# 1. INTRODUCTION

## 1.1. Background Motivation

For Michel Foucault, school is only a component of the area of disciplinary power, and assessment is only one of its instruments. Thus, if we talk about grades and grading, we should refer to their ability to create conformity. The French thinker highlights the fact that examination is defined as “normative”, as “surveillance that allows definition, classification and punishment” (see Michel Foucault, p. 266). Michel Foucault demonstrates that the power of King, which was strengthened and re-legitimized through lavish ceremony by placing himself in broad daylight, was replaced by disciplinary power, diffuse, tiny taking care to hide and to shed light on the subject upon which is manifested. Thus, the subject is (only) changed into the object of power. Analyzing the changes in school assessment reveals a striking analogy with the process described above: examination as a means of highlighting the teacher’s power has been increasingly replaced with devices and dissipation mechanisms of assessment, all united under the name rhythmic grading. The process of replacement continues, because today we are talking (and use) more and more complementary or alternative methods of assessment (softer, less visible in their concrete manifestation). In other words, the efficiency of disciplinary power depends on the “banality”, the insignificance of its elements, both socially and at school. It follows that grades and grading are part of the wider flux of the action of the disciplinary power, even if it leaves the impression of technical mechanisms for recording the academic achievement of students. In this context, we intend to discuss the relationship between grades and student motivation for learning, among others, because the type and level of motivation of students depend significantly on the social-community environment, outside school, “the way in which media present illiterate or semi-illiterate persons who have acquired fame, political power and financial resources steadily erodes learning motivation for students in Romania. This is compounded by the dramatic fall in social prestige associated with the teaching staff, their economic status is in itself de-motivating.” (see StefanPopenici, p. 25) Besides the significant correlation between external environment and motivation for learning, one should reveal the correlation (at least) equally significant for involvement in the educational scenario of teachers and the motivation for learning of students.

## 1.2. problem statement

Researchers have discovered a litany of student complaints about group grad ing. According to Kagan (1995, 1996), group grades are unfair, debase grade reports, undermine motivation, convey incorrect messages, violate individual accountability, generate resistance to cooperative learning, and often are challenged in court. In a critical essay on group project assessment cast from the perspective of game theory, Pitt (2000) concludes that problems with group assessment should be expected. Under the tenets of the theory, students' desires to receive the highest individual grades are, in these systems, at odds with effective cooperative learning. From Pitt's perspective, the following conclusions are drawn: 1. Any method of selecting groups and allocating projects, whether random or systematic, in general will give some groups an advantage and some a disadvantage. 2. Giving all students the same mark means that a sensible group strategy would involve having the weaker students contribute less. 3. Although the allocation of marks is a motivator, factors such as teamwork and contribution to the group are hard to define and essentially impossible to assess fairly. 4. Rating students on some perceived performance has as much to do with perception as performance and may sometimes be unfair; for example, the student who contributed least to the problem solving may give the most confident presentation. 5. Some assessment factors can actually promote dishonesty and competition. (239-40) All of the alleged problems described should concern instructors and administrators, both of whom are responsible forth justification and defense of grading methods and must deal with grade com plaints. When instructors assign one grade to an entire group, they are essentially using a mean to describe a population, with no thought given to dispersion. Some instructors assign the group grade on the basis of performance of the lowest group member, the highest group member, or even a member selected at random. Instructors concerned about measurement validity and reliability will have serious problems accepting grading methods that appear to maximize measurement error. Some of these problems are difficult to manage. For example, groups can vary widely in terms of individual student talent, suggesting that more work and lower grades could be a natural consequence of being in a bad group. For this reason, heterogeneous grouping is suggested as an "essential element" (Sheeran 1994, 20) of group projects. Yet, this practice singles out and isolates students who lack skills or talent. In some cases, these students are at risk academically, special students, or minority group members. One example of such a quandary is the decision to split a small group of female students in an engineering class into several work groups. Although educators do admit that the isolation of minor ties and low performers could cause these students to be fired by the group or to drop out of class (Felder and Brent 2001; Rosser 1998), they acknowledge the intractability of the problem: "The minorities will often find themselves iso late in workgroups on the job, and they may as well start learning how to deal with it while still in college" (Felder and Brent 2001, 70). The issue is whether or not to simulate the problems of the work world more accurately at the expense of fairness and methodological rigor. Educators' recent focus on these problems has prompted the emergence of a significant body of work concerned with finding ways to negotiate the minefields encountered in implementing group grading schemes (Cheng and Warren 2000; Goldfinch and Resided 1990; Kroll, Masingila, and Mau 1992; Lejk, Wyvill, and Farrow 1996; Maranto and Gresham 1998; Sheeran 1994). One interesting approach is to have the students themselves assign a grade or portion of a grade. Numerous possibilities exist with in this strategy. The instructor can assign a grade to the entire group that is modified by the individual member's personal rating. Another variation on that theme is one in which students either privately or publicly provide feedback on the other members' performances directly to the instructor, who then assigns the grades. Finally, group members can be given a proportion of bonus points based on the overall quality of the final project, which can then be awarded to group members equally or unequally according to predetermined rules or criteria. When group members are involved in evaluating the performance of their peers, it can be argued that an abandonment of instructional responsibility has occurred. If a grade complaint alleging unfairness is filed, the instructor cannot be confident that the grade was fairly arrived at when he or she was not present in group deliberations or evaluations. Second, this process can be prone to halo error. Well-liked students, argumentative students, ethnically divergent students, and apprehensive students may be subconsciously rated on characteristics other than the quality of their performances by peers who are inadequately trained about such issues. Finally, in a system in which only a limited number of high grades are made available, team members may be inclined to down grade the performance of others because they themselves are part of the competition. Conversely, where there are no limits on high grades, students may inflate their ratings of each other in a conscious or sub conscious evaluation conspiracy. Generally speaking, then, group projects are graded by assigning a group grade to all members, with the many disadvantages discussed earlier, or group members may assign grades. Attempts to provide a balance between these two approaches probably result in some level of each set of problem.

## 1.3 project goal:

Generally, the goal of grading is to evaluate individual students' learning and performance. Although grades are sometimes treated as a proxy for student learning, they are not always a reliable measure. Moreover, they may incorporate criteria – such as attendance, participation, and effort – that are not direct measures of learning. The goal of assessment is to improve student learning. Although grading can play a role in assessment, assessment also involves many ungraded measures of student learning. Moreover, assessment goes beyond grading by systematically examining patterns of student learning across courses and programs and using this information to improve educational practices.

## 1.4 Project Objectives:

1. Since the students are grouped in different types of grading scales, it gives them an idea about their weaknesses and strengths. This not only allows teachers to focus more on the average and below-average students but also provides an opportunity for students to improve upon their weak points.

2. The grading system in India also gives parents an opportunity to assess their child's capabilities and find the aspects they need to focus on to help their child become better at studies.

3. It ensures uniformity across schools and colleges both in India and abroad.

4. As compared to the older version, the modernly updated grading system in India no longer allows classifying students solely based on marks.

5. It also provides students with the necessary educational prerequisites to mound themselves to become better versions of themselves.

## 1.5 Existing System:

To complete these studies, most researchers use a questionnaire survey method to collect data from specific students in specific circumstances. However, the method of collecting data has some limitations. First, it is impossible to capture students' current state in a timely manner with this method because surveys are conducted on a scheduled basis, such as one per academic year or semester. If students with unexpected behavioral patterns cannot be identified in a timely manner, there may be serious consequences. Second, students exhibiting anomalous behaviors may deliberately supply false information to make them appear normal, while normal students may not carefully fill out the survey, so the collected data could contain noise or false information.

## 1.6 Proposed System:

we propose an unsupervised ensemble clustering framework to use student behavioral data to discover behavioral patterns. Because the behavioral data produced by students on campus are available in real time without intentional bias, clustering analysis can be relatively efficient and reliable. The proposed framework extracts behavior features from the two perspectives of statistics and entropy and then combines density-based spatial clustering of applications with noise (DBSCAN) and k-means algorithms to discover behavioral patterns. To evaluate the performance of the proposed framework, we carry out experiments on six types of behavioral data produced by undergraduates in a university in Beijing and analyze the relationships between different behavioral patterns and students' grade point averages (GPAs).

## 1.7 Benefits of proposed system:

It will minimize misclassification of students on the basis of marks. It will eliminate unhealthy competition among high achievers. It will reduce societal pressure and will provide the learner with more flexibility. It will lead to a focus on a better learning environment Operational It will facilitate joyful and stress free learning.

# 2. LITERATURE REVIEW

## 2.1 Project Literature:

Early to bed, early to rise! Sleep habits and academic performance in college student's Prior studies have placed emphasis on the need for adequate total sleep time for student performance. We sought to investigate the relative importance of total sleep time compared to the timing of sleep and wakefulness for academic performance. We performed a questionnaire-based survey of college students in October 2007. The questionnaire gathered detailed information on sleep habits including naps, reasons for missing sleep, academic performance, study habits, time spent working outside of school, and stimulant use. Timing of sleep and wakefulness correlated more closely with academic performance than total sleep time and other relevant factors. These findings have important implications for programs intended to improve academic performance by targeting sleep habits of students.

**ASSOCIATION OF WEEKLY STRENGTH EXERCISE FREQUENCY AND ACADEMIC PERFORMANCE AMONG STUDENTS AT A LARGE UNIVERSITY IN THE UNITED STATES**

Keating, XD, Castelli, D, and Ayers, SF. Association of weekly strength exercise frequency and academic performance among students at a large university in the United States. J Strength Cond Res 27(7): 1988–1993, 2013—The study aimed to examine (a) the association between weekly strength exercise frequency and grade point average (GPA), and (b) the demographic characteristics of weekly strength exercise frequency among undergraduate students at a large southern state university in the United States. Health behavior data (N = 1125) collected by the American College Health Association at the university in 2008 were analyzed. Analysis of variance was used to investigate weekly strength exercise frequency differences in GPA, sex, ethnicity, and year in university. The results revealed that those who more frequently engaged in strength exercise had significantly higher GPA. There was a significant difference in weekly strength exercise frequency by sex and ethnicity. Findings suggest that regular engagement in strength exercise may not only have physical health benefits investigate the mechanism of strength exercise on GPA among university students.

## Association between Eating Behavior and Academic Performance in University Students

To determine the association between academic performance and eating behavior in university students in Chile. A total of 680 college students, 409 (60%) women and 271 (40%) men, were randomly recruited and the mean age of the entire sample was 26. The Three-Factor Eating Questionnaire (TFEQ), which evaluates 3 dimensions of eating behavior—cognitive restriction (limiting own intake), uncontrolled eating (inclination to eat), and emotional eating (control of food intake in the context of negative emotions)—was used. Academic performance was measured by the grade point average (GPA) and was associated with eating behavior.

Women had significantly higher scores in the "emotional eating" dimension than men (p D 0.002). The eating behavior analysis showed that female students with higher GPAs (above 5.5) had statistically significantly lower uncontrolled eating scores (p D 0.03) and higher cognitive restriction scores (p D 0.05) than women with lower academic performance (below 5.5).

There were no significant associations between eating behavior and academic performance in men. A positive association between eating behavior and academic performance was observed in female university students in Chile. Further studies are needed to explore the causes of this association and determine how to improve the nutritional habits of this population.

### Modelling the impact of study behaviors on academic performance to inform the design of a persuasive system.

Information technology is deeply ingrained in most aspects of everyday life and can be designed to influence users to behave in a certain way. Influencing students to improve their study behavior would be a useful application of this technology. As a preamble to the design of a persuasive system for learning, we collected data to identify the study behaviors of students and recent alumni. We then developed two models to measure which behaviors have the most significant impact on learning performance. Current students reported more foundational behaviors whereas alumni demonstrated more higher-order thinking traits.

Focusing on behavioral relationship analysis, this paper presents a new method based on genetic algorithm to efficiently discover behavioral correlates of mental health (i.e., depression, stress, loneliness, self-perceived success) and academic performance. We conduct the experiments based on the open dataset called “Student Life”. Ultimately, we discover interesting relationships which have never been considered before. We believe this approach provides a novel and effective way to mine massive relationships from mobile contextual data in the future.

Technology can be used to support students to improve their study behaviors and strategies. As there are a multitude of behaviors that could potentially benefit student learning, we used a statistical process to identify which study behaviors have the greatest impact on academic performance. Two models were created that covered different dimensions of performance self-perception and results achieved. We then modelled the data based on current students and alumni and identified a general trend towards behaviors and factors that provide immediate benefits for current students, and higher-order thinking behaviors and factors for alumni. The models outlined in this paper form the basis from which persuasive systems can be designed to improve learning outcomes, as they provide a richer picture of how student learning behaviors naturally develop. Drawing on this knowledge, persuasive systems for education can now aim to influence the natural their behavioral deficiencies. Mining relationships between mental health academic performance and human behaviors the relationships among human behavior, mental health and academic performance have attracted researchers in psychology and education for many years. With the increase of the embedded sensor, the mobile phone can collect the data of the users' behavior from different aspects and different dimensions continuously and unconsciously.

Focusing on behavioral relationship analysis, this paper presents a new method based on genetic algorithm to efficiently discover behavioral correlates of mental health (i.e., depression, stress, loneliness, self-perceived success) and academic performance. We conduct the experiments based on the open dataset called “Student Life”. Ultimately, we discover interesting relationships which have never been considered before. We believe this approach provides a novel and effective way to mine massive relationships from mobile contextual data in the future.

## 2.2 Existing system:

To complete these studies, most researchers use a questionnaire survey method to collect data from specific students in specific circumstances. However, the method of collecting data has some limitations. First, it is impossible to capture students' current state in a timely manner with this method because surveys are conducted on a scheduled basis, such as one per academic year or semester. If students with unexpected behavioral patterns cannot be identified in a timely manner, there may be serious consequences. Second, students exhibiting anomalous behaviors may deliberately supply false information to make them appear normal, while normal students may not carefully fill out the survey, so the collected data could contain noise or false information.

## 2.3 proposed system:

we propose an unsupervised ensemble clustering framework to use student behavioral data to discover behavioral patterns. Because the behavioral data produced by students on campus are available in real time without intentional bias, clustering analysis can be relatively efficient and reliable. The proposed framework extracts behavior features from the two perspectives of statistics and entropy and then combines density-based spatial clustering of applications with noise (DBSCAN) and k-means algorithms to discover behavioral patterns. To evaluate the performance of the proposed framework, we carry out experiments on six types of behavioral data produced by undergraduates in a university in Beijing and analyze the relationships between different behavioral patterns and students' grade point averages (GPAs).

## 2.4 Applications:

In contract grading instructors list activities students can participate in or objectives they can achieve, usually attaching a specified number of points for each activity (e.g. book report = 30 points, term paper = 60 points). Students select the activities and/or objectives that will give them the grade they want and a contract is signed. It is advisable to have qualitative criteria stated in the contract in addition to listing the activities.

In some classes, a portion of a student's grade is determined by peers' evaluation of his/her performance. If students are told what to look for and how to grade, they generally can do a good job. Agreement between peer and instructor rating is about 80%. Peer grading is often used in composition classes and speech classes. It can also be a useful source of information for evaluating group work; knowing that group members have the opportunity to evaluate each other's work can go a long way in motivating peers to pull their weight on a project and to reassure group members that their contributions will be recognized. If used, peer evaluation should always be done anonymously.

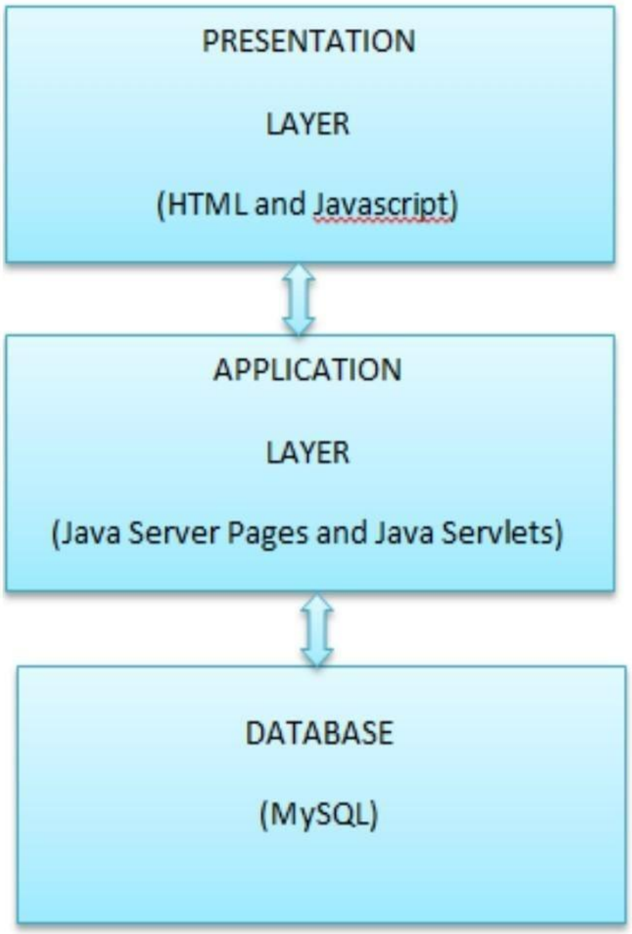
Students can also be asked to assess their own work in the class and their assessment can be a portion of the final grade. This method has educational value as learning to assess one's own progress contributes to the university's goal of preparing our students to be life-long learners. A research analysis found that the percentages of self-assessors whose grades agree with those of faculty graders vary from 33% to 99%. Experienced students tend to rate themselves quite similarly to the faculty while less experienced students generally give themselves higher grades than a faculty grader. Students in science classes also produced self-assessments that closely matched faculty assessment. Not surprisingly, student and instructor assessments are more likely to agree if the criteria for assessment have been clearly articulated. Without these shared understandings, students, for example, don't know whether to assess themselves on the amount of work they put into a course, on the improvement they've seen in certain skills, or on their final level of achievement. If self-assessment is used, the instructor and student should meet to discuss the student's achievement before the self-evaluation is made.

## 2.5 Summary:

When the subjects are to be allocated to the faculty, the Head of the Department should enter everything in the Excel sheets. Then the staff enters corresponding subject's attendance and marks of a student then those must also be entered in the Excel sheets and validations are to be done by the user itself. So, there will be a lot of work to be done and must be more conscious during the entrance of details. So, more risk is involved. PROBLEMS IN THE EXISTING SYSTEM: Storing and accessing the data in the form of Excel sheets and account books is a tedious work. It requires a lot of laborious work. It may often yield undesired results. Maintaining these records as piles may turn out to be a costlier task than any other of the colleges and institutions RISKS INVOLVED IN EXISTING SYSTEM: Present System is time-consuming and also results in lack of getting inefficient results. Some of the risks involved in the present system are During the entrance of marks and attendance, if any mistake is done at a point, then this becomes cumulative and leads to adverse consequences If there is any need to retrieve results it may seem to be difficult to search.

## Architecture:

The Online Grading System is designed and developed using a 3-tier architecture. The 3- tier architecture consists of a presentation layer, an application layer, and a database. The presentation layer is the front-end of a web application that is used to provide a rich user interface experience to the end user. The application layer is the middle layer of a web application that is used to process all the requests sent by the user by using front-end application. The database is used to store all the information required by the Online Grading System.



## Presentation Layer:

The presentation layer is the part of the web application that is visible to the end user. The presentation layer is used by the end user to interact with the web application and send requests to process. In the Online Grading System, HTML and Javascript is used to develop the presentation layer. Application Layer: The application layer is the middle layer of a web application. The application layer is used to process the requests sent by end user using the presentation layer. The application layer also works as a middle level layer to process transactions between the presentation layer and the database. In the Online Grading System, JSP and Java servlets are used to develop the application layer.

Database: The database is the third and final layer of a web application. The database is used to store and retrieve the user information when necessary. In the Online Grading System, MySQL data is used to store and retrieve the user information.

Application Design: The Online Grading System application consists of three types of users. They are Administrator, Professors, and Students. Detailed user privileges are as follows:

### Administrator:

1. Administrator has the highest privilege.
2. Administrator can add professors.
3. Administrator can add a semester at the start of a university academic semester.

### Professors:

1. Professors can add students to the system.
2. Professors can delete a student from the system.
3. Professors can add a course to the system.
4. Professors can drop a course from the system.
5. Professors can post assignments and assign deadlines to the corresponding assignment.
6. Professors can view the assignments submitted by the students.
7. Professors can compile and execute the programming assignments using the Online Grading System.
8. Professors can view the results of the programming assignments.
9. Professors can assign grades for student assignments.

## Students:

1. Students can register in the system.
2. Students can enroll for a course or drop a course.
3. Students can view the assignments posted by professors for enrolled courses.
4. Students can submit assignments before the deadlines.
5. Students can view the assignment.

# 3. SOFTWARE REQUIREMENTS AND SPECIFICATIONS

## 3.1 Software requirements:

- Operating System: Windows XP
- Programming Language: Python

## 3.2 Hardware requirements:

- Processor - Pentium –IV
- Speed - 1.1 Ghz
- RAM - 256 MB(min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

## 3.3 System feasibility:

- **ECONOMIC FEASIBILITY**

  A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

- **OPERATIONAL FEASIBILITY**

  Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

- **TECHNICAL FEASIBILITY**

  Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

## Software Requirement Specification

Overall Description A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers.

Projects are subject to three sorts of requirements:

- Business requirements describe in business terms what must be delivered or accomplished to provide value.
- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economic feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs. The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, there is nominal expenditure and economic feasibility for certain.

- OPERATIONAL FEASIBILITY Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation.

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The help in the improvement of performance status.

- TECHNICAL FEASIBILITY Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web-based user interface for audit workflow at NIC-CSD. Thus, it provides an easy access to. the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security.

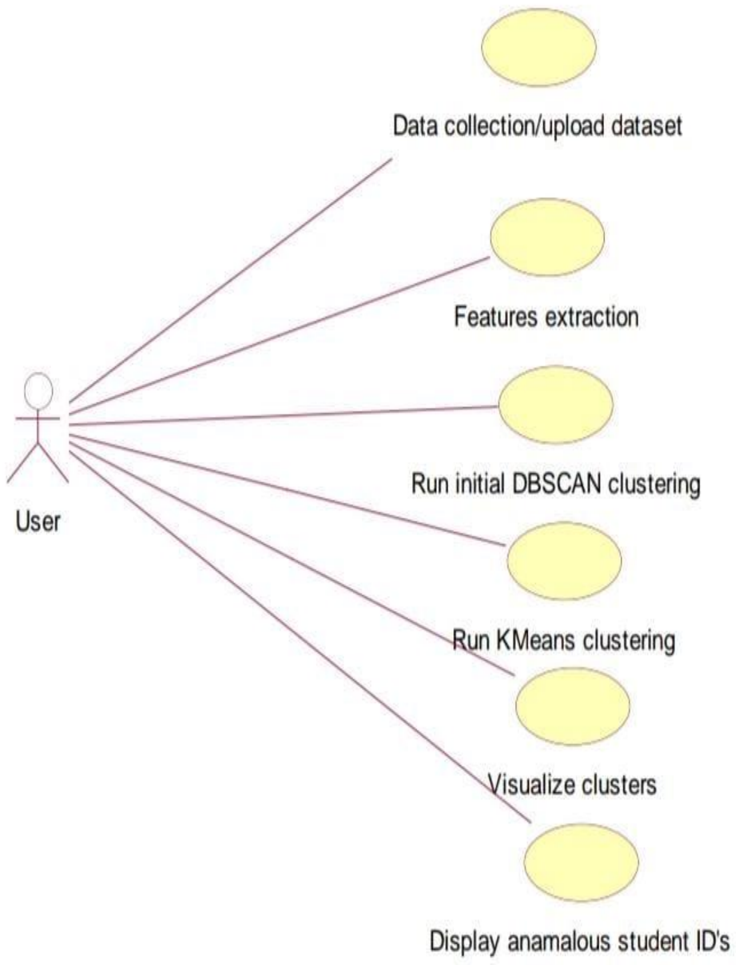
## Architecture:

The Online Grading System is designed and developed using a 3-tier architecture. The 3- tier architecture consists of a presentation layer, an application layer, and a database. The presentation layer is the front-end of a web application that is used to provide a rich user interface experience to the end user. The application layer is the middle layer of a web application that is used to process all the requests sent by the user by using front-end application. The database is used to store all the information required by the Online Grading System.

# 4. SYSTEM DESIGN

## 4.1 Use case diagram:

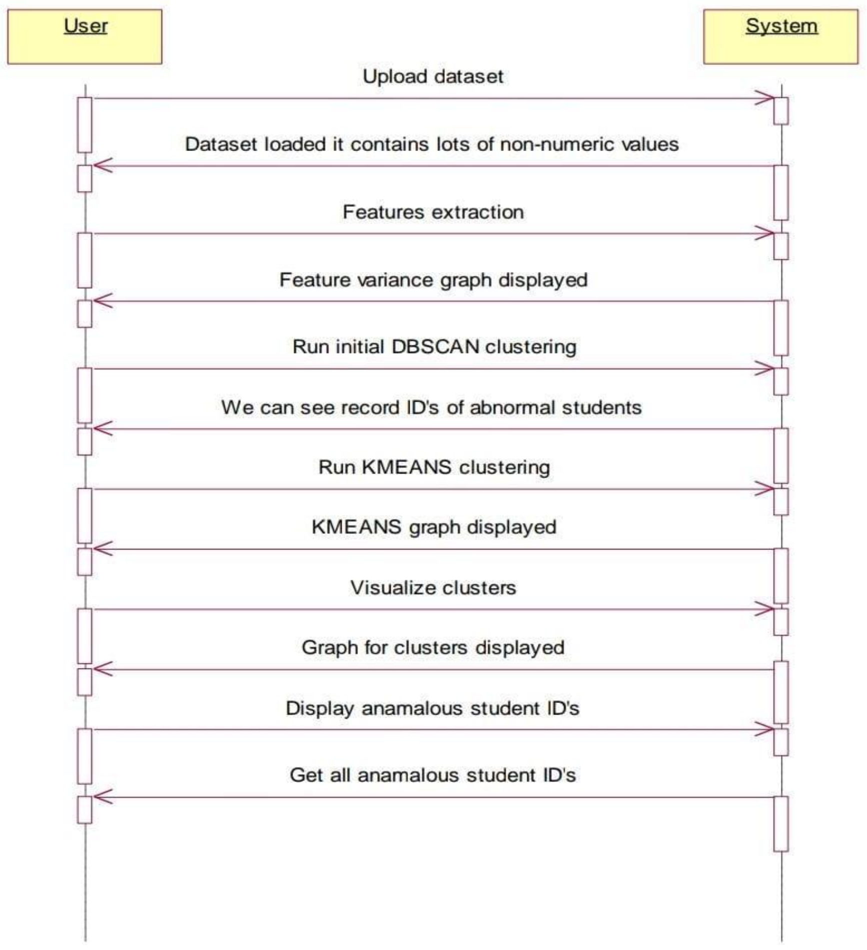
A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.


Data collection/upload dataset
Features extraction
Run initial DBSCAN clustering
Run KMeans clustering
Visualize clusters
Display anamalous student ID's
User

## 4.2 Sequence diagram:

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

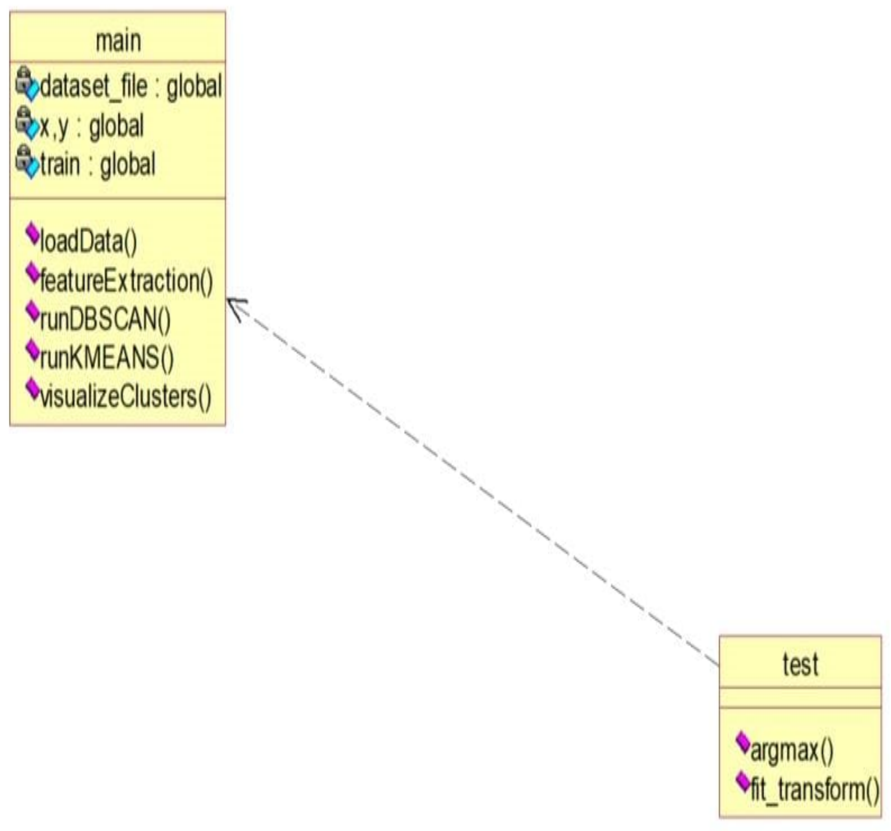
**Sequence diagram:**


User
System
Upload dataset
Dataset loaded it contains lots of non-numeric values
Features extraction
Feature variance graph displayed
Run initial DBSCAN clustering
We can see record ID's of abnormal students
Run KMEANS clustering
KMEANS graph displayed
Visualize clusters
Graph for clusters displayed
Display anamalous student ID's
Get all anamalous student ID's

## 4.3 Class diagram:

The class diagram is the main building block of object-oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts the upper part holds the name of the class and the middle part contains the attributes of the class and the bottom part gives the methods or operations the class can take or undertake.

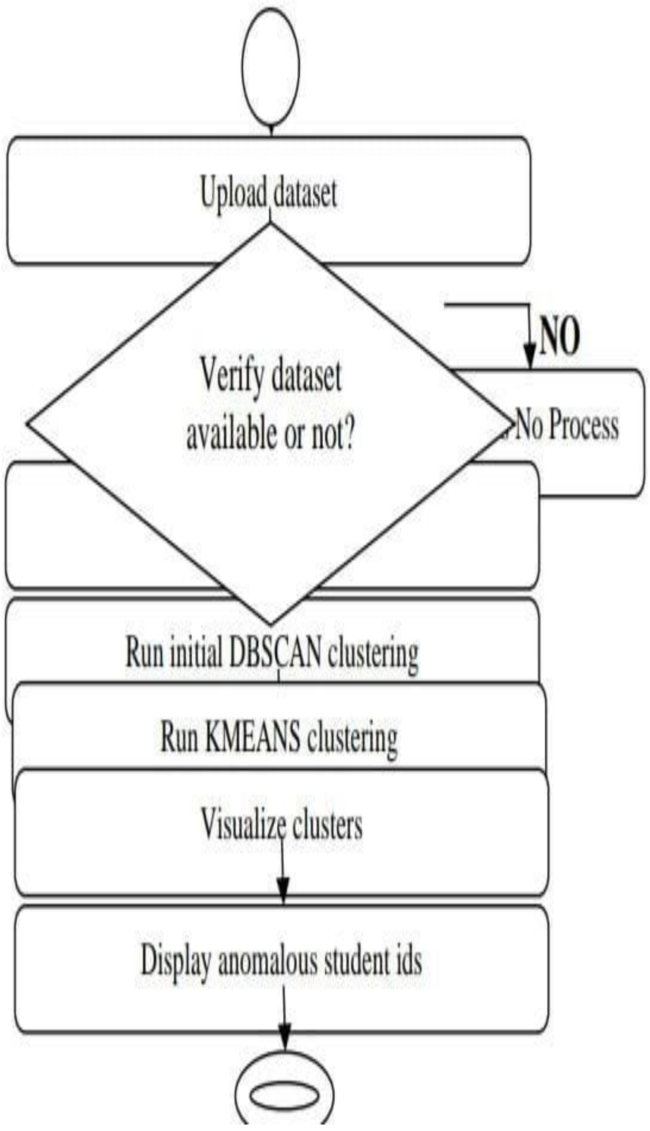
**Class diagram:**



## 4.4 Activity diagram:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

**Activity diagram:**


Upload dataset
Verify dataset available or not?
NO
No Process
Run initial DBSCAN clustering
Run KMEANS clustering
Visualize clusters
Display anomalous student ids

# 5. IMPLEMENTATION

## 5.1Enivronmental setup:

An important task in education field is discovering student behavioural patterns to take timely action to improve student activities or grades. Sometime students may fell into depression due to understanding of subjects or due to low grade which leads into abnormal behaviour and by identifying such abnormal behaviour institutions can take necessary steps to improve student’s condition. In propose paper author is analyzing student behaviour by monitoring various activities such as logging to internet access, attending library timing, dinner, breakfast etc.

Students who are in normal mood will not care much on their activities and students who are in abnormal states may concentrate more on normal activities to pretend like normal person and such abnormal concentration will generate noise in output and this noise helps in identifying anomalous or abnormal student behaviour. In all existing work they were using single clustering algorithm to cluster student behaviour where all similar behaviour data will group into same cluster and all noise or abnormal behaviour data will group into other cluster. Author is saying applying single clustering technique may not give accurate abnormal behaviour so author applying ensemble (combination of multiple algorithm) algorithms called DBSCAN and KMEANS clustering.

In initial step DBSCAN clustering will apply on dataset to get clusters and all similar normal behaviour will go to single main stream cluster and abnormal behaviour data will go to noise cluster. Single Main Stream normal cluster may contain huge number of records and may further contains abnormal behaviour so author applying KMEANS clustering to further decompose MAIN SINGLE cluster to form multiple clusters and the cluster will least number of records will be considered as ANOMALOUS or abnormal students records. To implement this project author has built his own dataset from his university students and not publish on internet and no related dataset also available on internet which contains students’ behaviour on library, breakfast or dinner etc. So, we used student performance dataset from KAGGLE and below screen showing some records from student performance dataset

**gender, NationalITy, PlaceofBirth, StageID, GradeID, SectionID, Topic, Semester, Relation, raisedhands, VisITedResources, AnnouncementsView, Discussion, ParentAnsweringSurvey, ParentschoolSatisfaction, StudentAbsenceDays, Class**

M,KW,KuwaIT,lowerlevel,G-04,A,IT,F,Father,15,16,2,20,Yes,Good,Under-7,M

M,KW,KuwaIT,lowerlevel,G-04,A,IT,F,Father,20,20,3,25,Yes,Good,Under-7,M

M,KW,KuwaIT,lowerlevel,G-04,A,IT,F,Father,10,7,0,30,No,Bad,Above-7,L

M,KW,KuwaIT,lowerlevel,G-04,A,IT,F,Father,30,25,5,35,No,Bad,Above-7,L

M,KW,KuwaIT,lowerlevel,G-04,A,IT,F,Father,40,50,12,50,No,Bad,Above-7,M

F,KW,KuwaIT,lowerlevel,G-04,A,IT,F,Father,42,30,13,70,Yes,Bad,Above-7,M

M,KW,KuwaIT,MiddleSchool,G-07,B,IT,F,Father,36,30,20,80,No,Bad,Above-7,M

M,KW,KuwaIT,MiddleSchool,G-07,A,Math,F,Father,55,13,35,90,No,Bad,Above-7,M

F,KW,KuwaIT,MiddleSchool,G-07,A,IT,F,Mum,69,15,36,96,Yes,Good,Under-7,M

M,KW,KuwaIT,MiddleSchool,G-07,B,IT,F,Mum,70,50,40,99,Yes,Good,Under-7,H

In above records all bold names are the dataset column names and below those bold names are the dataset values and in above dataset we have semester and marks details and class label as Medium (M), HIGH (H) and LOW (L) are available and by applying this dataset we will identify students who are abnormal in scoring grades or doing other activities.

In this paper author has design four modules

1. Data Collection: In this module author uploading dataset to application

2. Feature Extraction: In above dataset some columns contain non-numeric values and clustering algorithm won’t take non-numeric values so we will applying reprocessing technique to convert non-numeric values to numeric by replacing MALE with 0 and FEMALE with 1 in gender column. Dataset contains 17 columns and all columns are not important so we apply PCA (principal component Analysis) algorithm which calculate importance of each feature and select only those important features and by applying this algorithm we are selecting best 6 features from dataset

3. DBSCAN Clustering: Using this module we will run DBSCAN clustering algorithm on above dataset to generate MAIN and NOISE cluster.

4. KMEANS Clustering: using this module we will apply KMEANS on Main cluster to further decompose into multiple clusters and the cluster with least records will be consider as ANAMALOUS or abnormal behavior.

### 5.1.1 Introduction of technologies used

### About Python:

Python is one of those rare languages which can claim to be both simple and powerful. You will be pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than on the syntax (i.e., the structure of the program that you are writing) of the language.

**The official Python introduction is**

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms

About Python: Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-baseddevelopment model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

## Python Versions

1. Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release, the development process became more transparent and community backed.

2. Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions. However, many of its major features have been backported to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

3. Python 2.7's end-of-life date was initially set at 2015, and then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. In January 2017, Google announced work on a Python 2.7 to Go trans compiler to improve performance under concurrent workloads.

## 5.2 Module description:

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints). System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers.

Projects are subject to three sorts of requirements:

- Business requirements describe in business terms what must be delivered or accomplished to provide value.
- Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
- Process requirements describe activities performed by the developing organization. For instance, process requirements could specify. Preliminary investigation examines project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time.

## 5.3 Software description:

### About Python:

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. In July 2018, Van Rossum stepped down as the leader in the language community after 30 years. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community-based envelopment model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

**Python Versions**

1. Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release, the development process became more transparent and community backed.

2. Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions. However, many of its major features have been backported to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

3. Python 2.7's end-of-life date was initially set at 2015, and then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. In January 2017, Google announced work on a Python 2.7 to Go trans compiler to improve performance under concurrent workloads.

## Features of Python

1. Simple Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English (but very strict English!). This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the syntax i.e., the language itself.

2. Easy to Learn As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax as already mentioned.

3. Free and Open-Source Python is an example of a FLOSS (Free/Library and Open-Source Software). In simple terms, you can freely distribute copies of this software, read the software's source code, make changes to it, use pieces of it in new free programs, and that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and improved by a community who just want to see a better Python.

4. High-level Language When you write programs in Python, you never need to bother about low-level details such as managing the memory used by your program.

5. Portable Due to its open-source nature, Python has been ported (i.e. changed to make it work on) to many platforms. All your Python programs will work on any of these platforms without requiring any changes at all. However, you must be careful enough to avoid any system dependent features. You can use Python on Linux, Windows, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

6. Interpreted This requires a little explanation. A program written in a compiled language like C or C++ is translated from the source language i.e., C/C++ into a language spoken by your computer (binary code i.e., 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software just stores the binary code in the computer's memory and starts executing from the first instruction in the program. When you use an interpreted language like Python, there is no separate compilation and execution steps. You just run the program from the source code. Internally, Python converts the source

code into an intermediate form called bytecodes and then translates this into the native language of your specific computer and then runs it. All this makes using Python so much easier. You just run your programs you never have to worry about linking and loading with libraries, etc. They are also more portable this way because you can just copy your Python program into another system of any kind and it just works!

7. Object Oriented Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simple way of doing object-oriented programming, especially, when compared to languages like C++ or Java.

8. Extensible If you need a critical piece of code to run very fast, you can achieve this by writing that piece of code in C, and then combine that with your Python program.

9. Embeddable You can embed Python within your C/C++ program to give scripting capabilities for your program's users.

10. Extensive Libraries the Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces) using Tk, and also other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the "batteries included" philosophy of Python. Besides the standard library, there are various other high-quality libraries such as the Python Imaging Library which is an amazingly simple image manipulation library.

## 5.4 sample code:

### StudentBehavior.py

```
from statistics import mode
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
import matplotlib.pyplot as plt
from tkinter.filedialog import askopenfilename
import os
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.preprocessing import normalize
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn import metrics
from numpy import unique
from numpy import where
from sklearn.cluster import KMeans

main = tkinter.Tk()

main.title("An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns")

main.geometry("1300x1200")

global dataset_file
global X, Y
global train
dbscan_main = []
dbscan_noise = []
kmeans_main = []



def loadData():
    global dataset_file
    global train
    dataset_file = filedialog.askopenfilename(initialdir="Dataset")

    pathlabel.config(text=dataset_file)
    text.delete('1.0', END)
    text.insert(END,dataset_file+"student behaviour dataset loaded\n\n")
    train = pd.read_csv(dataset_file)
    text.insert(END,str(train.head()))

def featuresExtraction():
    global train
    global X, Y
    text.delete('1.0', END)
    le = LabelEncoder()
    train['gender'] = pd.Series(le.fit_transform(train['gender']))
    train['NationalITy'] = pd.Series(le.fit_transform(train['NationalITy']))
    train['PlaceofBirth'] = pd.Series(le.fit_transform(train['PlaceofBirth']))
    train['StageID'] = pd.Series(le.fit_transform(train['StageID']))
    train['GradeID'] = pd.Series(le.fit_transform(train['GradeID']))
    train['SectionID'] = pd.Series(le.fit_transform(train['SectionID']))
    train['Topic'] = pd.Series(le.fit_transform(train['Topic']))
    train['Semester'] = pd.Series(le.fit_transform(train['Semester']))
    train['Relation'] = pd.Series(le.fit_transform(train['Relation']))
    train['ParentAnsweringSurvey'] =
pd.Series(le.fit_transform(train['ParentAnsweringSurvey']))
    train['ParentschoolSatisfaction'] =
pd.Series(le.fit_transform(train['ParentschoolSatisfaction']))
    train['StudentAbsenceDays'] = pd.Series(le.fit_transform(train['StudentAbsenceDays']))
    train['Class'] = pd.Series(le.fit_transform(train['Class']))
    text.insert(END,str(train.head()))



    X = train.values[:, 0:17]
    Y = train.values[:, 16]
    X = normalize(X)
    pca = PCA(n_components = 6)
    X = pca.fit_transform(X)
    n_pcs= pca.components_.shape[0]
    most_important = [np.abs(pca.components_[i]).argmax() for i in range(n_pcs)]
    initial_feature_names = train.columns
    text.insert(END,"\n\nAll Features Available in Dataset :
"+str(initial_feature_names)+"\n\n")
    most_important_names = [initial_feature_names[most_important[i]] for i in range(n_pcs)]
    text.insert(END,"Selected Important Features using PCA :
"+str(most_important_names)+"\n\n")
    df_corr = train.corr()

    df_corr[['Semester']].plot(kind='bar')
    plt.show()



def runDBSCAN():
    global X, Y
    global dbscan_main
    global dbscan_noise
    dbscan_main.clear()
```

```
dbscan_noise.clear()
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
unique_labels = set(labels)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]
index = 0
for k, col in zip(unique_labels, colors):

    if k == -1:
        col = [0, 0, 0, 1]
    class_member_mask = (labels == k)
    xy = X[class_member_mask & core_samples_mask]
    if index == 0:
        for i in range(len(X)):
            if class_member_mask[i] == True:
                dbscan_main.append(X[i])
            if class_member_mask[i] == False:
                dbscan_noise.append(i)
    index = 1

    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col), markeredgecolor='k',
markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),markeredgecolor='k',
markersize=6)

dbscan_main = np.asarray(dbscan_main)
dbscan_noise = np.asarray(dbscan_noise)
```

```
    text.delete('1.0', END)
    text.insert(END,"Total records found in dataset : "+str(len(X))+"\n")
    text.insert(END,"Total Mainstream student found by DBSCAN : "+str(len(dbscan_main))+"\n")

    text.insert(END,"Total Anomalous student found by DBSCAN :"+str(len(dbscan_noise))+"\n\n")
    text.insert(END,"Below are the Anomalous Students ID : \n\n")
    text.insert(END,str(dbscan_noise))
    plt.title('Estimated number of clusters: %d'+str(len(unique_labels)))
    plt.show()

def runKMEANS():
    text.delete('1.0', END)
    global X, Y
    global train
    global kmeans_main
    kmeans_main.clear()
    global dbscan_main
    dbscan_main = np.asarray(dbscan_main)
    kmeans = KMeans(n_clusters=10, random_state=0)
    kmeans.fit(dbscan_main)
    labels = kmeans.labels_
    clusters = unique(labels)

    for cluster in clusters:
        class_member_mask = (labels == cluster)
        kmeans_main.append(class_member_mask.nonzero()[0])
        row_ix = where(labels == cluster)
        plt.scatter(dbscan_main[row_ix, 0], dbscan_main[row_ix, 1])
    plt.show()
```

```
def visualizeClusters():
    global kmeans_main
    print(kmeans_main)
    height = []
    bars = []
    for i in range(len(kmeans_main)):
        height.append(len(kmeans_main[i]))
        bars.append("Cluster "+str(i+1))
    y_pos = np.arange(len(bars))
    plt.bar(y_pos, height)
    plt.xticks(y_pos, bars)
    plt.show()



def anomalousStudent():
    global kmeans_main
    text.delete('1.0', END)
    text.insert(END,"Students found Anomalous after applying KMEANS\n\n")
    for i in range(len(kmeans_main)):
        if len(kmeans_main[i]) < 30:
            text.insert(END,"Anomalous Students ID : "+str(kmeans_main[i])+"\n")

font = ('times', 16, 'bold')
title = Label(main, text='An Unsupervised Ensemble Clustering Approach for the Analysis of
Student Behavioral Patterns')
title.config(bg='brown', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)
font1 = ('times', 13, 'bold')
uploadButton = Button(main, text="Data Collection/Upload Dataset", command=loadData)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)
pathlabel = Label(main)
pathlabel.config(bg='brown', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=460,y=100)
fsButton = Button(main, text="Features Extraction", command=featuresExtraction)
fsButton.place(x=50,y=150)
fsButton.config(font=font1)
dbscanButton = Button(main, text="Run Initial DBSCAN Clustering",
command=runDBSCAN)
dbscanButton.place(x=330,y=150)
dbscanButton.config(font=font1)
kmeansButton = Button(main, text="Run KMeans Clustering", command=runKMEANS)
kmeansButton.place(x=630,y=150)
kmeansButton.config(font=font1)
visualButton = Button(main, text="Visualize Clusters", command=visualizeClusters)

visualButton.place(x=50,y=200)
visualButton.config(font=font1)
stdButton = Button(main, text="Display Anomalous Student ID's",
command=anomalousStudent)
stdButton.place(x=330,y=200)
stdButton.config(font=font1)
font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)
main.config(bg='brown')
main.mainloop()
```

## test.py

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.preprocessing import normalize
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from sklearn import metrics
from numpy import unique
from numpy import where
from sklearn.cluster import KMeans

train = pd.read_csv('C://Users//Rohith//Desktop//rori//Dataset//xAPI-Edu-Data.csv')
le = LabelEncoder()
train['gender'] = pd.Series(le.fit_transform(train['gender']))
train['NationalITy'] = pd.Series(le.fit_transform(train['NationalITy']))
train['PlaceofBirth'] = pd.Series(le.fit_transform(train['PlaceofBirth']))
train['StageID'] = pd.Series(le.fit_transform(train['StageID']))
train['GradeID'] = pd.Series(le.fit_transform(train['GradeID']))
train['SectionID'] = pd.Series(le.fit_transform(train['SectionID']))
train['Topic'] = pd.Series(le.fit_transform(train['Topic']))

train['Semester'] = pd.Series(le.fit_transform(train['Semester']))
train['Relation'] = pd.Series(le.fit_transform(train['Relation']))
train['ParentAnsweringSurvey'] = pd.Series(le.fit_transform(train['ParentAnsweringSurvey']))
train['ParentschoolSatisfaction'] =
pd.Series(le.fit_transform(train['ParentschoolSatisfaction']))
train['StudentAbsenceDays'] = pd.Series(le.fit_transform(train['StudentAbsenceDays']))
train['Class'] = pd.Series(le.fit_transform(train['Class']))
X = train.values[:, 0:17]
Y = train.values[:, 16]
X = normalize(X)
```

```
pca = PCA(n_components = 6)
X = pca.fit_transform(X)

n_pcs= pca.components_.shape[0]
most_important = [np.abs(pca.components_[i]).argmax() for i in range(n_pcs)]

initial_feature_names = train.columns
print(initial_feature_names)
most_important_names = [initial_feature_names[most_important[i]] for i in range(n_pcs)]
print(most_important_names)

'''
selected = pd.DataFrame(pca.components_,columns=train.columns,index = ['PC-1','PC-2','PC3','PC-4','PC-5','PC-6'])
print(selected)
print(X)



df_corr = train.corr()
df_corr[['Semester']].plot(kind='bar')
plt.show()
'''
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

print(labels)
clusters = unique(labels)
for cluster in clusters:
    row_ix = where(labels == cluster)
    temp = np.asarray(row_ix)
     plt.scatter(X[row_ix, 0], X[row_ix, 1])
    # show the plot
plt.show()
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
subclusters = []
noise_cluster = []
unique_labels = set(labels)
colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1, len(unique_labels))]
index = 0
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]
    class_member_mask = (labels == k)
    xy = X[class_member_mask & core_samples_mask]

    if index == 0:
        for i in range(len(X)):


            if class_member_mask[i] == True:
                subclusters.append(X[i])
            if class_member_mask[i] == False:
                noise_cluster.append(X[i])
    index = 1
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),markeredgecolor='k',
markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),markeredgecolor='k',
markersize=6)
plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
subclusters = np.asarray(subclusters)
```

```
print(subclusters.shape)

kmeans_result = []
kmeans = KMeans(n_clusters=10, random_state=0)
kmeans.fit(subclusters)
labels = kmeans.labels_
print(labels)
clusters = unique(labels)
for cluster in clusters:
    class_member_mask = (labels == cluster)
    kmeans_result.append(class_member_mask.nonzero()[0])
    row_ix = where(labels == cluster)
    temp = np.asarray(row_ix)

    plt.scatter(subclusters[row_ix, 0], subclusters[row_ix, 1])
# show the plot
plt.show()
```

# 6. SYSTEM TESTING

## 6.1 Implementation and Testing:

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

### Unit Testing:

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

Each module can be tested using the following two Strategies:

## 6.2 Black Box Testing:

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories: Incorrect or missing functions Interface errors in data structure or external database access Performance errors Initialization and termination errors. In this testing only the output is checked for correctness. The logical flow of the data is not checked.

Prerequisite – Software Testing | Basics Black box testing is a type of software testing in which the functionality of the software is not known. The testing is done without the internal knowledge of the products.

Black box testing can be done in the following ways:

1. Syntax Driven Testing – This type of testing is applied to systems that can be syntactically represented by some language. For example- compilers, language that can be represented by a context-free grammar. In this, the test cases are generated so that each grammar rule is used at least once.

2. Equivalence partitioning – It is often seen that many types of inputs work similarly so instead of giving all of them separately we can group them and test only one input of each

group. The idea is to partition the input domain of the system into several equivalence classes such that each member of the class works similarly, i.e., if a test case in one class results in some error, other members of the class would also result in the same error.

The technique involves two steps:

Identification of equivalence class – Partition any input domain into a minimum of two sets: valid values and invalid values. For example, if the valid range is 0 to 100 then select one valid input like 49 and one invalids like 104.

Generating test cases – (i) To each valid and invalid class of input assigns a unique identification number. (ii) Write a test case covering all valid and invalid test cases considering that no two invalid inputs mask each other. To calculate the square root of a number, the equivalence classes will be: (a) Valid inputs:

The whole number which is a perfect square- output will be an integer.

The whole number which is not a perfect square- output will be a decimal number.

Positive decimals

Negative numbers (integer or decimal).

Characters other that numbers like “a”,”!”,”;”, etc.

3. Boundary value analysis – Boundaries are very good places for errors to occur. Hence if test cases are designed for boundary values of the input domain, then the efficiency of testing improves and the probability of finding errors also increases. For example – If the valid range is 10 to 100 then test for 10,100 also apart from valid and invalid inputs.

4. Cause effect Graphing – This technique establishes a relationship between logical input called causes with corresponding actions called the effect. The causes and effects are represented using Boolean graphs. The following steps are followed:

Identify inputs (causes) and outputs (effect).

Develop a cause-effect graph.

Transform the graph into a decision table.

Convert decision table rules to test cases.

For example, in the following cause-effect graph:

It can be converted into a decision table like:

Each column corresponds to a rule which will become a test case for testing. So there will be 4 test cases.

5. Requirement-based testing – It includes validating the requirements given in the SRS of a software system.

6. Compatibility testing – The test case result not only depends on the product but is also on the infrastructure for delivering functionality. When the infrastructure parameters are changed, it is still expected to work properly. Some parameters that generally affect the compatibility of software are:

Processor (Pentium 3, Pentium 4) and several processors.
Architecture and characteristics of machine (32 bit or 64 bit).
Back-end components such as database servers.
Operating System (Windows, Linux, etc).
Tools Used for Black Box Testing:

Appium
Selenium
Microsoft Coded UI
Applitools
HP QTP.

## 6.3 White Box Testing:

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

1. Guarantee that all independent paths have been executed.
2. Execute all logical decisions on their true and false Sides.
3. Execute all loops at their boundaries and within their operational bounds
4. Execute internal data structures to ensure their validity.

Prerequisite – Software Testing | Basics

White box testing techniques analyze the internal structures the used data structures, internal design, code structure and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing.

## Working process of white box testing:

<u>Input</u>: Requirements, Functional specifications, design documents, source code.
<u>Processing</u>: Performing risk analysis for guiding through the entire process.
<u>Proper test planning</u>: Designing test cases so as to cover entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
<u>Output</u>: Preparing final report of the entire testing process.
<u>Testing techniques</u>: Statement coverage: In this technique, the aim is to traverse all statement at least once. Hence, each line of code is tested. In case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, helps in pointing out faulty code.
Minimum 2 test cases are required so that all the nodes can be traversed at least once.

Statement Coverage Example

Branch Coverage: In this technique, test cases are designed so that each branch from all decision points are traversed at least once. In a flowchart, all edges must be traversed at least once.

4 test cases required such that all branches of all decisions are covered, i.e, all edges of flowchart are covered

Condition Coverage: In this technique, all individual conditions must be covered as shown in the following example:

```
READ X, Y
IF(X == 0 || Y == 0)
PRINT '0'
```

In this example, there are 2 conditions: X == 0 and Y == 0. Now, test these conditions get TRUE and FALSE as their values. One possible example would be:

#TC1 – X = 0, Y = 55
#TC2 – X = 5, Y = 0
Multiple Condition Coverage: In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once. Let’s consider the following example:

```
READ X, Y
IF(X == 0 || Y == 0)
PRINT ‘0’
```

#TC1: X = 0, Y = 0
#TC2: X = 0, Y = 5
#TC3: X = 55, Y = 0
#TC4: X = 55, Y = 5
Hence, four test cases required for two individual conditions.
Similarly, if there are n conditions then 2n test cases would be required.

Basis Path Testing: In this technique, control flow graphs are made from code or flowchart and then Cyclomatic complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path.

Steps:
Make the corresponding control flow graph
Calculate the cyclomatic complexity
Find the independent paths
Design test cases corresponding to each independent path
Flow graph notation: It is a directed graph consisting of nodes and edges. Each node represents a sequence of statements, or a decision point. A predicate node is the one that represents a decision point that contains a condition after which the graph splits. Regions are bounded by nodes and edges.

Cyclomatic Complexity: It is a measure of the logical complexity of the software and is used to define the number of independent paths. For a graph G, V(G) is its cyclomatic complexity.
Calculating V(G):

V(G) = P + 1, where P is the number of predicate nodes in the flow graph
V(G) = E – N + 2, where E is the number of edges and N is the total number of nodes
V(G) = Number of non-overlapping regions in the graph
Example:

V(G) = 4 (Using any of the above formulae)
No of independent paths = 4

#P1: 1 – 2 – 4 – 7 – 8
#P2: 1 – 2 – 3 – 5 – 7 – 8
#P3: 1 – 2 – 3 – 6 – 7 – 8
#P4: 1 – 2 – 4 – 7 – 1 – . . . – 7 – 8

**Loop Testing:** Loops are widely used and these are fundamental to many algorithms hence, their testing is very important. Errors often occur at the beginnings and ends of loops.
Simple loops: For simple loops of size n, test cases are designed that:
Skip the loop entirely
Only one pass through the loop
2 passes
m passes, where m < n
n-1 ans n+1 passes

**Nested loops:** For nested loops, all the loops are set to their minimum count and we start from the innermost loop. Simple loop tests are conducted for the innermost loop and this is worked outwards till all the loops have been tested.
Concatenated loops: Independent loops, one after another. Simple loop tests are applied for each.
If they're not independent, treat them like nesting.

**Integrating Testing:**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

Integration testing -- also known as integration and testing (I&T) -- is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers.
The aim of integration testing is to test the interfaces between the modules and expose any defects that may arise when these components are integrated and need to interact with each other.

Also known as *string testing* or *thread testing*, integration testing involves integrating the various modules of an application and then testing their behaviour as a combined, or integrated, unit. Verifying if the individual units are communicating with each other properly and working as intended post-integration is essential.

To perform integration testing, testers use test drivers and stubs, which are dummy programs that act as substitutes for any missing modules and simulate data communications between modules for testing purposes. They also use integration-testing tools, including the following:

- Selenium is an open-source suite that facilitates automated web application testing.
- Pytest is a Python testing tool which, according to Pytest.org, enables small tests to be written easily "yet scales to support complex functional testing for applications and libraries."
- IBM's RFT (Rational Functional Tester) is an object-oriented automated functional testing tool for performing automated functional, regression, GUI and data-driven testing.
- Junit is an open source framework designed for writing and running tests for Java and Java Virtual Machine (JVM).
- Mockito is an open-source testing framework for Java.
- Jasmine is a development framework for testing JavaScript
- FitNesse is an open-source framework in which software testers, developers and customers can work together to build test cases on a wiki.

- Steam, developed by GitHub, is an open-source framework used to test JavaScript-enabled websites.
- LDRA tools are used for integration testing for organizations requiring verification to compliance standards.

Integration testing is usually done simultaneously with development. But this can create a challenge if the modules to be tested are not yet available.

Integration testing is conducted after unit testing, where the functional correctness of the smallest piece of code, or *unit*, is tested. Each unit can be logically isolated in the software. The smaller the unit, the more granular insights unit testing can reveal.
The main difference between unit testing and integration testing is that in unit testing, individual modules are tested. In integration testing, these modules are combined and tested as a single unit to check the functionality of the overall application.

**System Testing:**

In order to deliver a product that will successfully fulfill the demands of the stakeholders, it is important that no unexpected behaviour occurs while it is in use. Hence, thorough testing must be done in order to find as many potential issues as possible before the product is rolled out to stakeholders. For this reason, a plan for testing was made and executed, which will be described in this chapter. Testing of software can generally be split up into several categories, from low to high level. Namely unit testing, integration/system testing and usability testing. Although naturally the chronological order of these categories within the project development cycle is to some extent the same, in the project workflow, this chronological order is not so strict and all categories are to some extent tested from the very start. This is done both in order to show visual progress in a reasonable time to the stakeholders and to catch wider problems (for example, a bad user interface) before it is too late to change significantly. Firstly, unit testing is about testing individual components (units) in isolation, often automated. This allows for the discovery of crucial bugs early in the development process and also makes the next forms of testing easier, as it is much harder to trace the causes of bugs in higher level testing. The downsides of this form of testing is that bugs relating to a connection between components comprehensive unit testing was somewhat restricted to only the most complex components.

**Automated Testing:**

With the back end, unit testing (and partially integration testing) was suitable for service classes, which implement the business logic of the application. Most of the dependencies that were there were mocked; what this means is that the dependent object is not actually created, but instead a version where every function returns null is used. Of course, testing would not be possible if every dependent function just returned null, so a technique called stubbing is used. This means manually giving return values to a mocked object’s functions given certain parameters, allowing for simulating the function actually being called. An example where this is useful is in database retrieval, where without actually making use of the database the desired object is pretended to be retrieved from it using stubbing. For mocking the Mockito library was used. This, in conjuction with JUnit, Java’s unit testing framework and AssertJ for more readable and helpful assertions, provided everything needed for unit testing the application. Most tests were similar, and can be summarized in four steps. First, the test models are created (e.g., a rubric to test something rubric related), then any expected return values are defined, next the mocked methods are stubbed and finally the return value of the tested function is compared to the expected value. In some cases, there were other components that could be compared, for example a certain field in a test model which should have been changed by the function. Furthermore, there were some cases where an exception should be thrown, and this was then asserted and its messages checked to correspond to the expected message. For void methods sometimes the opposite was the most useful, checking instead that no exception was thrown given intended input variables.

**Test Approach:**

Testing can be done in two ways: Bottom-up approach & Top-down approach

**Bottom-up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower-level ones they are tested individually and then linked with the previously examined lower-level modules.

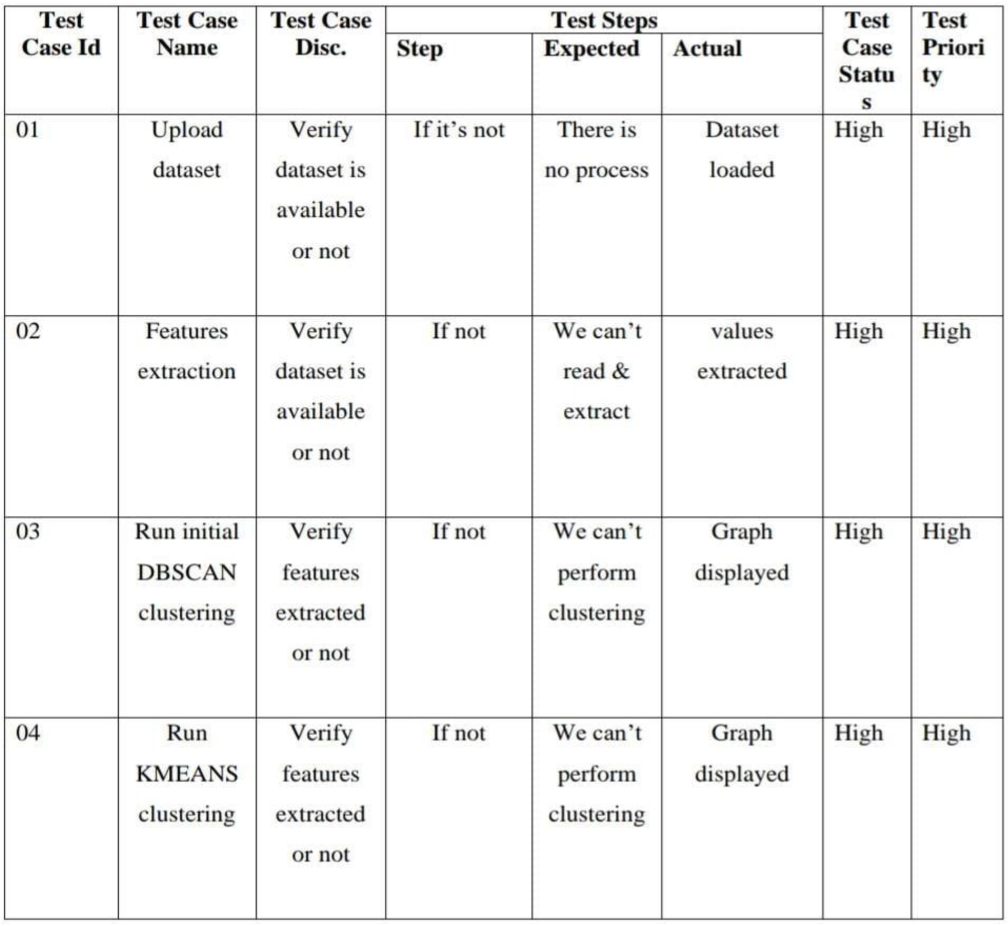
**Top-down approach:**

This type of testing starts from upper-level modules. Since the detailed activities usually performed in the lower-level routines are not provided stubs are written. A stub is a module shell called by upper-level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower-level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

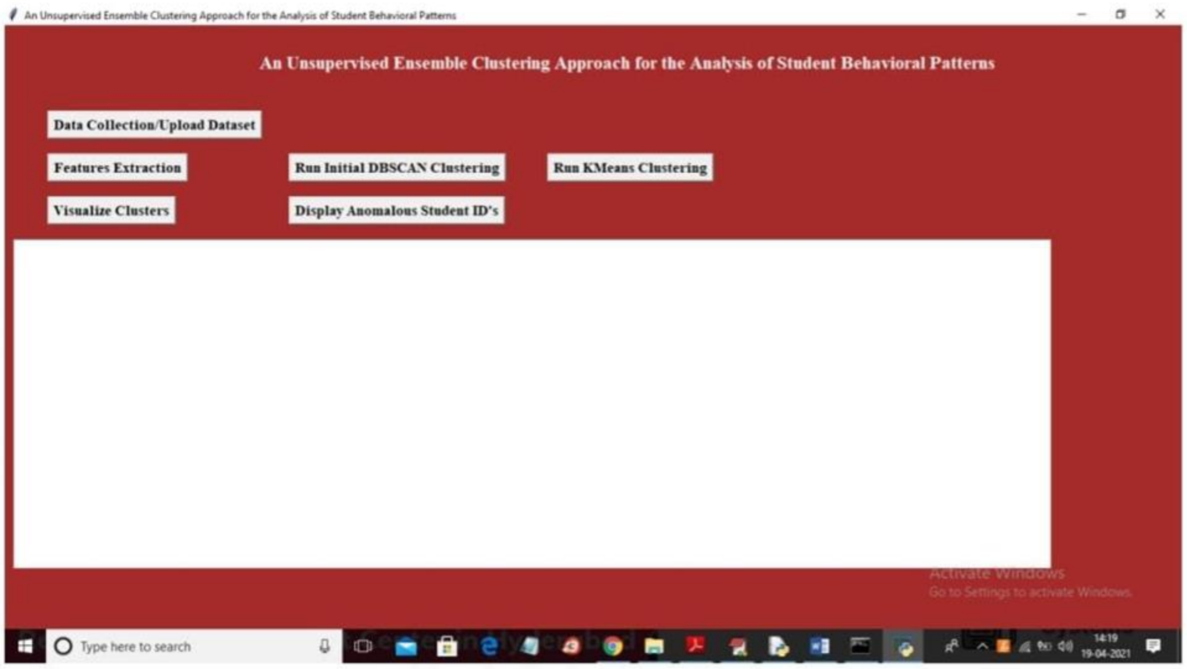
**Test cases**



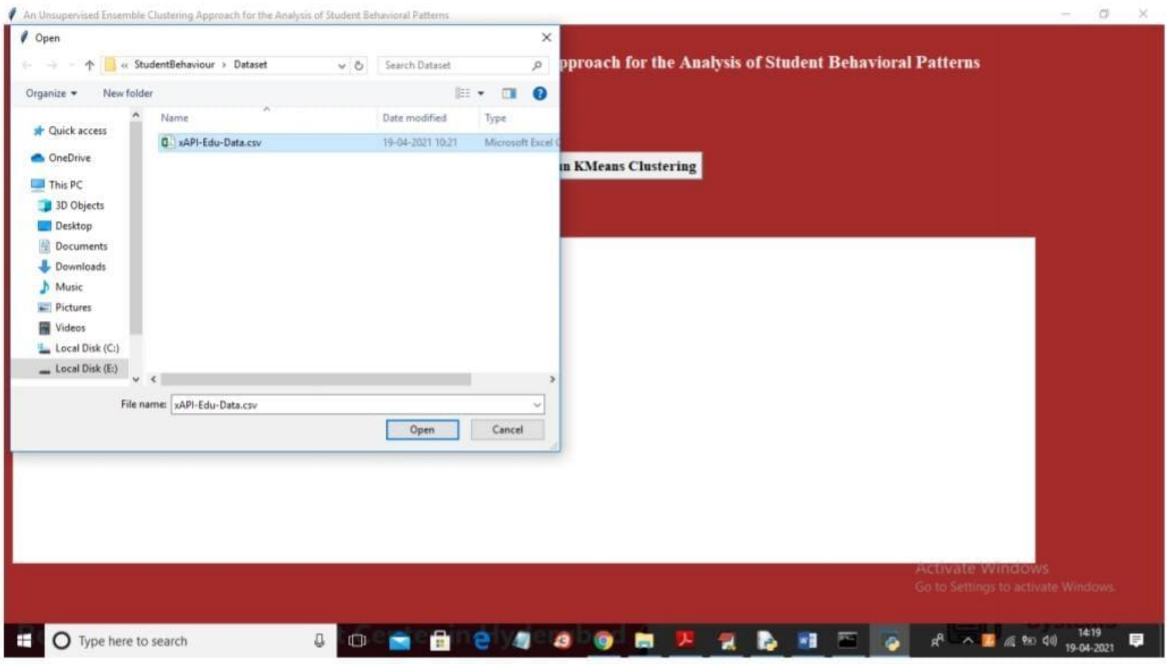
| Test Case Id | Test Case Name | Test Case Disc. | Test Steps | | | Test Case Status | Test Priority |
|---|---|---|---|---|---|---|---|
| | | | Step | Expected | Actual | | |
| 01 | Upload dataset | Verify dataset is available or not | If it's not | There is no process | Dataset loaded | High | High |
| 02 | Features extraction | Verify dataset is available or not | If not | We can't read & extract | values extracted | High | High |
| 03 | Run initial DBSCAN clustering | Verify features extracted or not | If not | We can't perform clustering | Graph displayed | High | High |
| 04 | Run KMEANS clustering | Verify features extracted or not | If not | We can't perform clustering | Graph displayed | High | High |

# 7. RESULT SCREENSHOTS

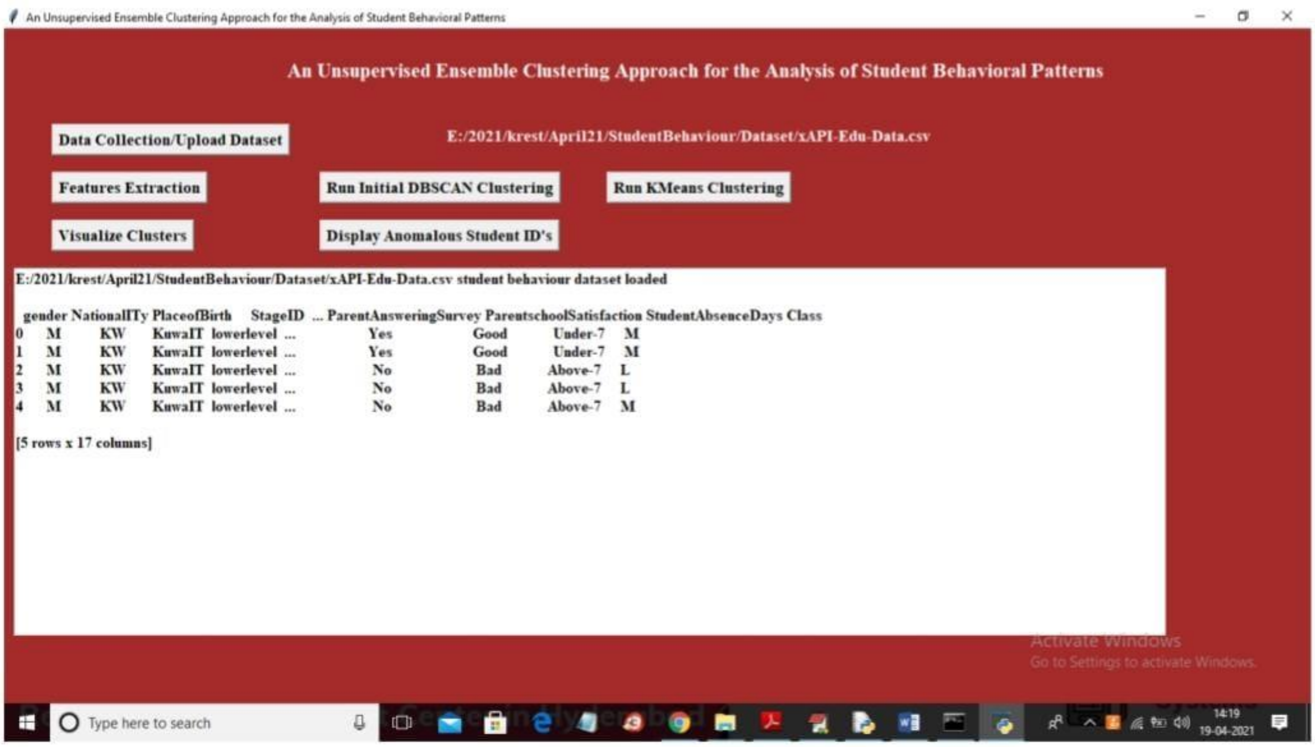
To run project double, click on ‘run.bat’ file to get below screen


An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
Data Collection/Upload Dataset
Features Extraction
Run Initial DBSCAN Clustering
Run KMeans Clustering
Visualize Clusters
Display Anomalous Student ID's

In above screen click on ‘Data Collection/Upload Dataset’ button to upload dataset

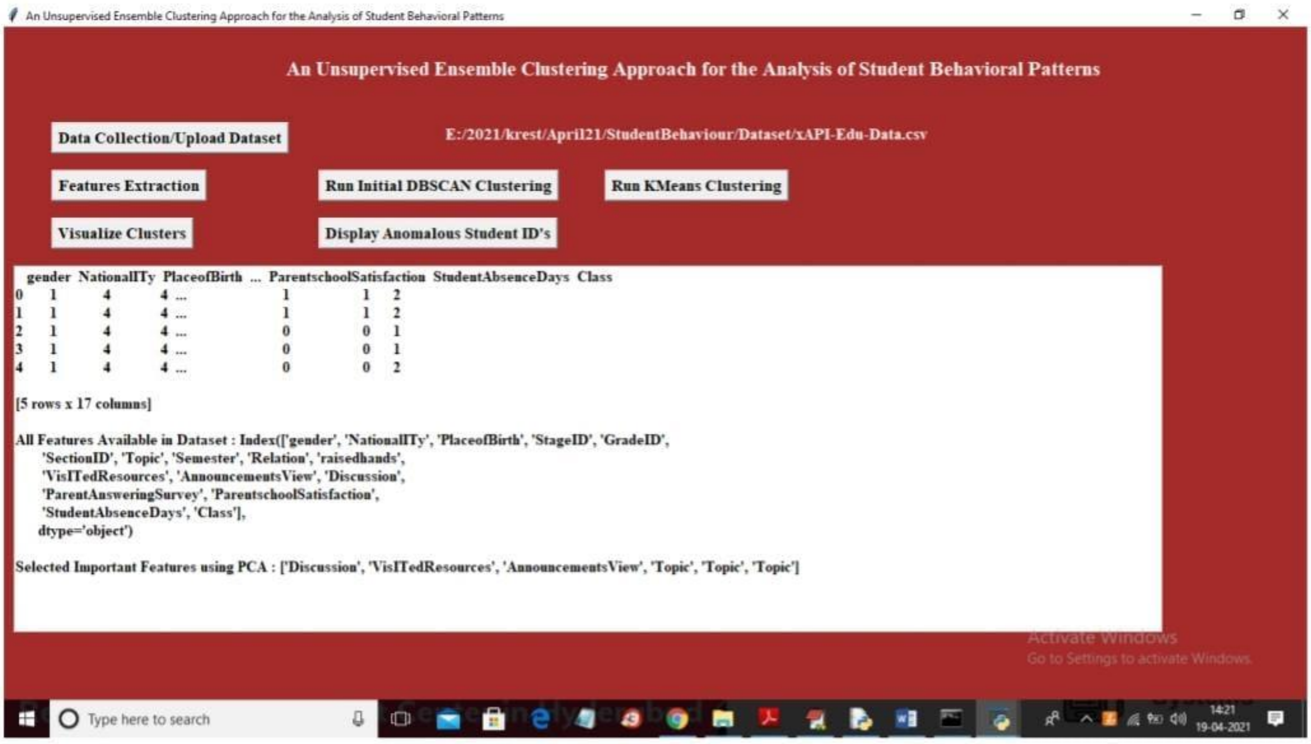


In above screen select and upload dataset file to get below screen


An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
Data Collection/Upload Dataset
E:/2021/krest/April21/StudentBehaviour/Dataset/xAPI-Edu-Data.csv
Features Extraction
Run Initial DBSCAN Clustering
Run KMeans Clustering
Visualize Clusters
Display Anomalous Student ID's

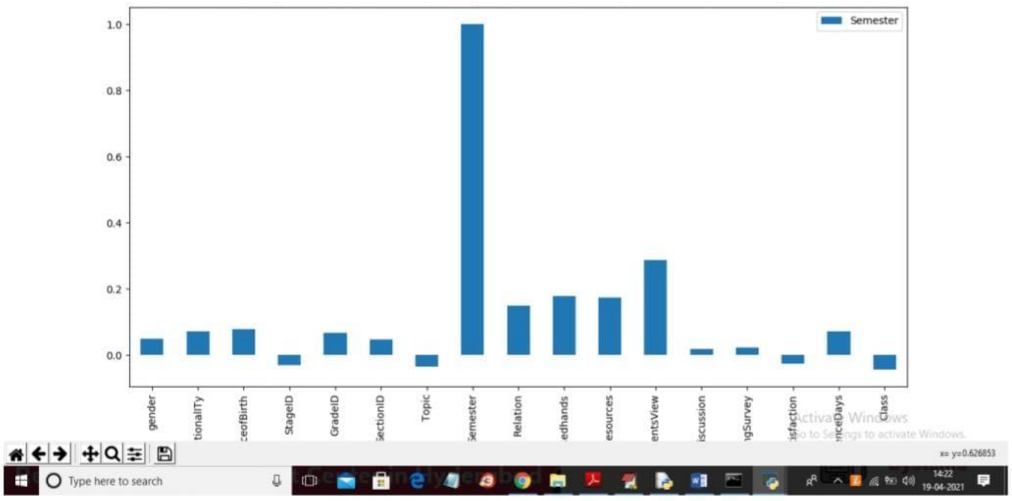
```
E:/2021/krest/April21/StudentBehaviour/Dataset/xAPI-Edu-Data.csv student behaviour dataset loaded

 gender NationalITy PlaceofBirth  StageID ... ParentAnsweringSurvey ParentschoolSatisfaction StudentAbsenceDays Class
0   M       KW      KuwaIT  lowerlevel ...        Yes              Good        Under-7    M
1   M       KW      KuwaIT  lowerlevel ...        Yes              Good        Under-7    M
2   M       KW      KuwaIT  lowerlevel ...        No               Bad         Above-7    L
3   M       KW      KuwaIT  lowerlevel ...        No               Bad         Above-7    L
4   M       KW      KuwaIT  lowerlevel ...        No               Bad         Above-7    M

[5 rows x 17 columns]
```

In above screen dataset loaded but it contains lots of non-numeric values so click on ‘Features Extraction’ button to convert non-numeric values to numeric and then apply PCA to select important features

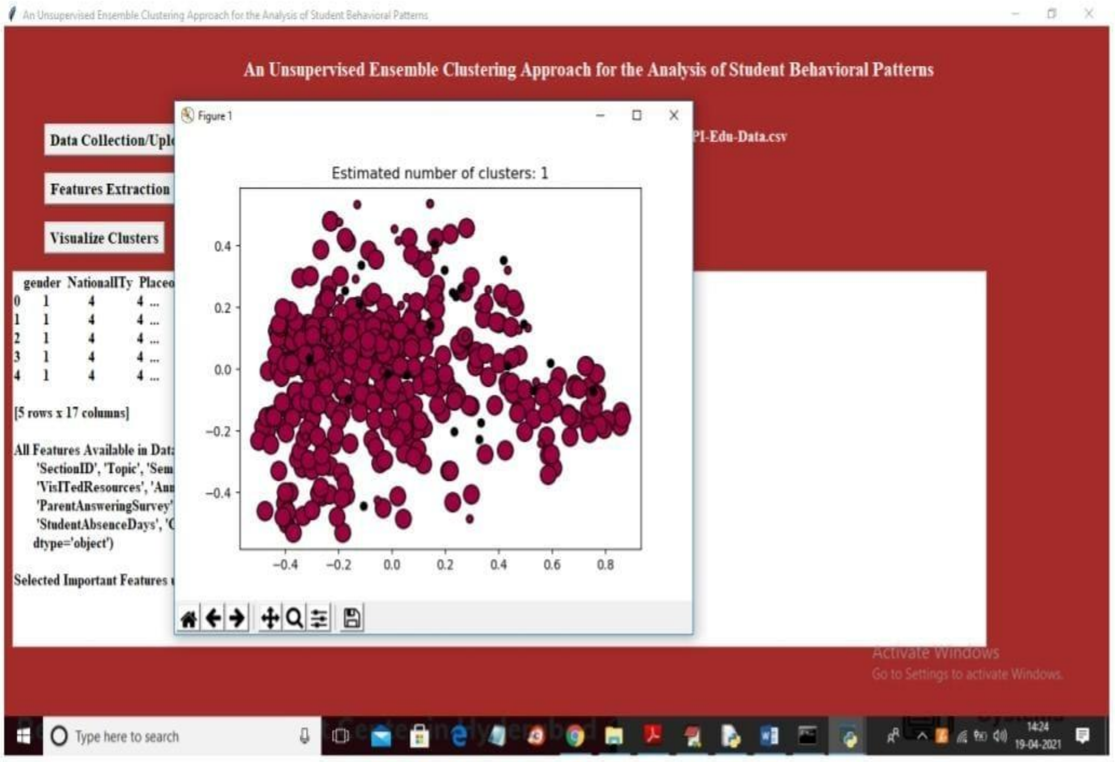

An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
Data Collection/Upload Dataset
E:/2021/krest/April21/StudentBehaviour/Dataset/xAPI-Edu-Data.csv
Features Extraction
Run Initial DBSCAN Clustering
Run KMeans Clustering
Visualize Clusters
Display Anomalous Student ID's
Type here to search

```
   gender  NationalITy  PlaceofBirth  ...  ParentschoolSatisfaction  StudentAbsenceDays  Class
0       1            4             4  ...                         1                   1      2
1       1            4             4  ...                         1                   1      2
2       1            4             4  ...                         0                   0      1
3       1            4             4  ...                         0                   0      1
4       1            4             4  ...                         0                   0      2

[5 rows x 17 columns]

All Features Available in Dataset : Index(['gender', 'NationalITy', 'PlaceofBirth', 'StageID', 'GradeID',
       'SectionID', 'Topic', 'Semester', 'Relation', 'raisedhands',
       'VisITedResources', 'AnnouncementsView', 'Discussion',
       'ParentAnsweringSurvey', 'ParentschoolSatisfaction',
       'StudentAbsenceDays', 'Class'],
      dtype='object')

Selected Important Features using PCA : ['Discussion', 'VisITedResources', 'AnnouncementsView', 'Topic', 'Topic', 'Topic']
```

In above screen we can see all values are converted to numeric and we can see dataset contains total 17 columns or attributes and after applying PCA we got 6 features and the selected 6 features names you can see in above screen and while applying PCA we got feature variance graph for each features and this graph you can see in below screen

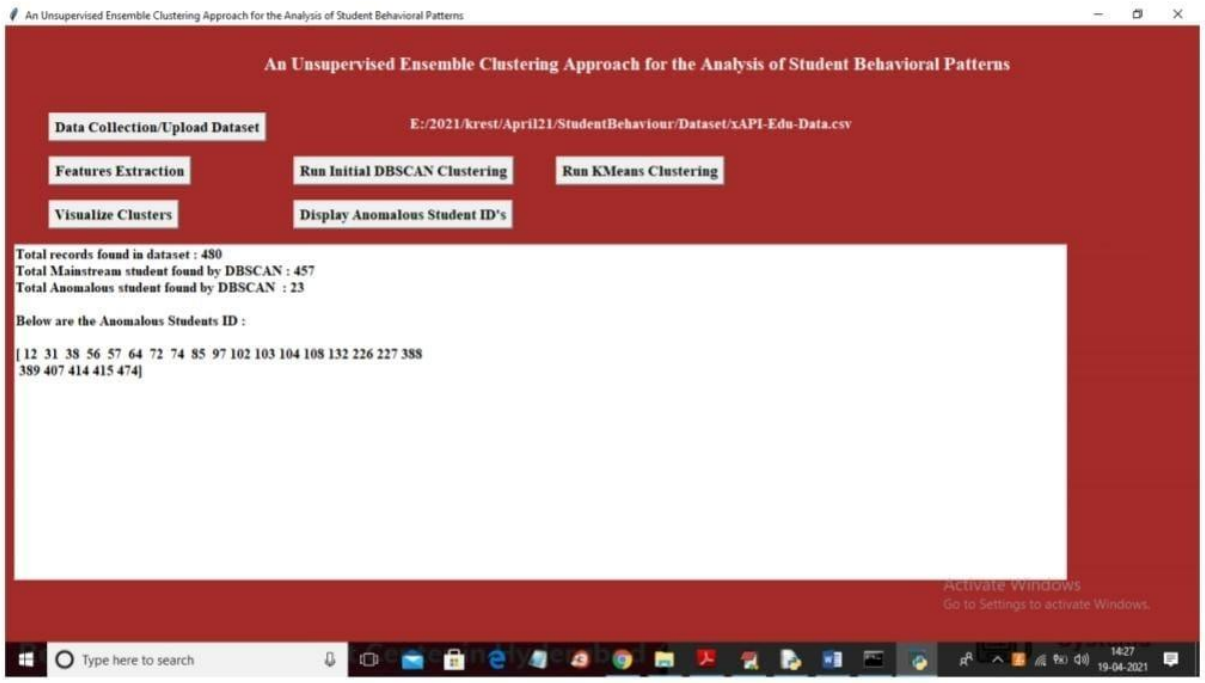


In above graph x-axis represents features names and y-axis represents importance value between 0 and 1 and if feature is important then its value will be closer to 1 else 0. Now close above graph and click on ‘Run Initial DBSCAN Clustering’ button to perform clustering and to get below graph


An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
Figure 1
Data Collection/Upl
Features Extraction
Visualize Clusters
PI-Edu-Data.csv
Estimated number of clusters: 1
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Type here to search
14:24
19-04-2021

```
   gender  NationalITy  Placeo
0    1         4          4 ...
1    1         4          4 ...
2    1         4          4 ...
3    1         4          4 ...
4    1         4          4 ...

[5 rows x 17 columns]

All Features Available in Data
      'SectionID', 'Topic', 'Sem
      'VisITedResources', 'Ann
      'ParentAnsweringSurvey'
      'StudentAbsenceDays', 'C
     dtype='object')

Selected Important Features
```

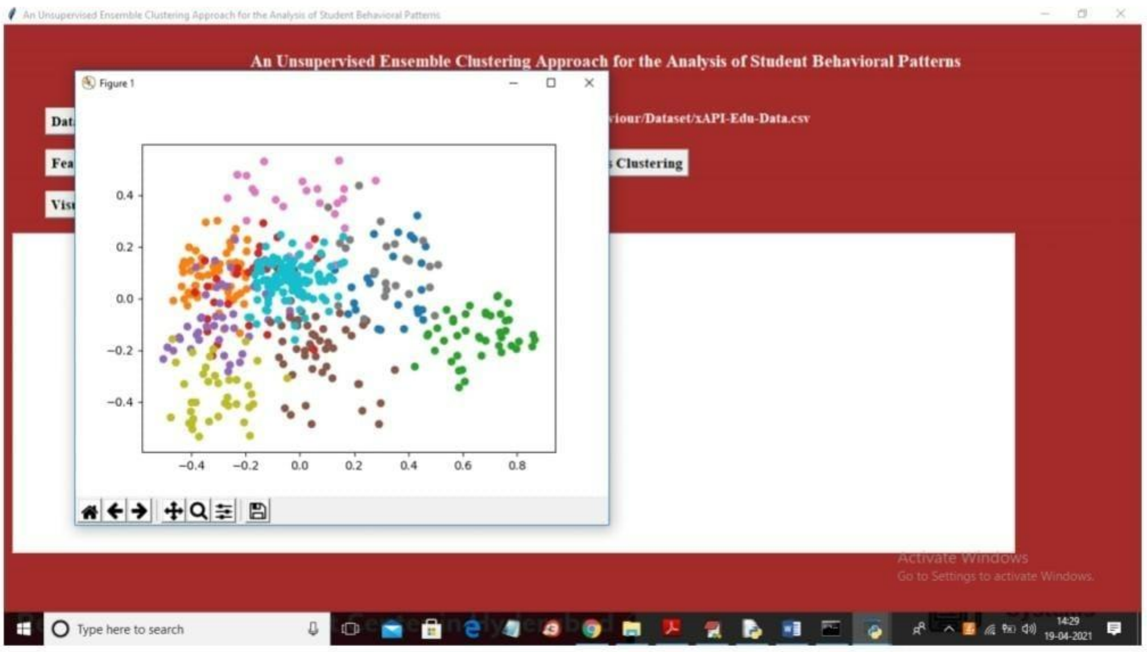
In above graph small black dots are the noise cluster records and big dots are main single cluster which contains normal behaviour and now close above graph to get below screen


An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
Data Collection/Upload Dataset
Features Extraction
Run Initial DBSCAN Clustering
Run KMeans Clustering
Visualize Clusters
Display Anomalous Student ID's
Type here to search

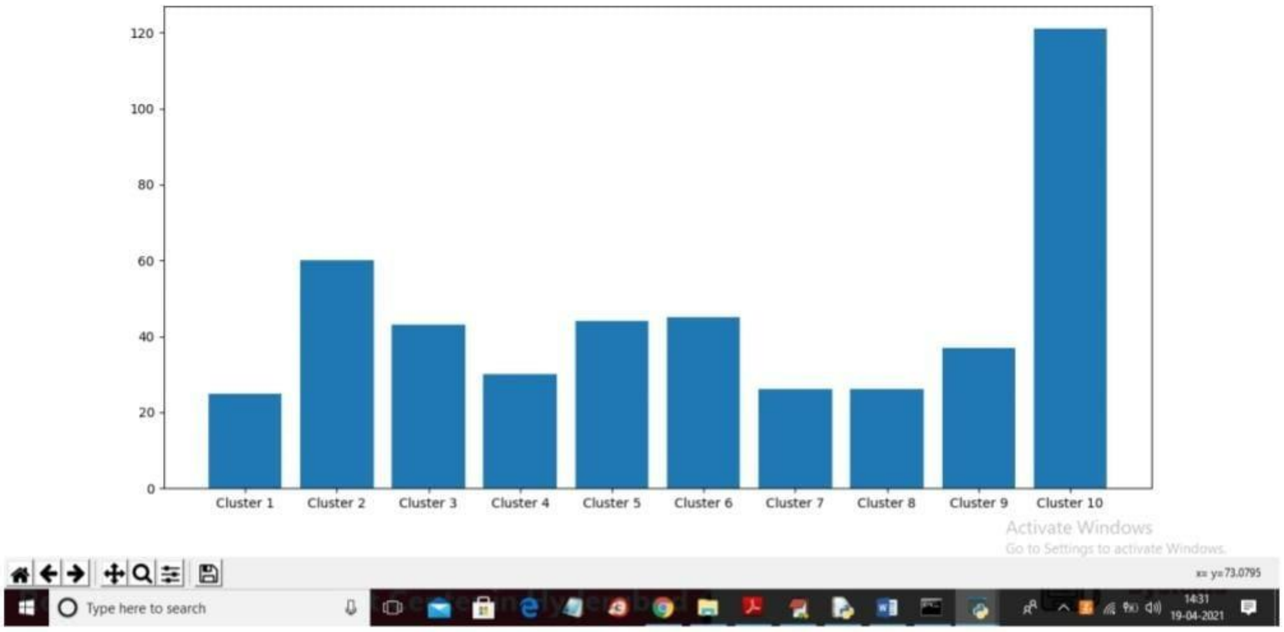
E:/2021/krest/April21/StudentBehaviour/Dataset/xAPI-Edu-Data.csv

```
Total records found in dataset : 480
Total Mainstream student found by DBSCAN : 457
Total Anomalous student found by DBSCAN  : 23

Below are the Anomalous Students ID :

[ 12  31  38  56  57  64  72  74  85  97 102 103 104 108 132 226 227 388
 389 407 414 415 474]
```

In above screen we can see total records contains in dataset and total number of records groups into DBSCAN main cluster and total records into noise clusters and in above screen we can see record IDS of abnormal behaviour students and now click on ‘Run KMEANS Clustering’ button to apply KMEANS to main single DBSCAN cluster to get below graph


An Unsupervised Ensemble Clustering Approach for the Analysis of Student Behavioral Patterns
Figure 1
0.4
0.2
0.0
−0.2
−0.4
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
Type here to search

In above KMEANS graph we can see 10 different colours dots which means 10 different clusters are created and then cluster with least students will be consider as ANOMALOUS and now click on ‘Visualize Clusters’ graph to get below graph of KMEANS



In above graph x-axis represents cluster no and y-axis represents number of students in that cluster and the cluster with least students will be identified as ANOMALOUS and now close above graph and click on ‘Display Anomalous Student ID's’ button to get all anomalous students ID

In above graph we can see 3 different clusters in square brackets which contains least records and consider as ANOMLAOUS so out of 10 clusters above 3 clusters are ANOMALOUS or noise cluster. Inside square bracket we can see student id and this id you can see dataset record numbers.

# 8. CONCLUSION

We proposed an ensemble unsupervised clustering framework for the analysis of students' behavioral patterns by combining DBSCAN and k-means algorithms. To evaluate the effect of the proposed method, we collect six types of behavioral data produced by 9024 undergraduates on campus and extract behavioral features through the two aspects of statistics and entropy. The experimental results demonstrate that the proposed method can not only detect anomalous behavioral patterns but also more precisely identify mainstream behavioral patterns. Based on the clustering results, student departments can adopt more targeted measures for intervention and specialized services. At the end of the paper, we discuss three issues: whether we can cluster the behavioral feature space using only the k-means algorithm, the difficulty of applying the proposed method to high-dimensional multisource behavior features, and the relationship between different behavioral patterns and academic performance levels.

## 8.1 FUTURE SCOPE

- Today in this competitive world every work has been become computerized. The manual way of working has become very hectic, time consuming as well as very difficult. So when I got opportunity to make the system of "School Management System" Computerized I made use of it.

- According Schools, all task done manually weather adding students information, teacher information, fees structure any relevant information.Manually added by one or more people, this make burden for saving information.

- With the existing scenario the work become very time consuming and needed more people and facing several problems during work.

- In existing system fees collection mechanism is very difficult to collect each student fees and generate report manually.

The Online Grading System contains the necessary features to support the assignment management system. For the future course of the project,

1. A tree structure can be implemented to better manage the files across different semesters and years. This can also be used to add or delete a folder and upload/download files so that other files besides assignments can be uploaded and saved for a corresponding semester by the professors.

2. Currently the Online Grading System supports C, C++ and Java programming assignments. In the future more programming languages can be supported.

3. Further enhancements to the Online Grading System such as approval method can be implemented.

# 9. BIBLIOGRAPHY

[1] A. H. Eliasson, C. J. Lettieri, and A. H. Eliasson, ‘‘Early to bed, early to rise! Sleep habits and academic performance in college students,’’ Sleep Breathing, vol. 14, no. 1, pp. 71–75, Feb. 2010, doi: 10.1007/s11325-009- 0282-2.

[2] X. D. Keating, D. Castelli, and S. F. Ayers, ‘‘Association of weekly strength exercise frequency and academic performance among students at a large university in the united states,’’ J. Strength Conditioning Res., vol. 27, no. 7, pp. 1988–1993, Jul. 2013, doi: 10.1519/JSC.0b013e318276bb4c.

[3] M. Valladares, E. Duran, A. Matheus, S. Duran-Agueero, A. M. Obregon, and R. RamirezTagle, ‘‘Association between eating behavior and academic performance in university students,’’ J. Amer. College Nutrition, vol. 35, no. 8, pp. 699–703, 2016, doi: 10.1080/07315724.2016.1157526.

[4] J. Filippou, C. Cheong, and F. Cheong, ‘‘Modelling the impact of study behaviours on academic performance to inform the design of a persuasive system,’’ Inf. Manage., vol. 53, no. 7, pp. 892–903, Nov. 2016, doi: 10.1016/j.im.2016.05.002.

[5] S. Ghosh and S. K. Ghosh, ‘‘Exploring the association between mobility behaviours and academic performances of students: A context-aware trajgraph (CTG) analysis,’’ Prog. Artif. Intell., vol. 7, no. 4, pp. 307–326, Dec. 2018, doi: 10.1007/s13748-018-0164-6.

[6] Z. Yang, X. Mo, D. Shi, and R. Wang, ‘‘Mining relationships between mental health, academic performance and human behaviour,’’ in Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, Aug. 2017, pp. 1–8.

[7] T. Phan, S. G. McNeil, and B. R. Robin, ‘‘Students’ patterns of engagement and course performance in a massive open online course,’’ Comput. Edu., vol. 95, pp. 36–44, Apr. 2016, doi: 10.1016/j.compedu.2015.11.015

[8] I. JO, Y. Park, J. Kim, and J. Song, ‘‘Analysis of online behavior and prediction of learning performance in blended learning environments,’’ Educ. Technol. Int., vol. 15, no. 2, pp. 71–88, 2014.

[9] G. Kostopoulos, S. Kotsiantis, N. Fazakis, G. Koutsonikos, and C. Pierrakeas, ‘‘A semisupervised regression algorithm for grade prediction of students in distance learning courses,’’ Int. J. Artif. Intell. Tools, vol. 28, no. 4, Jun. 2019, Art. no. 1940001, doi: 10.1142/ S0218213019400013.

[10] D. Hooshyar, M. Pedaste, and Y. Yang, ‘‘Mining educational data to predict Students’ performance through procrastination behavior,’’ Entropy, vol. 22, no. 1, p. 12, Dec. 2019, doi: 10.3390/e2201001.