(3), 395–409. https://doi.org/10.1080/03075079.2023.2238006