A

PROJECT REPORT ON

Academic Performance Prediction Based on Multisource, Multifeature Behavioral Data

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

By

P. LALITHA

(23BFF00089)

Under the esteemed guidance of

Dr. N. Sudhakar Reddy, Principal,

M.Tech, P.hD, FIE, FIETE, MISTE.

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

SRI VENKATESWARA COLLEGE OF ENGINEERING (AUTONOMOUS)

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)

Accredited by NAAC with 'A' Grade

Opp. LIC Training Centre, Karakambadi Road, TIRUPATI–517507

2023-2025

***

SRI VENKATESWARA COLLEGE OF ENGINEERING (AUTONOMOUS)

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)

Accredited by NAAC with 'A' Grade

Opp. LIC Training Centre, Karakambadi Road, TIRUPATI–517507

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

2023-2025

CERTIFICATE

This is to certify that the project report entitled "Academic Performance Prediction Based on Multisource, Multifeature Behavioral Data" is a bonafide record of the project work done and submitted by

P. LALITHA

(23BFF00089)

for the partial fulfillment of the requirements for the award of **MASTER OF COMPUTER APPLICATIONS** *Degree from SRI VENKATESWARA COLLEGE OF ENGINEERING (AUTONOMOUS) Affiliated to JNT University Anantapur.*

GUIDE **HEAD OF THE DEPARTMENT**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

I am thankful to my guide **Dr. N. Sudhakar Reddy**, Principal for his valuable guidance and encouragement. His helping attitude and suggestions have helped me in the successful completion of the project**.**

I would like to express my sincere thanks to **Dr. E. Sreedevi,** Professor, Head of the Department of MCA, for her kind help and encouragement during the course of my study and in the successful completion of the project work.

I express my sincere thanks to **Dr. N. Sudhakar Reddy**, Principal, S.V College of Engineering, Tirupati.

Successful completion of any project cannot be done without proper support and encouragement. My sincere thanks to the Management for providing all the necessary facilities during the Course of my study.

I would like to thank my parents and friends, who have the greatest contributions in all my achievements, for the great care and blessings in making me successful in all my endeavors.

I would like to express my deep gratitude to all those who helped me directly or indirectly to transform an idea into my working project.

P. LALITHA

(23BFF00089)

DECLARATION

I hereby declare that project entitled **"Academic Performance Prediction Based on Multisource, Multifeature Behavioral Data"** submitted to the Department of COMPUTER APPLICATIONS, **SRI VENKATESWARA COLLEGE OF ENGINEERING, TIRUPATI** in partial fulfillment of Requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS.**

This project is the result of my own effort and it has not been submitted to any other University or Institution for the award of any degree other than specified above.

SIGNATURE OF THE STUDENT

ABSTRACT

Digital data trails from disparate sources covering different aspects of student life are stored daily in most modern university campuses. However, it remains challenging to (i) combine these data to obtain a holistic view of a student, (ii) use these data to accurately predict academic performance, and (iii) use such predictions to promote positive student engagement with the university. To initially alleviate this problem, in this paper, a model named Augmented Education (AugmentED) is proposed. In our study, (1) first, an experiment is conducted based on a real-world campus dataset of college students (N = 156) that aggregates multisource behavioral data covering not only online and offline learning but also behaviors inside and outside of the classroom. Specifically, to gain in-depth insight into the features leading to excellent or poor performance, metrics measuring the linear and nonlinear behavioral changes (e.g., regularity and stability) of campus lifestyles are estimated; furthermore, features representing dynamic changes in temporal lifestyle patterns are extracted by the means of long short-term memory (LSTM). (2) Second, machine learning-based classification algorithms are developed to predict academic performance. (3) Finally, visualized feedback enabling students (especially at-risk students) to potentially optimize their interactions with the university and achieve a study-life balance is designed. The experiments show that the AugmentED model can predict students' academic performance with high accuracy.

TABLE OF CONTENTS

LISTOF FIGURES

1. INTRODUCTION

1.1 GENERAL

As an important step to achieving personalized education, academic performance prediction is a key issue in the education data mining field. It has been extensively demonstrated that academic performance can be profoundly affected by the following factors:

- Students' Personality (e.g., neuroticism, extraversion, and agreeableness)
- Personal Status (e.g., gender, age, height, weight, physical fitness, cardiorespiratory fitness, aerobic fitness, stress, mood, mental health, intelligence, and executive functions)
- Lifestyle Behaviors (e.g., eating, physical activity, sleep patterns, social tie, and time management)
- Learning Behaviors (e.g., class attendance, study duration, library entry, and online learning)

For example, investigated the incremental validity of the Big Five personality traits in predicting college GPA. Demonstrated that physical fitness in boys and obesity status in girls could be important factors related to academic achievement. Meanwhile, showed that a regular lifestyle could lead to good performance among college students. Showed that the degree of effort exerted while working could be strongly correlated with academic performance.

Additionally, showed that compared with high- and medium-achieving students, low-achieving students were less emotionally engaged throughout the semester and tended to express more confusions during the final stage of the semester. By analyzing the effect of the factors influencing academic performance, many systems using data to predict academic performance have been developed in the literature.

For instance, academic performance was predicted based on passive sensing data and self-reports from students' smart phones. a multitask predictive framework that captures intersemester and intermajor correlations and integrates student similarity was built to predict students' academic performance, based on homework submission data, the academic

performance of students enrolled in a blended learning course was predicted. According to their predicted academic performance, early feedbacks and interventions could be individually applied to at-risk students.

According to their predicted academic performance, early feedbacks and interventions could be individually applied to at-risk students. For example, to help students with a low GPA, basic interventions are defined based on GPA predictions. However, the research on the feedback/intervention is still in the early stage, its achievements are relatively few.

Although many academic performance prediction systems have been developed for college students, the following challenges persist:

- capturing a sufficiently rich profile of a student and integrating these data to obtain a holistic view;
- exploring the factors affecting students' academic performance and using this information to develop a robust prediction model with high accuracy; and
- taking advantage of the prediction model to deliver personalized services that potentially enable students to drive behavioral change and optimize their study-life balance

1.2 OBJECTIVE OF THE PROJECT:

Objectives for the project "Academic Performance Prediction Based on Multisource, Multifeature Behavioral Data":

- **Data Acquisition and Integration:** To collect and integrate diverse behavioral data virtual learning environments).

- **Feature Engineering:** To extract a comprehensive set of meaningful behavioral features from the integrated data, capturing aspects of engagement, interaction, learning strategies, and performance indicators.

- **Feature Selection:** To identify the most predictive subset of engineered features using appropriate feature selection techniques.

- **Model Development:** To develop and train various machine learning models (e.g., regression, classification, deep learning) capable of accurately predicting student academic performance.

- **Model Evaluation:** To rigorously evaluate the performance of the developed models using appropriate metrics and benchmark them against existing methods.

- **Interpretability and Explainability:** To explore and apply techniques that enhance the interpretability of the prediction models, providing insights into the key behavioral factors influencing academic outcomes.

- **Identification of Key Predictors:** To identify and analyse the most significant behavioral features that correlate with and predict academic success or failure.

- **Prototype System Development (Optional):** To design and potentially develop a prototype system that can ingest new student behavioral data and generate predictions.

1.3 DESCRIPTION OF THE PROJECT:

Predicting student academic performance is a critical endeavor in educational institutions, enabling timely interventions, personalized learning strategies, and efficient resource allocation. Traditional methods often rely on limited historical data, failing to capture the dynamic and multifaceted nature of student learning behaviors across diverse digital platforms. The increasing availability of data from Learning Management Systems (LMS), online assessment platforms, virtual learning environments, and potentially student activity logs presents a unique opportunity to develop more robust and accurate predictive models.

This project aims to leverage this wealth of multisource, multifeature behavioral data to create a holistic understanding of the factors influencing student success. By integrating data from various learning environments and extracting meaningful behavioral indicators related to engagement, interaction, learning strategies, and performance within each platform, we aim to overcome the limitations of traditional approaches and build a more nuanced and predictive framework for academic outcomes.

This project will involve a comprehensive methodology encompassing data acquisition and integration from disparate sources, followed by rigorous feature engineering and selection to identify the most predictive behavioral patterns. We will develop and compare various machine learning models, including regression, classification, and potentially deep learning techniques, to predict academic performance metrics such as final grades, course completion, and risk of failure.

A key focus will be on model interpretability, employing explainable AI (XAI) methods to understand the contribution of different behavioral features and provide actionable insights into the drivers of student success. The potential impact of this project includes improved accuracy in identifying at-risk students, providing educators with data-driven insights for personalized learning, and contributing to the advancement of educational data mining.

2. LITERATURE SURVEY

1) Personality traits and intelligence predict academic school grades

AUTHORS: A. Furnham, and J. Monsen

This study examines the extent to which personality traits and intelligence scores predict school level academic performance (AP), (British GCSE: General Certificate of Secondary Education; America Grade 10) in different disciplines. The participant sample consisted of approximately 250 school pupils from three schools in the South East of England. A series of hierarchical regressions were performed with participant discipline-specific subject grades being the criterion variable and demographic, as well and intelligence and personality test scores, the predictor variables. For overall grade intelligence accounted for a fifth of the variance and personality an incremental validity of 8%. Whilst a combination of intelligence, personality and sex accounted for around a quarter of the variance in all four core subjects the pattern was rather different for elected subjects. The results are discussed in terms of the usefulness of psychometric assessments of candidates at selection.

2) Aptitude is not enough: How personality and behavior predict academic performance

AUTHORS: M. A. Conard

The study investigated the incremental validity of Big Five personality traits for predicting academic criteria (college GPA, course performance) while controlling for academic ability (SAT). Results showed that conscientiousness incrementally predicted each criterion over SAT. Results also showed that behavior (attendance) incrementally predicted GPA and course performance and it mediated the relationship between conscientiousness and both academic criteria. Personality measures are promising predictors of academic outcomes and they may have usefulness in admissions and student development.

3) Personality predicts academic performance: Evidence from two longitudinal university samples

AUTHORS: T. Chamorropremuzic, and A. Furnham

what extent and which personality traits predict academic performance was investigated in two longitudinal studies of two British university samples. Academic performance was assessed throughout a three years period and via multiple criteria (e.g., exams and final-year project). In addition several indicators of academic behaviour, e.g., absenteeism, essay writing, tutors' exam predictions, were also examined with regard to both academic performance and personality traits. In sample 1 (N=70), the Big Five personality factors (Costa & McCrae, 1992)—particularly Neuroticism and Conscientiousness—were found to predict overall final exam marks over and above several academic predictors, accounting for more than 10% of unique variance in overall exam marks. Results suggest that Neuroticism may impair academic performance, while Conscientiousness may lead to higher academic achievement. In sample 2 (N=75) the EPQ-R (Eysenck & Eysenck, 1985) was used as the personality measure and results showed the three superfactors were the most powerful predictor of academic performance, accounting for nearly 17% of unique variance in overall exam results. It is demonstrated that (like Neuroctisim) Psychoticism could limit academic success. The present results provide evidence supporting the inclusion of well-established personality measures in academic selection procedures, and run counter to the traditional view of ability measures as the exclusive psychometric correlate of academic performance.

4) The WHO health promoting school framework for improving the health and well-being of students and their academic achievement

AUHORS: R. Langford, C. P. Bonell, H. E. Jones, T. Pouliou, S. M. Murphy, and E. Waters

Health and education are strongly connected: healthy children achieve better results at school, which in turn are associated with improved health later in life. This relationship between health and education forms the basis of the World Health Organization's (WHO's) Health Promoting Schools (HPS) framework, an approach to promoting health in schools that addresses the whole

school environment. Although the HPS framework is used in many schools, we currently do not know if it is effective. This review aimed to assess whether the HPS framework can improve students' health and well-being and their performance at school.

Study characteristics

We searched 20 health, education, and social science databases, as well as trials registries and relevant websites, for cluster-randomised controlled trials of school-based interventions aiming to improve the health of young people aged four to 18 years. We only included trials of programmes that addressed all three points in the HPS framework: including health education in the curriculum; changing the school's social or physical environment, or both; and involving students' families or the local community, or both.

5) Learning technologies: Affective and social issues in computer-supported collaborative learning

AUTHORS: A. Jones, and K. Issroff

This paper is concerned with affective issues in learning technologies in a collaborative context. Traditionally in learning there has been a division between cognition and affect: where cognition is concerned with skills and processes such as thinking and problem-solving and affect with emotional areas such as motivation, attitudes, feelings. Affective issues have been viewed as somewhat problematic in studying learning, so although it is well known that learner attitude, motivation, and emotional state are very important, they have often been excluded from the frame of research, or studied separately from cognitive learning. This position is gradually changing and this paper considers what previous research has been conducted in these areas. It discusses the role of affective factors in three main areas of collaboration: in settings where learners are co-located, in on-line communities and to support and develop socio-emotional skills. It considers relevant developments in these areas, what the outcomes have been and suggests important directions for future research.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

Exiting system refers to four supervised learning algorithms (consisting of support vector machine (SVM), logistic regression (LR), decision tree and naïve Bayes) are used to classify students' performance.

DISADVANTAGES OF EXISTING SYSTEM:

- **Inconsistent Results Across Courses**: Using multiple data sources or features doesn't always lead to better predictions. Predictions can vary widely across different courses, even within the same institution.

- **Lack of Open-Access Large-Scale Multisource Datasets:** There is a lack of large-scale, open-access, and multisource data sets in the education field. This lack of data makes it difficult to compare the performance of different academic performance prediction algorithms effectively.

- **Simplicity of Existing Algorithms**: Most existing algorithms for predicting academic performance are basic, relying on simple statistical methods (e.g., ANOVA, Post hoc tests) or traditional machine learning techniques (e.g., SVM, LR).

.

3.2. PROPOSED SYSTEM:

The proposed system is an academic performance prediction framework developed using Python programming language and the Random Forest Classification algorithm. It operates on a dataset comprising 1044 records, each containing 31 features related to student behavior and demographic information, along with their final grades.

The system uses the Random Forest Classification algorithm, which builds multiple decision trees to analyze student attributes and predict academic performance. These attributes include demographic factors (age, gender, family background) and behavioral indicators (study habits, extracurricular activities, health status).

During data preprocessing, the system cleans and transforms the data by handling missing values, encoding categorical variables, and standardizing numerical features. Feature engineering is applied to extract key insights and improve prediction accuracy

The system is trained and tested to predict students' final grades with 98.76% accuracy, proving its effectiveness in identifying academic performance patterns.

ADVANTAGES OF PROPOSED SYSTEM:

- **High Accuracy:** The system makes reliable and accurate predictions of students' performance.
- **Comprehensive Data Use:** It incorporates a wide variety of data, including demographics and student behavior, for a more complete prediction.
- **Strong Algorithm:** The Random Forest algorithm combines multiple decision trees, ensuring better overall performance.
- **Adaptability:** The system can be applied to different data sets, courses making it flexible.
- **Valuable Insights:** The system helps identify key factors affecting student performance, aiding in better decision-making by educational institutions.

3.3. HARDWARE & SOFTWARE REQUIREMENTS:

3.3.1. SOFTWARE REQUIREMENT SPECIFICATIONS:

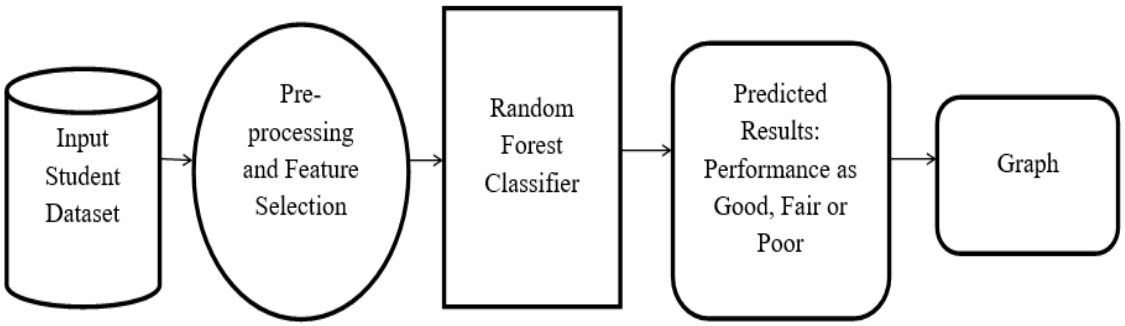
- Operating system : Windows 10.
- Coding Language : Python 3.8
- Web Framework : Flask

3.3.2. HARDWARE REQUIREMENT SPECIFICATIONS:

- System : Pentium i3 Processor.
- Hard Disk : 500 GB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- RAM : 4 GB

4. SYSTEM DESIGN

4.1 ARCHITECTURE



DATA FLOW DIAGRAM

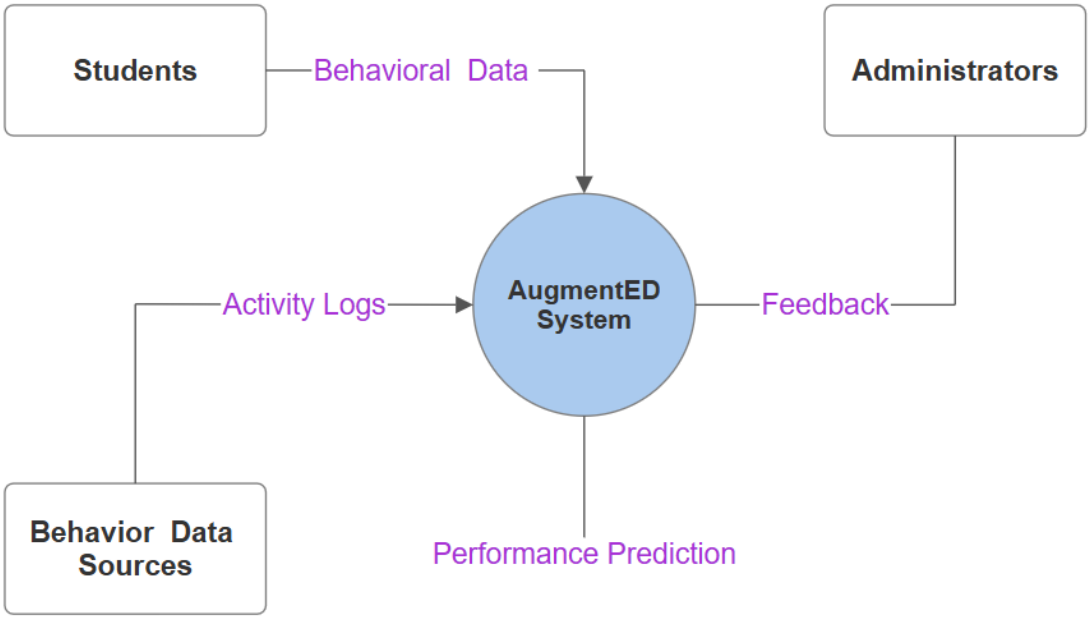
The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

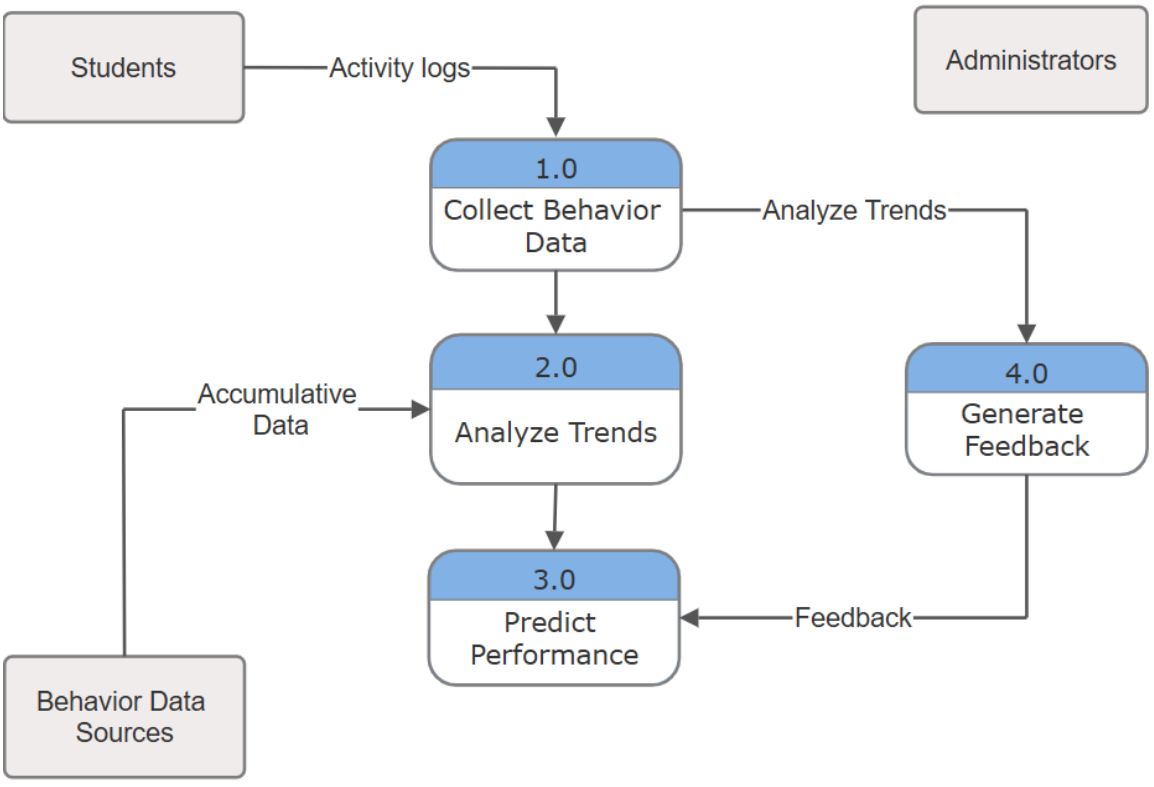
DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

DFD 0 Level



DFD 1 Level



4.2 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

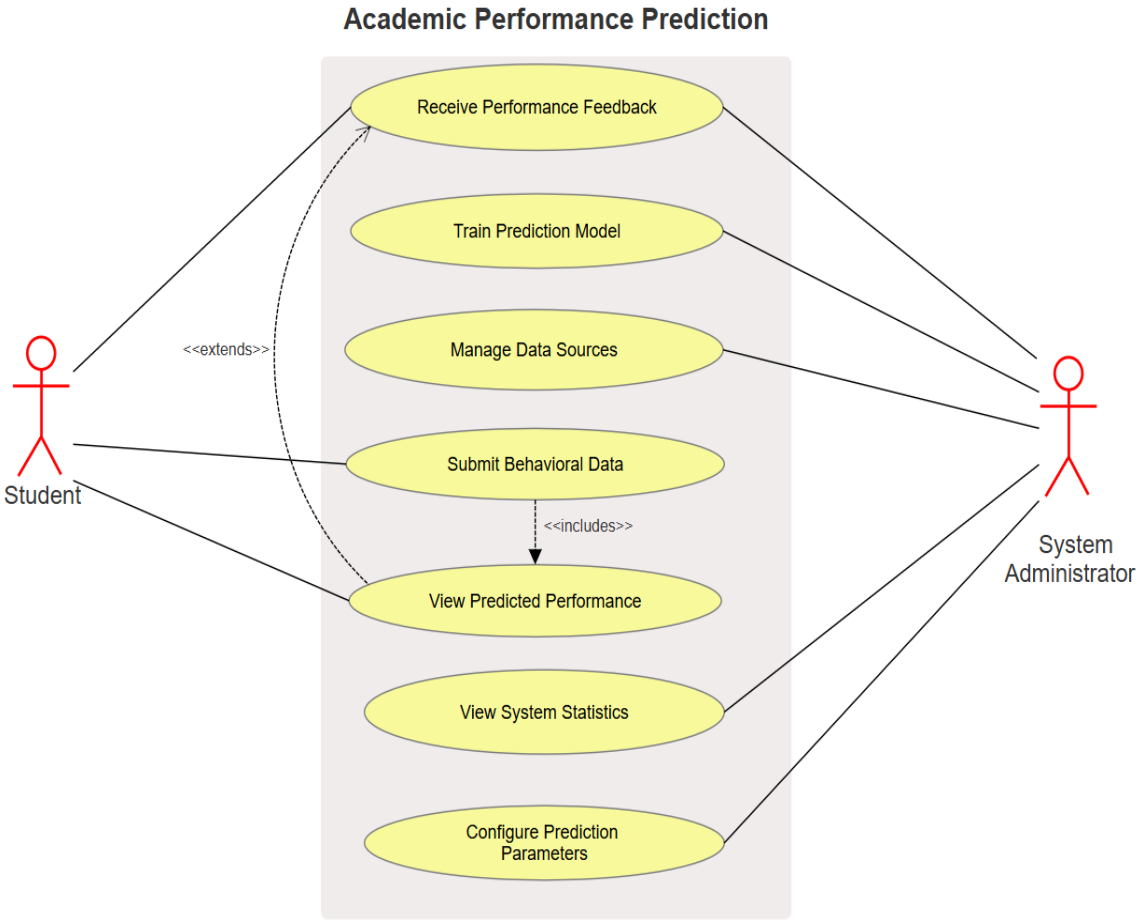
GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

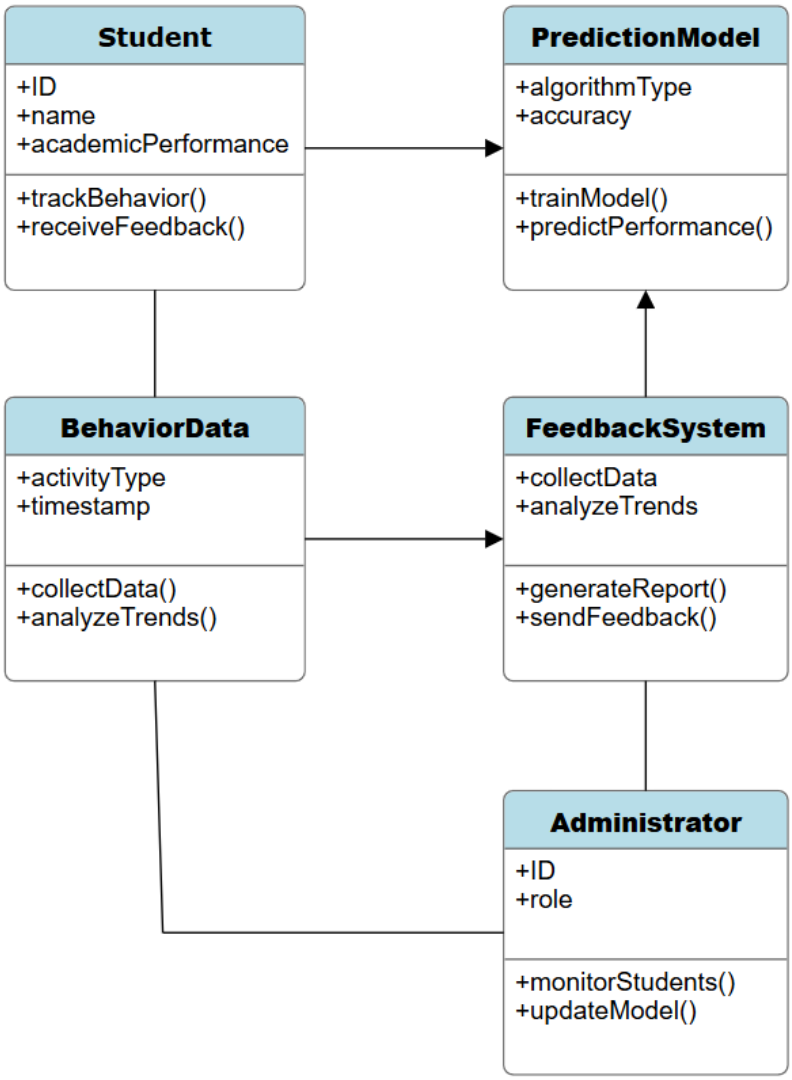
USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



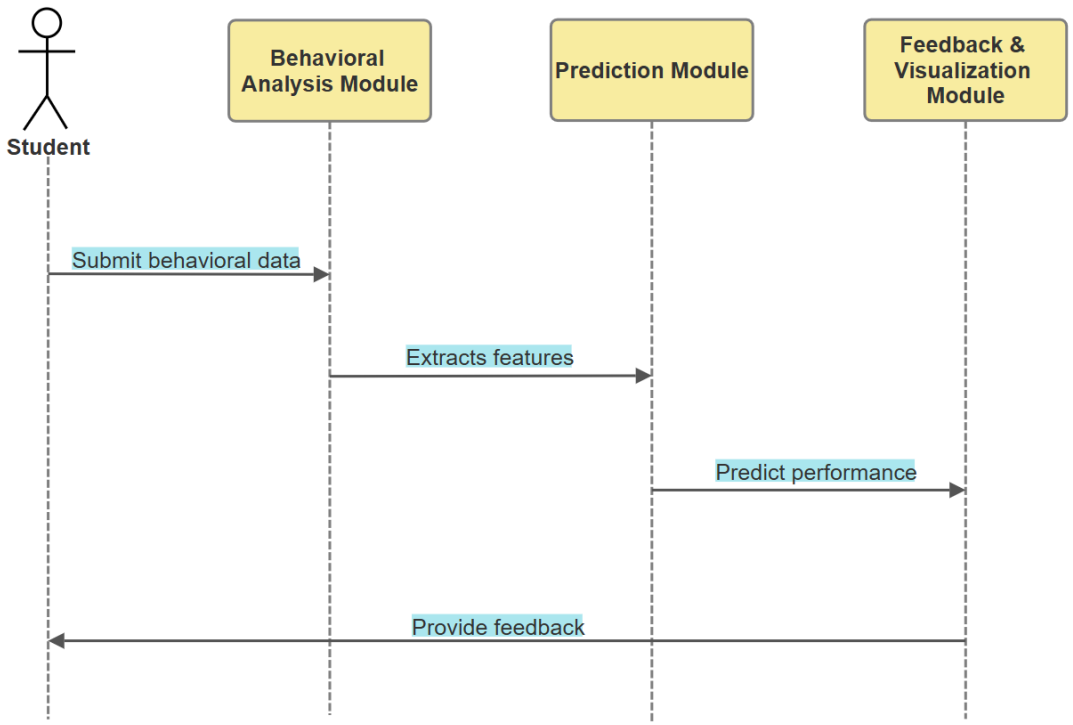
CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



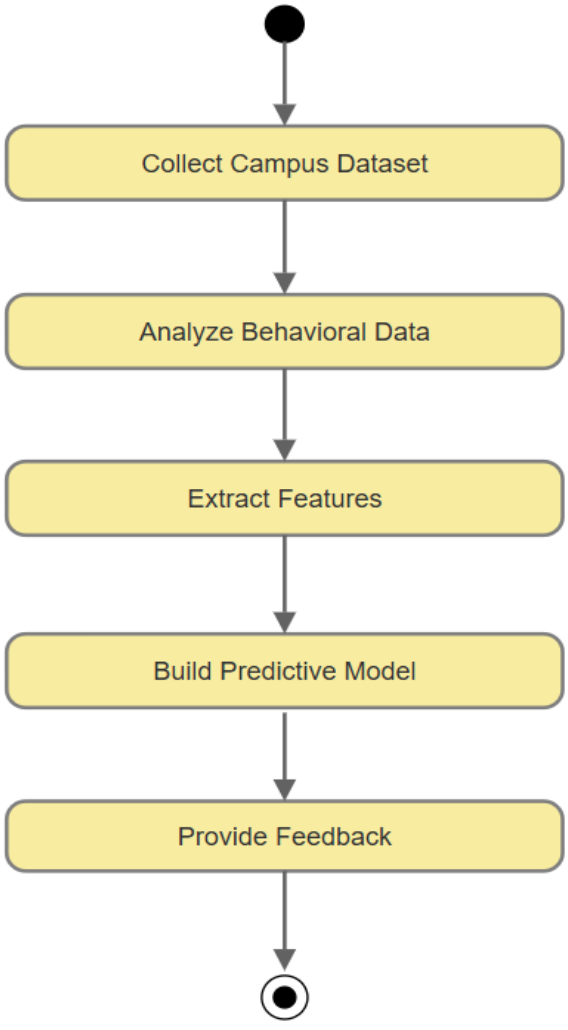
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



5. IMPLEMENTATION

MODULES:

- Data Collection
- Dataset
- Data Preparation
- Model Selection
- Analyze and Prediction
- Accuracy on test set
- Saving the Trained Model

5.1 MODULES DESCSRIPTION:

Data Collection:

In the first module of Academic Performance Prediction Based on Multisource, Multifeature Behavioral Data, we developed the system to get the input dataset. Data collection process is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform. There are several techniques to collect the data, like web scraping, manual interventions. Our dataset is placed in the project and it's located in the model folder. The dataset is referred from the popular standard dataset repository kaggle where all the researchers refer it. The dataset consists of numerical data. The following is the URL for the dataset referred from kaggle.

Kaggle Link:

https://www.kaggle.com/datasets/jayaprakashpondy/student-academic-performance

Dataset:

The dataset contains 1044 number of records.

In this data set we are taken 34 columns in the dataset, which are described below:

Id - studet's id

School - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)

Class - student's class (binary: 'mat' - Mathmatics or 'pot' - Portuguese language)

Sex - student's sex (binary: 'F' - female or 'M' - male)

Age - student's age (numeric: from 15 to 22)

Address - student's home address type (binary: 'U' - urban or 'R' - rural)

Famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than **3)**

Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)

Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')

guardian - student's guardian (nominal: 'mother', 'father' or 'other')

traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)

studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)

failures - number of past class failures (numeric: n if 1<=n<3, else 4)

schoolsup - extra educational support (binary: true or false)

famsup - family educational support (binary: true or false)

paid - extra paid classes within the course subject (Math or Portuguese) (binary: true or false)

activities - extra-curricular activities (binary: true or false)

nursery - attended nursery school (binary: true or false)

higher - wants to take higher education (binary: true or false)

internet - Internet access at home (binary: true or false)

romantic - with a romantic relationship (binary: true or false)

famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)

freetime - free time after school (numeric: from 1 - very low to 5 - very high)

goout - going out with friends (numeric: from 1 - very low to 5 - very high)
Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
health - current health status (numeric: from 1 - very bad to 5 - very good)
absences - number of school absences (numeric: from 0 to 93)
G3 - the final grade (numeric: from 0 to 20, output target)

Data Preparation:

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.). Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data. Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis. Split into training and evaluation sets.

Model Selection:

While creating a machine learning model, we need two dataset, one for training and other for testing. But now we have only one. So let's split this in two with a ratio of 80:20. We will also divide the dataframe into feature column and label column.

Here we imported train_test_split function of sklearn. Then use it to split the dataset. Also, *test_size = 0.2*, it makes the split with 80% as train dataset and 20% as test dataset.

The *random state* parameter seeds random number generator that helps to split the dataset. The function returns four datasets. Labelled them as *train_x, train_y, test_x, test_y*. If we see shape of this datasets we can see the split of dataset. We will use Random Forest Classifier, which fits multiple decision tree to the data. Finally I train the model by passing *train_x, train_y* to the *fit* method.

Once the model is trained, we need to Test the model. For that we will pass *test_x* to the predict method. Random Forest is one of the most powerful methods that is used in machine learning for classification problems. The random forest comes in the category of the supervised classification algorithm. This algorithm is carried out in two different stages the first one deals

with the creation of the forest of the given dataset, and the other one deals with the prediction from the classification.

Analyze and Prediction:

In the actual dataset, we chose only 20 features:

Sex - student's sex (binary: 'F' - female or 'M' - male)

Age - student's age (numeric: from 15 to 22)

Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')

guardian - student's guardian (nominal: 'mother', 'father' or 'other')

studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)

failures - number of past class failures (numeric: n if 1<=n<3, else 4)

schoolsup - extra educational support (binary: true or false)

famsup - family educational support (binary: true or false)

paid - extra paid classes within the course subject (Math or Portuguese) (binary: true or false)

activities - extra-curricular activities (binary: true or false)

internet - Internet access at home (binary: true or false)

freetime - free time after school (numeric: from 1 - very low to 5 - very high)

goout - going out with friends (numeric: from 1 - very low to 5 - very high)

health - current health status (numeric: from 1 - very bad to 5 - very good)

absences - number of school absences (numeric: from 0 to 93)

G3 - the final grade (numeric: from 0 to 20, output target)(good,bad,fair)

Accuracy on test set:

After training and evaluating the model on the validation set, the accuracy of the model will be assessed on the test set. The accuracy on the test set will be an important metric for evaluating the model's performance. We got an accuracy of 98.76% on test set.

Saving the Trained Model:

Once you're confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.
Make sure you have pickle installed in your environment.
Next, let's import the module and dump the model into .pkl file.

5.2 SOFTWARE ENVIRONMENT

Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

Easy-to-learn −Easy-to-read: Python code is more clearly defined and visible to the eyes.

Easy-to-maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building

large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.
- Python is available on a wide variety of platforms including Linux and Mac OS X.

Let's understand how to set up our Python environment.

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/
- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.
- The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt:

```
$ python

Python2.4.3(#1,Nov112010,13:34:43)

[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2

Type"help","copyright","credits"or"license"for more information.

>>>
```

Type the following text at the Python prompt and press the Enter:

```
>>>print"Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result:

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file:

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows:

```
$ python test.py
```

This produces the following result:

```
Hello, Python!
```

Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods:

S.No	Methods & Description
1	**GET:** Sends data in unencrypted form to the server. Most common method.
2	**HEAD** Same as GET, but without response body
3	**POST** Used to send HTML form data to server. Data received by POST method is not cached by server.
4	**PUT** Replaces all current representations of the target resource with the uploaded content.

5	**DELETE** Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

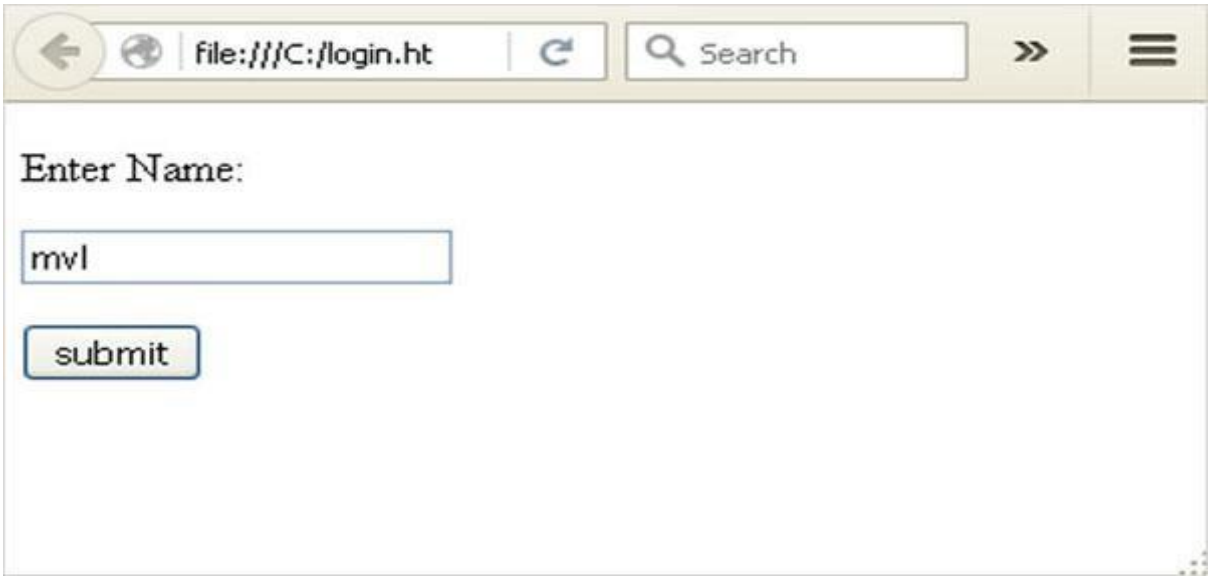
```
<html>

<body>

<formaction="http://localhost:5000/login"method="post">

<p>Enter Name:</p>

<p><inputtype="text"name="nm"/></p>

<p><inputtype="submit"value="submit"/></p>

</form>

</body>

</html>
```

Now enter the following script in Python shell.

```
from flask importFlask, redirect,url_for, request

app=Flask(__name__)

@app.route('/success/<name>')

def success(name):

return'welcome %s'% name
```

```
@app.route('/login',methods=['POST','GET'])

def login():

ifrequest.method=='POST':

user=request.form['nm']

return redirect(url_for('success',name= user))

else:

user=request.args.get('nm')

return redirect(url_for('success',name= user))

if __name__ =='__main__':

app.run(debug =True)
```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.



Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained by:

```
user = request.form['nm']
```

It is passed to **'/success'** URL as variable part. The browser displays a **welcome** message in the window.

Change the method parameter to **'GET'** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of 'nm' parameter is now obtained by:

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to 'nm' parameter is passed on to '/success' URL as before.

What is Python?

Python is a popular programming language. It was created in 1991 by Guido van Rossum. It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Python Install

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Python Quick start

Python is an interpreted programming language, this means that as a developer you write Python (.py) files in a text editor and then put those files into the python interpreter to be executed.

The way to run a python file is like this on the command line:

```
C:\Users\Your Name>python helloworld.py
```

Where "helloworld.py" is the name of your python file.

Let's write our first Python file, called helloworld.py, which can be done in any text editor.

helloworld.py

```
Print ("Hello, World!")
```

Simple as that. Save your file. Open your command line, navigate to the directory where you saved your file, and run:

```
C:\Users\Your Name>python helloworld.py
```

The output should read:

```
Hello, World!
```

Congratulations, you have written and executed your first Python program.

The Python Command Line

To test a short amount of code in python sometimes it is quickest and easiest not to write the code in a file. This is made possible because Python can be run as a command line itself.

Type the following on the Windows, Mac or Linux command line:

```
C:\Users\Your Name>python
```

From there you can write any python, including our hello world example from earlier in the tutorial:

```
C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32

bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> print("Hello, World!")
```

Which will write "Hello, World!" in the command line:

```
C:\Users\Your Name>python

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32

bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> print("Hello, World!") Hello, World!
```

Whenever you are done in the python command line, you can simply type the following to quit the python command line interface:

```
exit()
```

Execute Python Syntax

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
```

Hello, World!

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

C:\Users*Your Name*>python myfile.py

Python Indentations

Where in other programming languages the indentation in code is for readability only, in Python the indentation is very important. Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Comments

Python has commenting capability for the purpose of in-code documentation.

Comments start with a #, and Python will render the rest of the line as a comment:

Comments in Python:

```
# This is a comment

print("Hello, World!")
```

Docstrings

Python also has extended documentation capability, called docstrings.

Docstrings can be one line, or multiline.

Python uses triple quotes at the beginning and end of the docstring:

6. ALGORITHMS

Algorithm:

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision-making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2… On. Each object in S has one outcome for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi, we build a subsidiary decision tree by invoking the same procedure recursively on the set Si.

Gradient Boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training

time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by Tin Kam Ho[1] using the random subspace method, which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by Leo Breiman and Adele Cutler, who registered "Random Forests" as a trademark in 2006 (as of 2019, owned by Minitab, Inc.).The extension combines Breiman's "bagging" idea and random selection of features, introduced first by Ho[1] and later independently by Amit and Geman[13] in order to construct a collection of decision trees with controlled variance.

Random forests are frequently used as "Blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

SVM

In classification tasks a discriminant machine learning technique aims at finding, based on an independent and identically distributed (iid) instances. Unlike generative machine learning approaches, which require computations of training dataset, a discriminant function that can correctly predict labels for newly acquired conditional probability distributions, a discriminant classification function takes a data point *x* and assigns it to one of the different classes that are a part of the classification task.

Less powerful than generative approaches, which are mostly used when prediction involves outlier detection, discriminant approaches require fewer computational resources and less training data, especially for a multidimensional feature space and when only posterior probabilities are needed. From a geometric perspective, learning a classifier is equivalent to finding the equation for a multidimensional surface that best separates the different classes in the feature space.

SVM is a discriminant technique, and, because it solves the convex optimization problem analytically, it always returns the same optimal hyperplane parameter—in contrast to genetic algorithms (GAs) or perceptrons, both of which are widely used for classification in machine learning. For perceptron, solutions are highly dependent on the initialization and termination criteria.

For a specific kernel that transforms the data from the input space to the feature space, training returns uniquely defined SVM model parameters for a given training set, whereas the perceptron and GA classifier models are different each time training is initialized. The aim of GAs and perceptron is only to minimize error during training, which will translate into several hyperplanes' meeting this requirement.

7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub- assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

7.1 TYPES OF TESTS:

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- **Valid Input** : identified classes of valid input must be accepted.
- **Invalid Input** : identified classes of invalid input must be rejected.
- **Functions** : identified functions must be exercised.
- **Output** : identified classes of application outputs must be exercised.
- **Systems/Procedures :** interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

7.2 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8. RESULT

The project "Academic Performance Prediction Based on Multisource, Multifeature Behavioral Data" aims to predict students' academic performance by analyzing data from various sources on university campuses. Here's a summary of the key aspects and potential results:

Core Idea:

The project leverages the vast amounts of digital data generated by students daily across different campus systems (online learning platforms, library usage, attendance records, extracurricular activities, etc.).

It aims to move beyond traditional prediction methods that rely solely on academic grades or demographic information.

By analysing behavioral patterns and extracting relevant features from this multisource data, the project seeks to build more accurate and holistic predictive models.

- **High Prediction Accuracy:** Several sources suggest that the proposed models, such as the Augmented Education (AugmentED) model and the Multi-Source Deep Learning Model (MSDLM), can achieve high accuracy in predicting student academic performance (e.g., one study reports 91.1% accuracy for MSDLM).

- **Early Identification of At-Risk Students:** The models can potentially identify students who are likely to underperform early in their academic careers, allowing for timely interventions and support.

- **Understanding Key Behavioral Predictors:** The analysis of features can reveal which behavioral patterns are most strongly correlated with academic success or struggle.

- **Development of Intelligent Systems for Education:** The project contributes to the growing field of applying data science and machine learning to enhance educational outcomes.

9. CONCLUSION AND FUTURE WORK

In conclusion, this project has successfully developed an academic performance prediction system based on multisource, multifeature behavioral data analysis. Leveraging Python programming language and the Random Forest Classification algorithm, the system demonstrates remarkable accuracy in forecasting students' final grades, achieving a commendable accuracy of 98.76% on the test set.

Through the utilization of a comprehensive dataset comprising 1044 records and 31 features encompassing various aspects of student behavior and demographic information, the system provides valuable insights into the factors influencing academic outcomes. By incorporating demographic factors, familial background, study habits, extracurricular activities, and health status, the system offers a holistic perspective on student performance determinants.

The robustness of the Random Forest Classification algorithm ensures the system's ability to handle complex datasets, capture nonlinear relationships, and generalize well to unseen data. Rigorous preprocessing techniques enhance the quality of input data, contributing to the overall effectiveness of the predictive model.

The system's high accuracy, coupled with its ability to provide actionable insights, makes it a valuable tool for educational institutions seeking to improve student outcomes through data-driven decision-making. By offering a nuanced understanding of academic performance determinants, the system paves the way for targeted interventions, personalized support strategies, and resource allocation optimization.

Overall, this project underscores the potential of machine learning techniques in analyzing educational data and providing valuable insights for enhancing student success. It represents a significant step towards leveraging data-driven approaches to address challenges in education and contribute to the ongoing efforts in improving educational outcomes.

FUTURE WORK:

Integration of Additional Data Sources: Incorporating additional data sources such as socioeconomic factors, teacher characteristics, and school environment variables can enhance the predictive capabilities of the model. By integrating a broader range of data, future iterations of the project can provide a more comprehensive understanding of the factors influencing academic performance.

Exploration of Advanced Machine Learning Techniques: While Random Forest Classification has demonstrated strong performance in this project, exploring other advanced machine learning techniques such as neural networks, gradient boosting, or ensemble methods could further improve prediction accuracy. Comparative studies can be conducted to assess the effectiveness of different algorithms in academic performance prediction.

Fine-Tuning Model Parameters: Conducting thorough parameter tuning for the Random Forest algorithm and other machine learning models can optimize their performance. Techniques such as grid search or Bayesian optimization can be employed to identify the optimal hyperparameters, thereby improving the model's predictive accuracy and generalizability.

Longitudinal Analysis: Extending the analysis to include longitudinal data spanning multiple academic years can provide insights into students' academic trajectories and long-term performance trends. By tracking students' progress over time, the predictive model can offer valuable insights into factors influencing academic persistence and graduation rates.

Development of Intervention Strategies: Utilizing the predictive model to develop targeted intervention strategies and support programs can help improve student outcomes. By identifying at-risk students early and implementing timely interventions, educational institutions can mitigate academic challenges and enhance student success.

Deployment in Real-World Educational Settings: Testing the predictive model in real-world educational settings and assessing its impact on student outcomes is essential for practical implementation. Collaborating with educational institutions to deploy the model in live environments can provide valuable feedback for refinement and optimization.

Ethical and Privacy Considerations: Ensuring compliance with ethical guidelines and privacy regulations is paramount when deploying predictive models in educational settings. Future work should focus on addressing ethical concerns related to data privacy, bias, and fairness, while maintaining transparency and accountability in model development and deployment.

10. APPENDIX

10.1 APPENDIX

The appendix provides supplementary details that support the main content of the project report titled "Academic Performance Prediction Based on Multisource, Multifeature Behavioral Data." This section enhances the reader's understanding of the implementation and supports the reproducibility of the study.

10.2 SAMPLE CODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#%matplotlib inline
import seaborn as sns
sns.set_style('whitegrid')
mat = pd.read_csv("student-mat.csv", sep=';')
por = pd.read_csv("student-por.csv", sep=';')
df = pd.concat([mat,por])
df.columns = ['school','sex','age','address','family_size','parents_status','mother_education',
'father_education','mother_job','father_job','reason','guardian','commute_time','
study_time','failures','school_support','family_support','paid_classes','activities'
,'nursery','desire_higher_edu','internet','romantic','family_quality','free_time',
'go_out','weekday_alcohol_usage','weekend_alcohol_usage','health','absences','
period1_score','period2_score','final_score']
df['final_grade'] = 'na'
```

```
df.loc[(df.final_score >= 15) & (df.final_score <= 20), 'final_grade'] = 'good'

df.loc[(df.final_score >= 10) & (df.final_score <= 14), 'final_grade'] = 'fair'

df.loc[(df.final_score >= 0) & (df.final_score <= 9), 'final_grade'] = 'poor'

df.head(5)
```

```
school sex age addressfamily_size parents_status mother_education father_education
mother_job father_job ... free_time go_out weekday_alcohol_usage
weekend_alcohol_usage health absences period1_score period2_score final_score
final_grade
```

```
df['sex'].unique()

array([0, 1])

df['mother_job'].unique()

array([0, 1, 2, 3, 4])

df['reason'].unique()

array([0, 2, 1, 3])

df['father_job'].unique()

array([4, 2, 3, 1, 0])

df['guardian'].unique()

array([1, 0, 2])

df['school_support'].unique()

array([1, 0])

df['activities'].unique()

array([0, 1])

df.drop(columns=['final_grade'], inplace=True)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =train_test_split(df, y, test_size=0.3)
```

```
print(X_train.shape)
print(X_test.shape)
(730, 20)
(314, 20)
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
rf = RandomForestClassifier(n_estimators=10,random_state=33)
rf = rf.fit(X_train, y_train)
rf.score(X_train, y_train)
1.0
from sklearn.metrics import accuracy_score
y_pred = rf.predict(X_test)
accuracy_score(y_pred,y_test)
0.9840764331210191
from xgboost import XGBClassifier
model = XGBClassifier()
model = model.fit(X_train, y_train)
[12:25:40] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
C:\Users\jpinf\Anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0,
```

```
i.e. 0, 1, 2, ..., [num_class - 1].
warnings.warn(label_encoder_deprecation_msg, UserWarning)
model.score(X_train, y_train)
1.0
from sklearn.metrics import accuracy_score
y_pred = model.predict(X_test)
accuracy_score(y_pred,y_test)
1.0
import pickle
pickle.dump(model,open('model.pkl','wb'))
model = pickle.load(open('model.pkl', 'rb'))
print(model)
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=4, num_parallel_tree=1,
              objective='multi:softprob', random_state=0, reg_alpha=0,
              reg_lambda=1, scale_pos_weight=None, subsample=1,
              tree_method='exact', use_label_encoder=True,
              validate_parameters=1, verbosity=None)
import pickle
pickle.dump(rf,open('rf.pkl','wb'))
rf = pickle.load(open('rf.pkl', 'rb'))
```

```
print(rf)
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=33, verbose=0,
                       warm_start=False)
```

10.2 SCREEN SHOTS OF MODULES

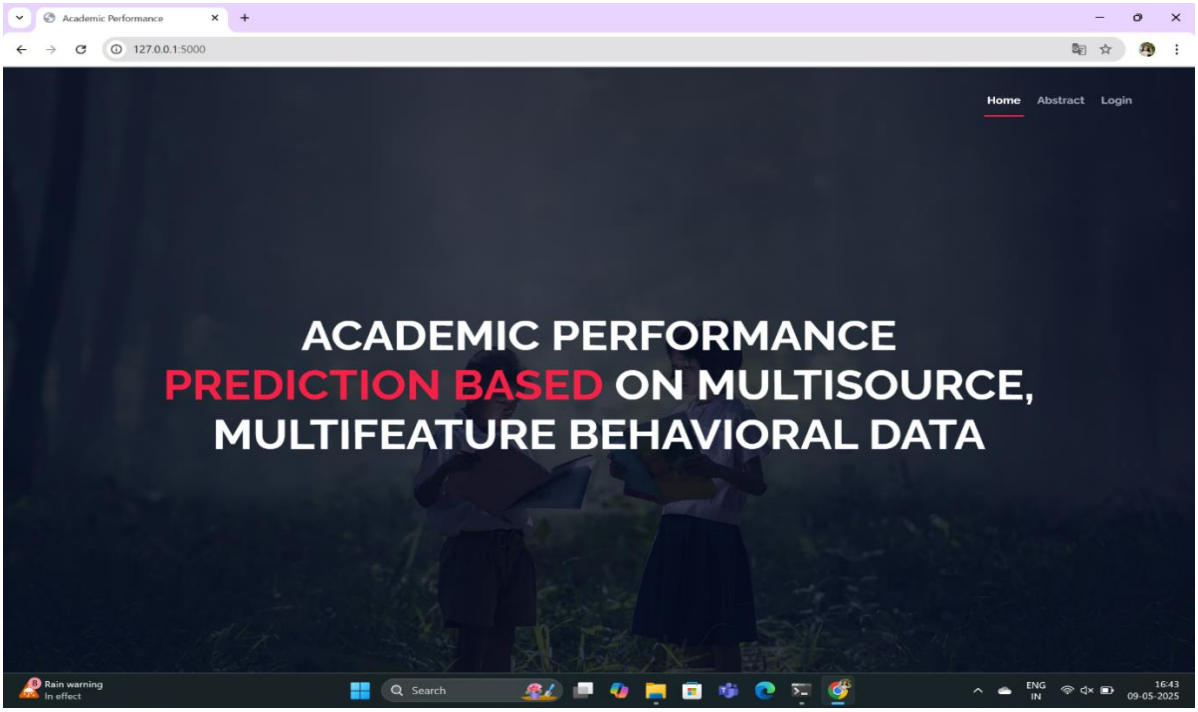


FIG 10.2.1: HOME PAGE

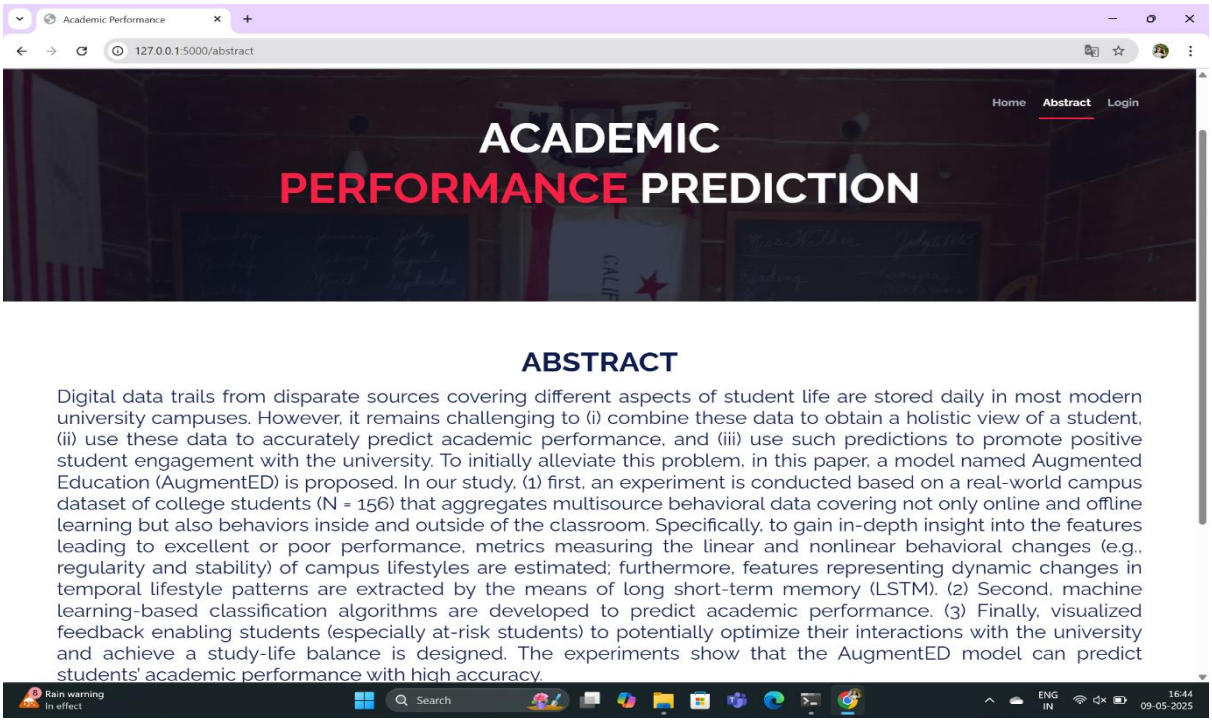


FIG 10.2.2: ABSTRACT PAGE

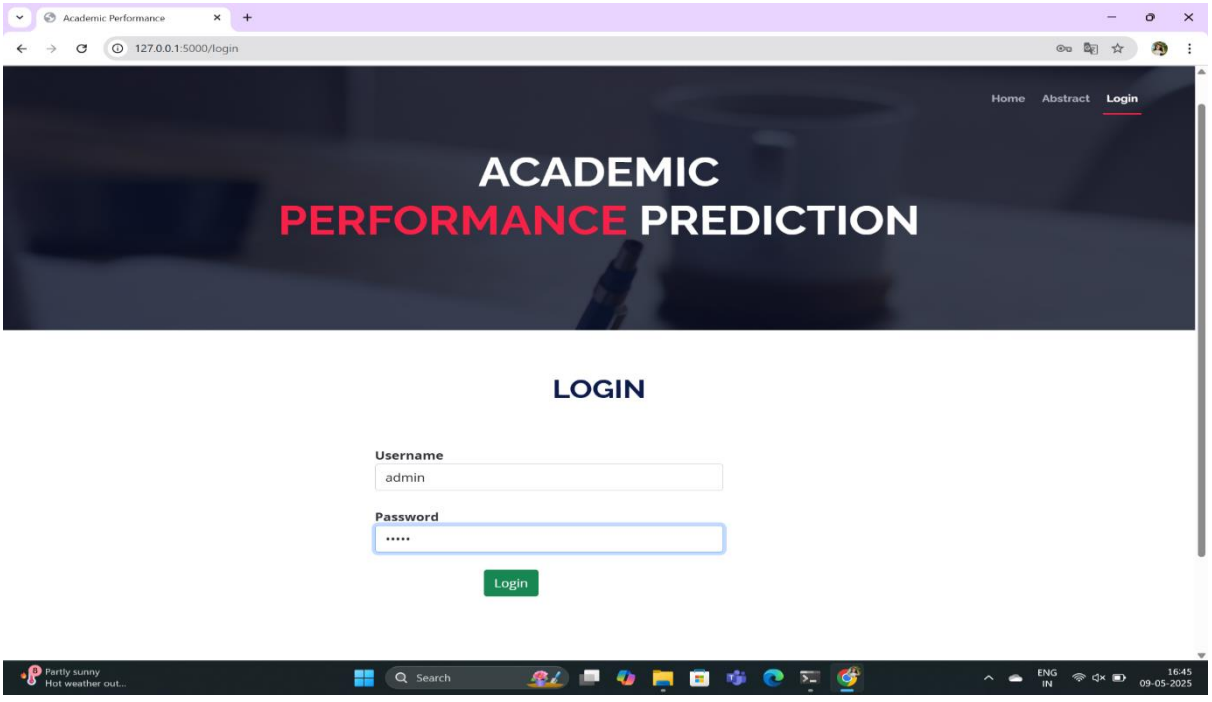


FIG 10.2.3: LOGIN PAGE

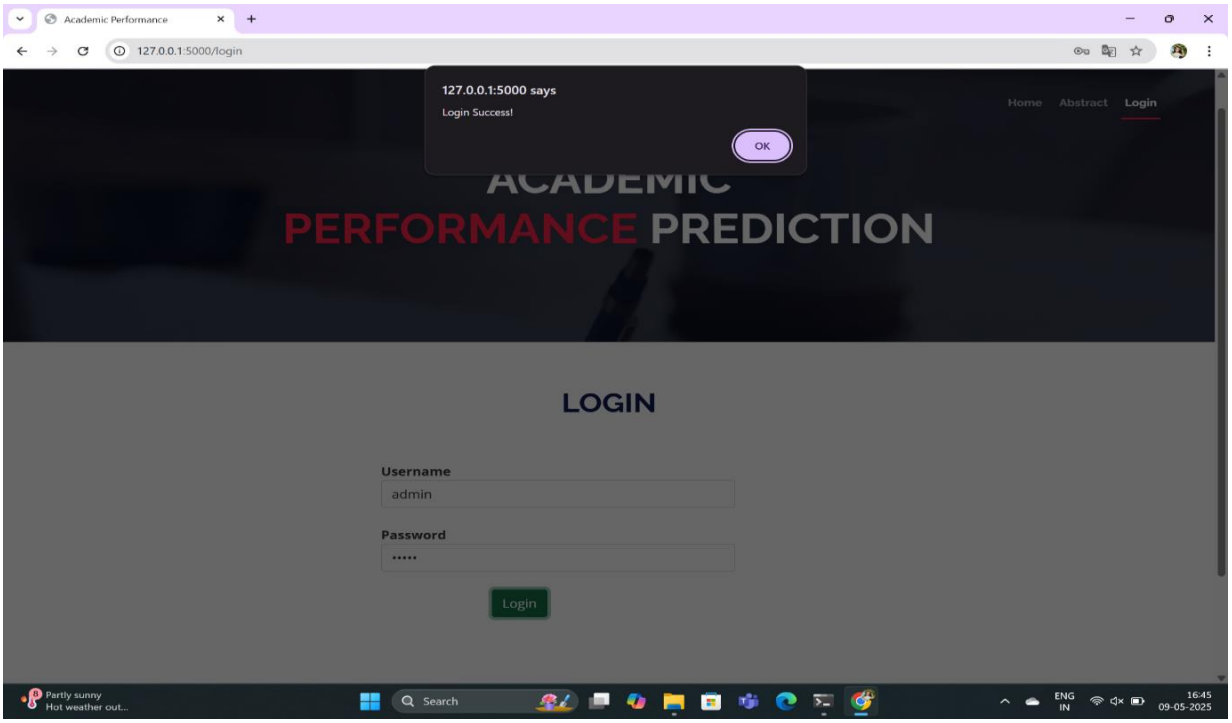


FIG 10.2.4 : LOGIN SUCCESS

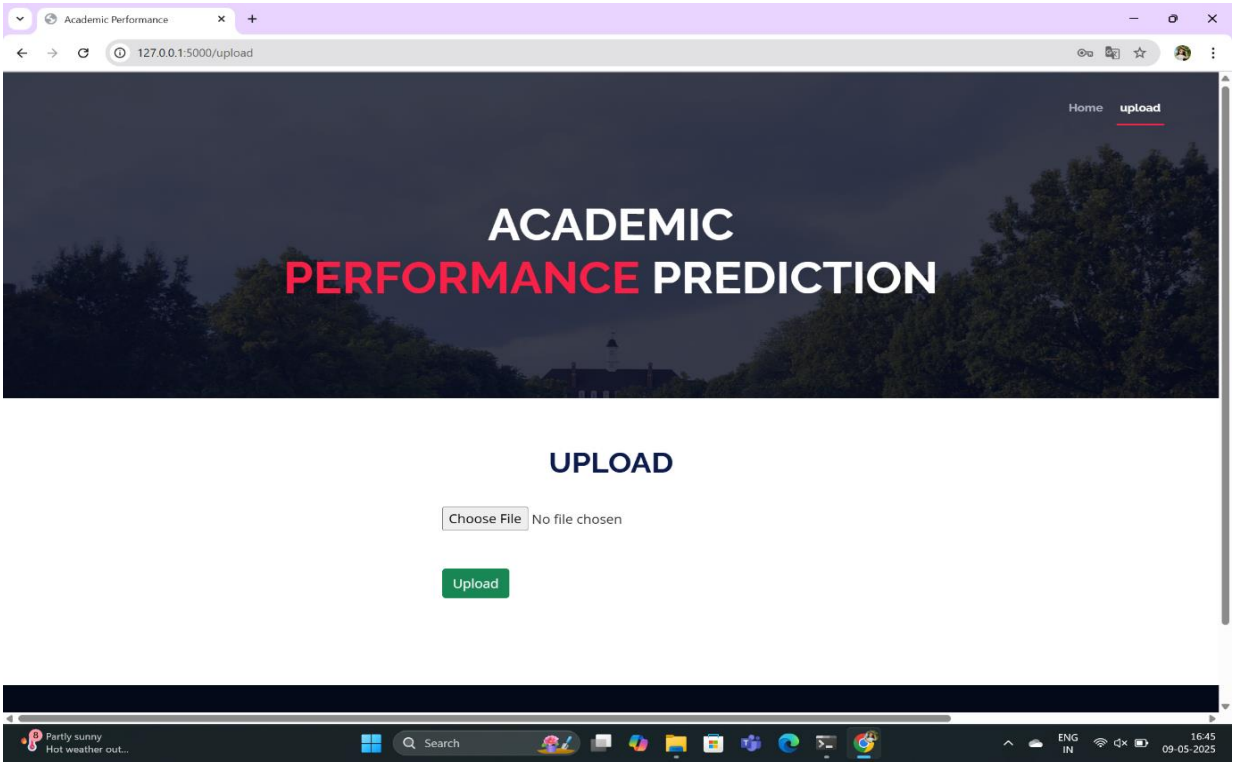
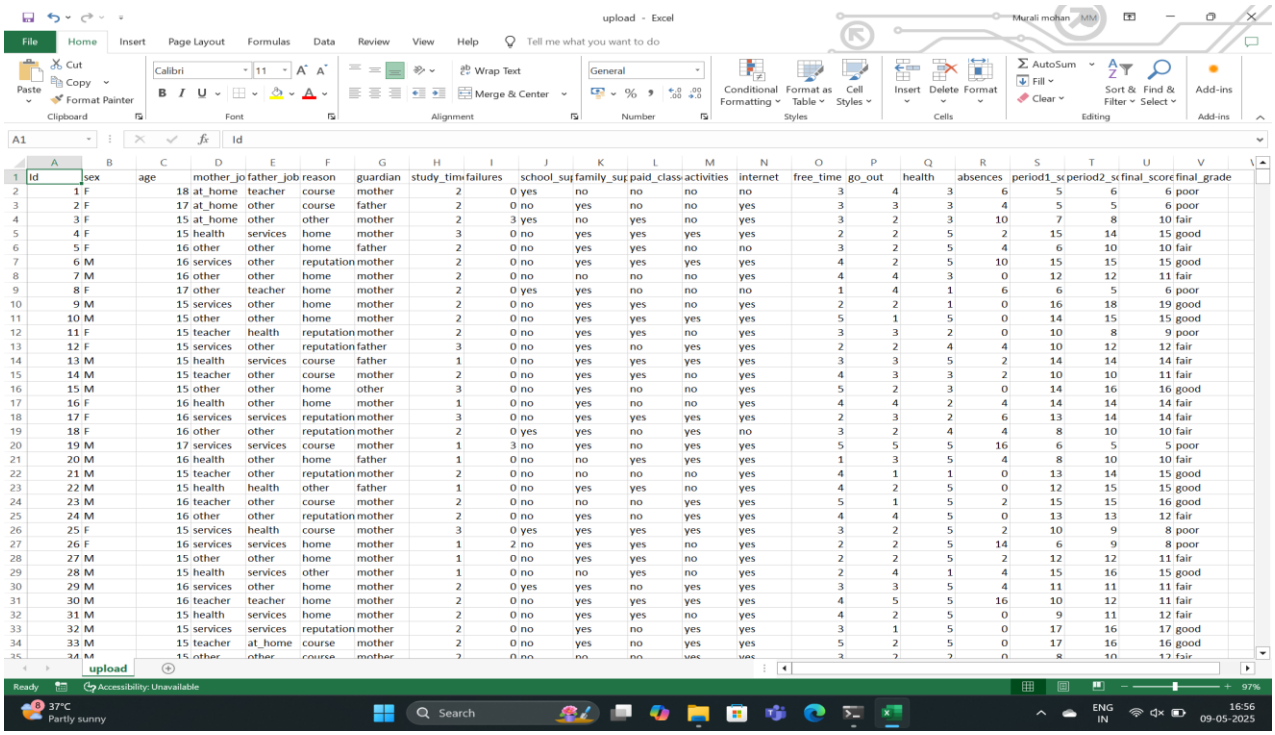
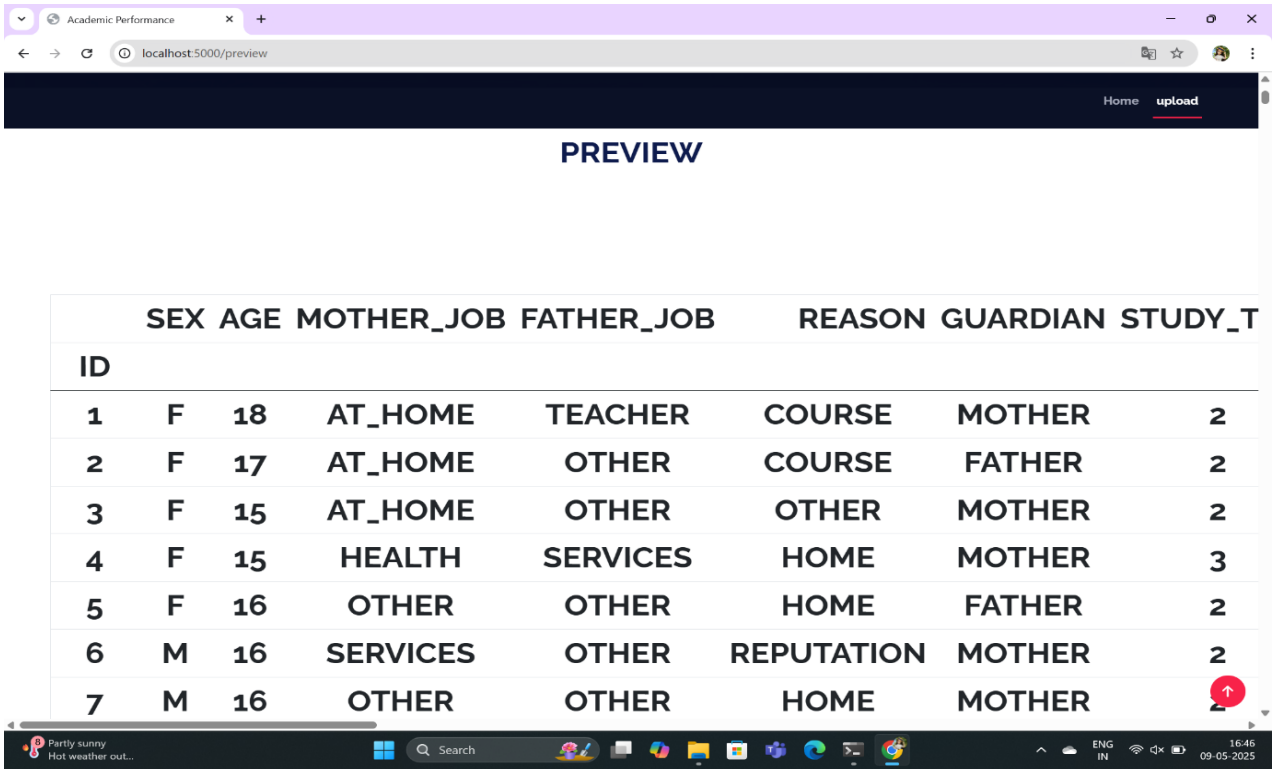


FIG 10.2.5: DATASET UPLOAD



Id	sex	age	mother_jo	father_job	reason	guardian	study_time	failures	school_su	family_su	paid_class	activities	internet	free_time	go_out	health	absences	period1_sc	period2_sc	final_score	final_grade
1	F	18	at_home	teacher	course	mother	2	0	yes	no	no	no	no	3	4	3	6	5	6	6	poor
2	F	17	at_home	other	course	father	2	0	no	yes	no	no	yes	3	3	3	4	5	5	6	poor
3	F	15	at_home	other	other	mother	2	3	yes	no	yes	no	yes	3	2	3	10	7	8	10	fair
4	F	15	health	services	home	mother	3	0	no	yes	yes	yes	yes	2	2	5	2	15	14	15	good
5	F	16	other	other	home	father	2	0	no	yes	yes	no	no	3	2	5	4	6	10	10	fair
6	M	16	services	other	reputation	mother	2	0	no	yes	yes	yes	yes	4	2	5	10	15	15	15	good
7	M	16	other	other	home	mother	2	0	no	no	no	no	yes	4	4	3	0	12	12	11	fair
8	F	17	other	teacher	home	mother	2	0	yes	yes	no	no	no	1	4	1	6	6	5	6	poor
9	M	15	services	other	home	mother	2	0	no	yes	yes	no	yes	2	2	1	0	16	18	19	good
10	M	15	other	other	home	mother	2	0	no	yes	yes	yes	yes	5	1	5	0	14	15	15	good
11	F	15	teacher	health	reputation	mother	2	0	no	yes	yes	no	yes	3	3	2	0	10	8	9	poor
12	F	15	services	other	reputation	father	3	0	no	yes	no	yes	yes	2	2	4	4	10	12	12	fair
13	M	15	health	services	course	father	1	0	no	yes	yes	yes	yes	3	3	5	2	14	14	14	fair
14	M	15	teacher	other	course	mother	2	0	no	yes	yes	no	yes	4	3	3	2	10	10	11	fair
15	M	15	other	other	home	other	3	0	no	yes	no	no	yes	5	2	3	0	14	16	16	good
16	F	16	health	other	home	mother	1	0	no	yes	no	no	yes	4	4	2	4	14	14	14	fair
17	F	16	services	services	reputation	mother	3	0	no	yes	yes	yes	yes	2	3	2	6	13	14	14	fair
18	F	16	other	other	reputation	mother	2	0	yes	yes	no	yes	no	3	2	4	4	8	10	10	fair
19	M	17	services	services	course	mother	1	3	no	yes	no	yes	yes	5	5	5	16	6	5	5	poor
20	M	16	health	other	home	father	1	0	no	no	yes	yes	yes	1	3	5	4	8	10	10	fair
21	M	15	teacher	other	reputation	mother	2	0	no	no	no	no	yes	4	1	1	0	13	14	15	good
22	M	15	health	health	other	father	1	0	no	yes	yes	no	yes	4	2	5	0	12	15	15	good
23	M	16	teacher	other	course	mother	2	0	no	no	no	no	yes	5	1	5	2	15	15	16	good
24	M	16	other	other	reputation	mother	2	0	no	yes	no	yes	yes	4	4	5	0	13	13	12	fair
25	F	15	services	health	course	mother	3	0	yes	yes	yes	yes	yes	3	2	5	2	10	9	8	poor
26	F	16	services	services	home	mother	1	2	no	yes	yes	no	yes	2	2	5	14	6	9	8	poor
27	M	15	other	other	home	mother	1	0	no	yes	yes	no	yes	2	2	5	2	12	12	11	fair
28	M	15	health	services	other	mother	1	0	no	no	yes	no	yes	2	4	1	4	15	16	15	good
29	M	16	services	other	home	mother	2	0	yes	yes	no	yes	yes	3	3	5	4	11	11	11	fair
30	M	16	teacher	teacher	home	mother	2	0	no	yes	yes	yes	yes	4	5	5	16	10	12	11	fair
31	M	15	health	services	home	mother	2	0	no	yes	yes	no	yes	4	2	5	0	9	11	12	fair
32	M	15	services	services	reputation	mother	2	0	no	yes	no	yes	yes	3	1	5	0	17	16	17	good
33	M	15	teacher	at_home	course	mother	2	0	no	yes	no	yes	yes	5	2	5	0	17	16	16	good

FIG 10.2.6: CSV FILE



PREVIEW

ID	SEX	AGE	MOTHER_JOB	FATHER_JOB	REASON	GUARDIAN	STUDY_T
1	F	18	AT_HOME	TEACHER	COURSE	MOTHER	2
2	F	17	AT_HOME	OTHER	COURSE	FATHER	2
3	F	15	AT_HOME	OTHER	OTHER	MOTHER	2
4	F	15	HEALTH	SERVICES	HOME	MOTHER	3
5	F	16	OTHER	OTHER	HOME	FATHER	2
6	M	16	SERVICES	OTHER	REPUTATION	MOTHER	2
7	M	16	OTHER	OTHER	HOME	MOTHER	2

FIG 10.2.7: VIEW DATASET

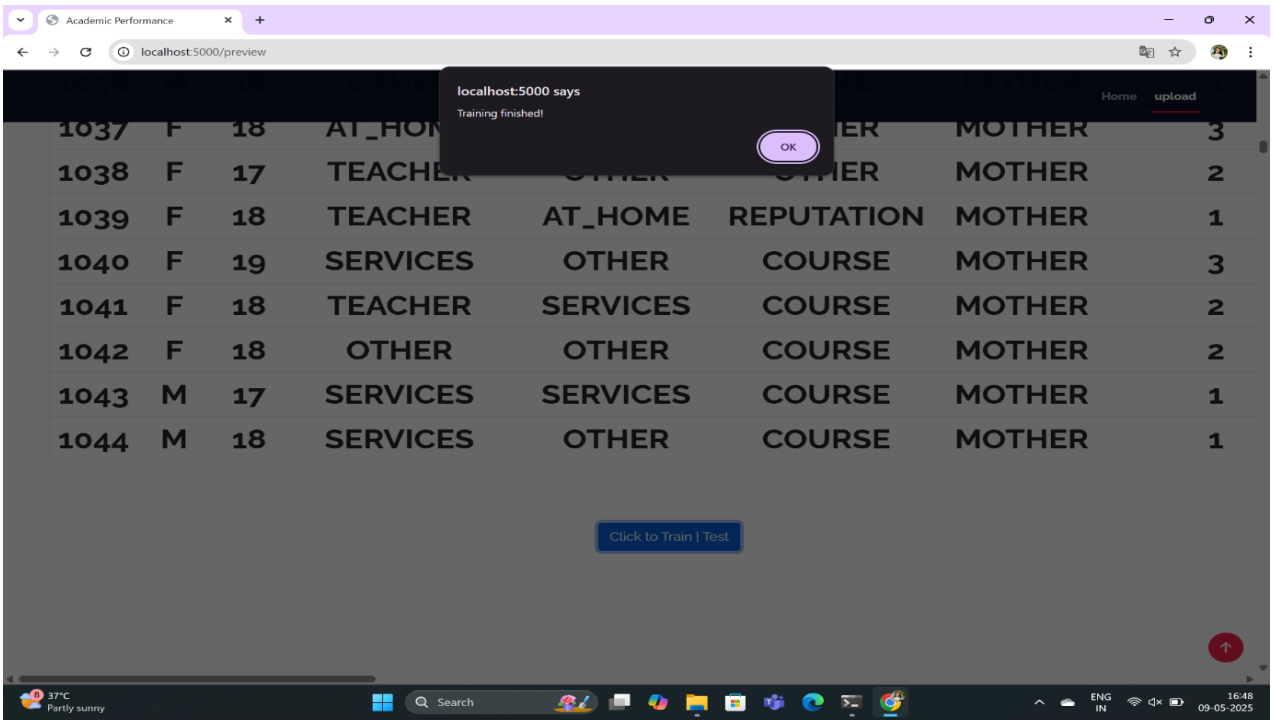


FIG 10.2.8: TRAIN DATA

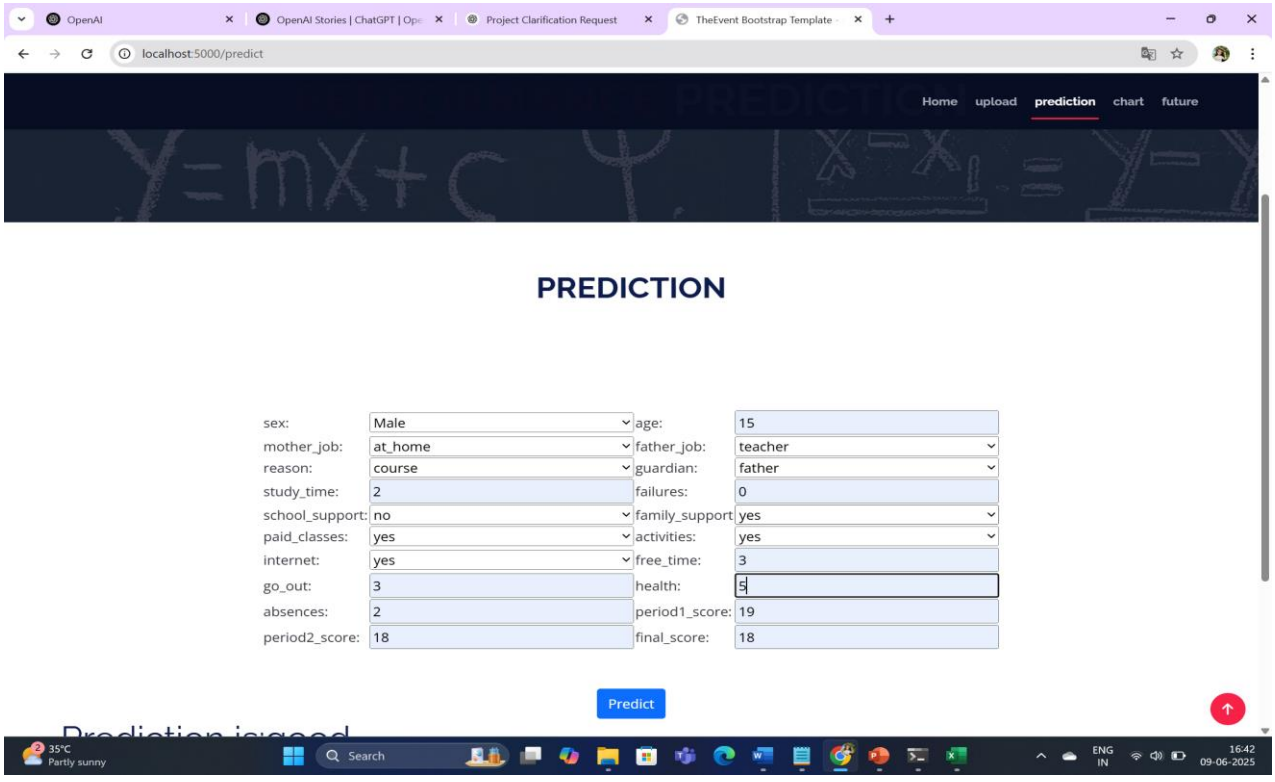


FIG 10.2.9: ENTER DETAILS 1

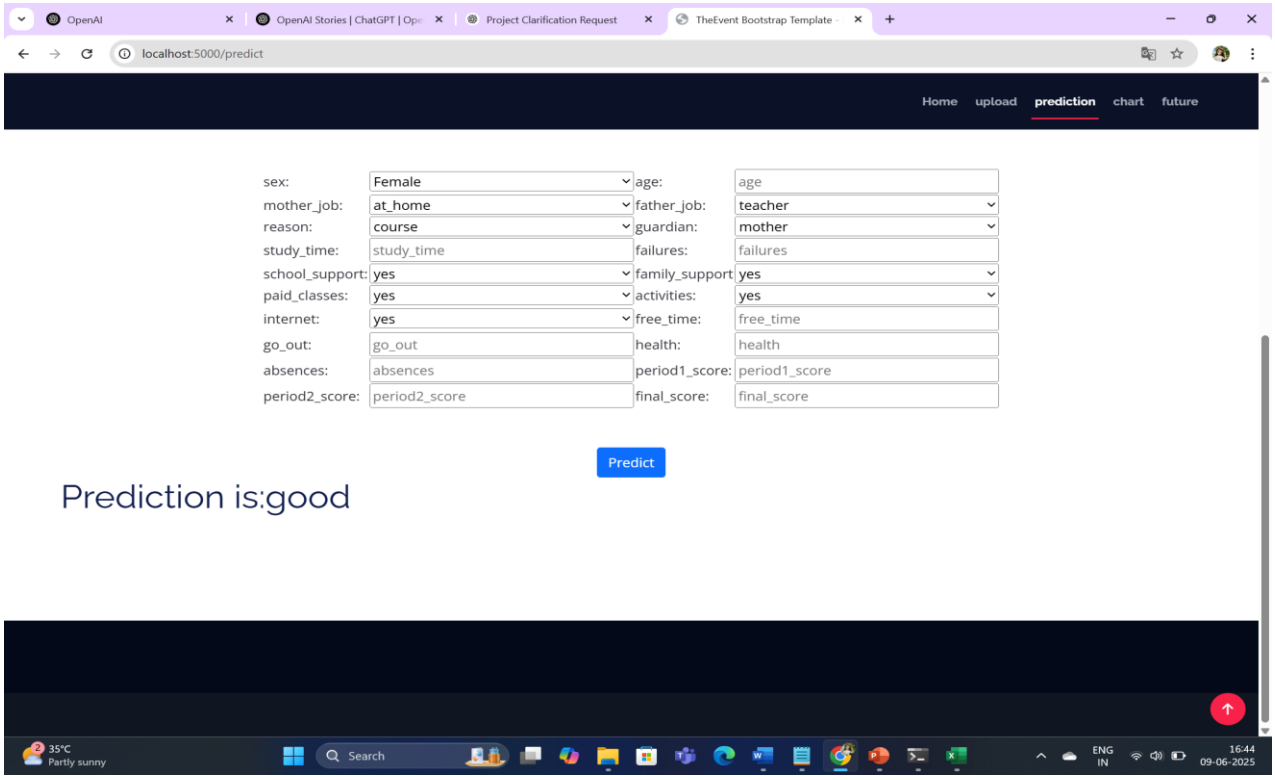


FIG 10.2.10: RESULT 1

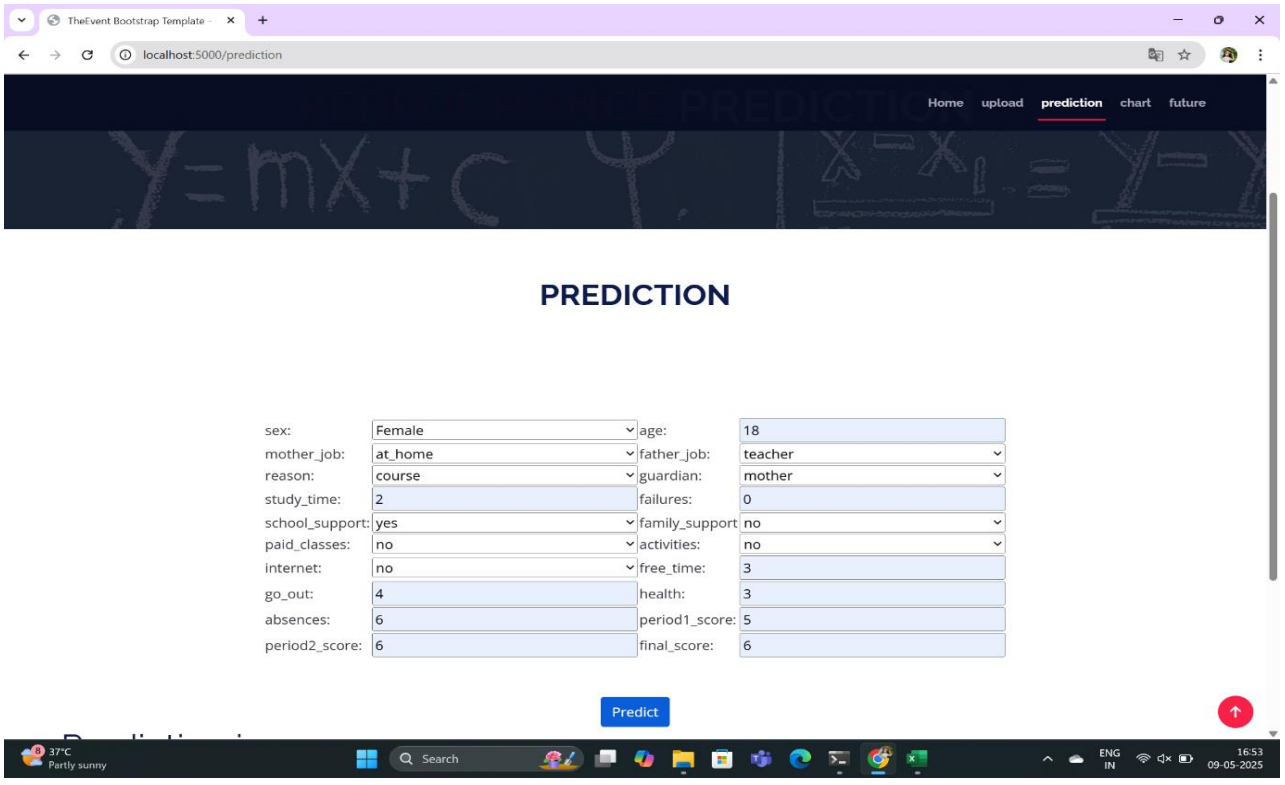


FIG 10.2.11: ENTER DETAILS 2

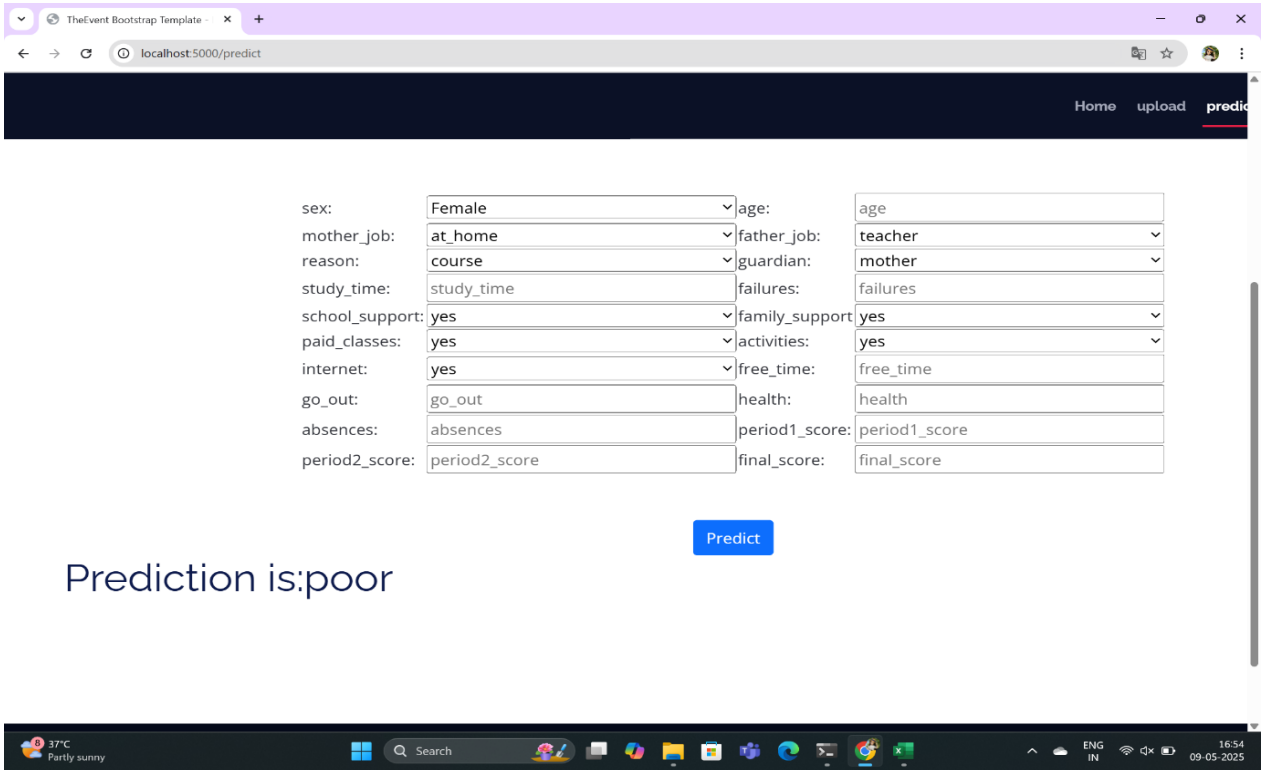


FIG 10.2.12: RESULT 2

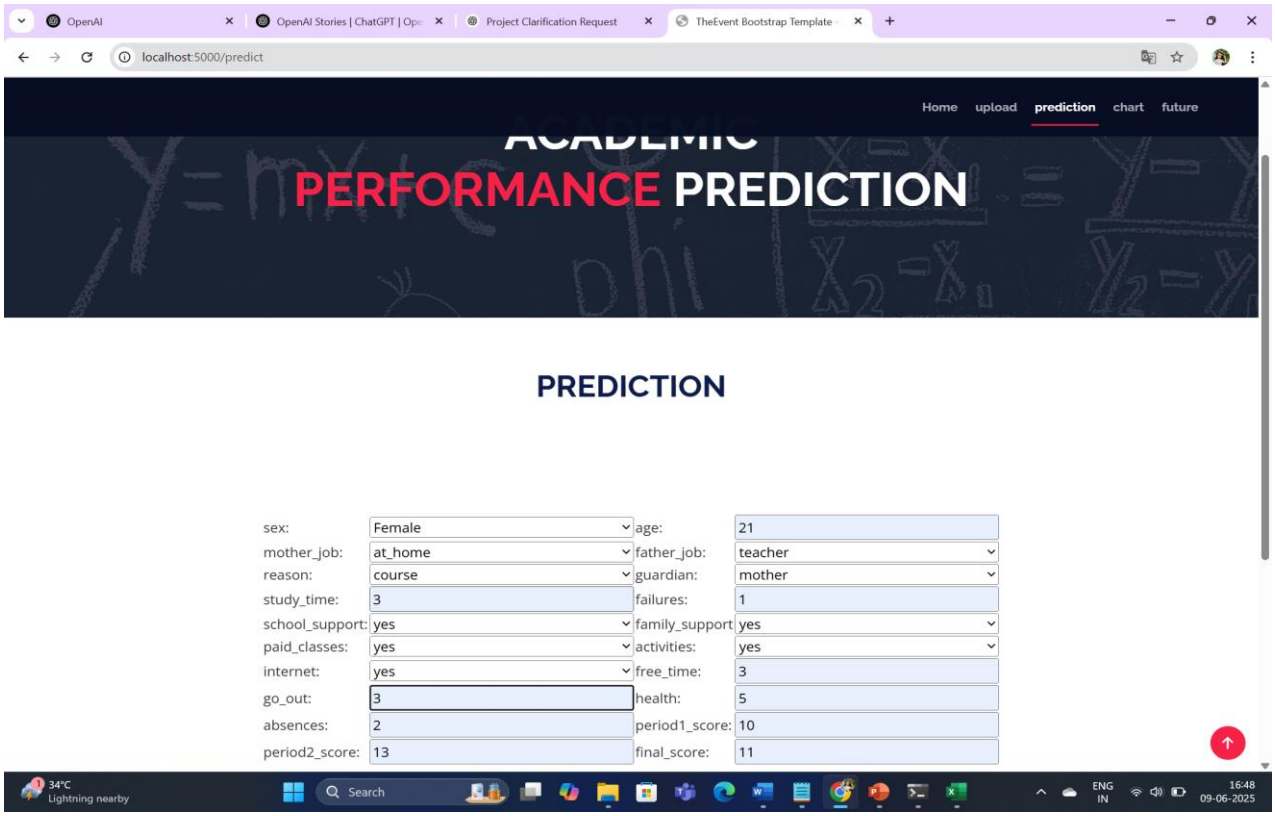


FIG 10.2.13: ENTER DETAILS 3

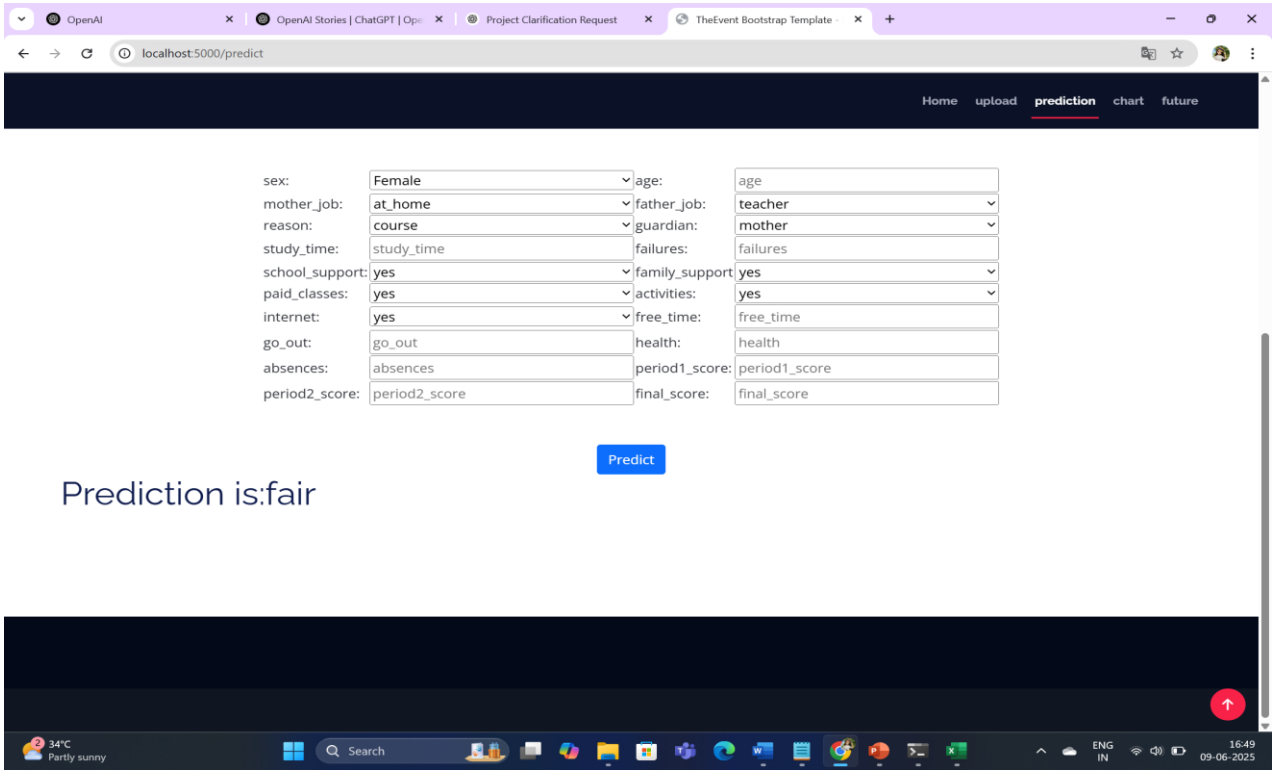


FIG 10.2.14: PREDICTION 3

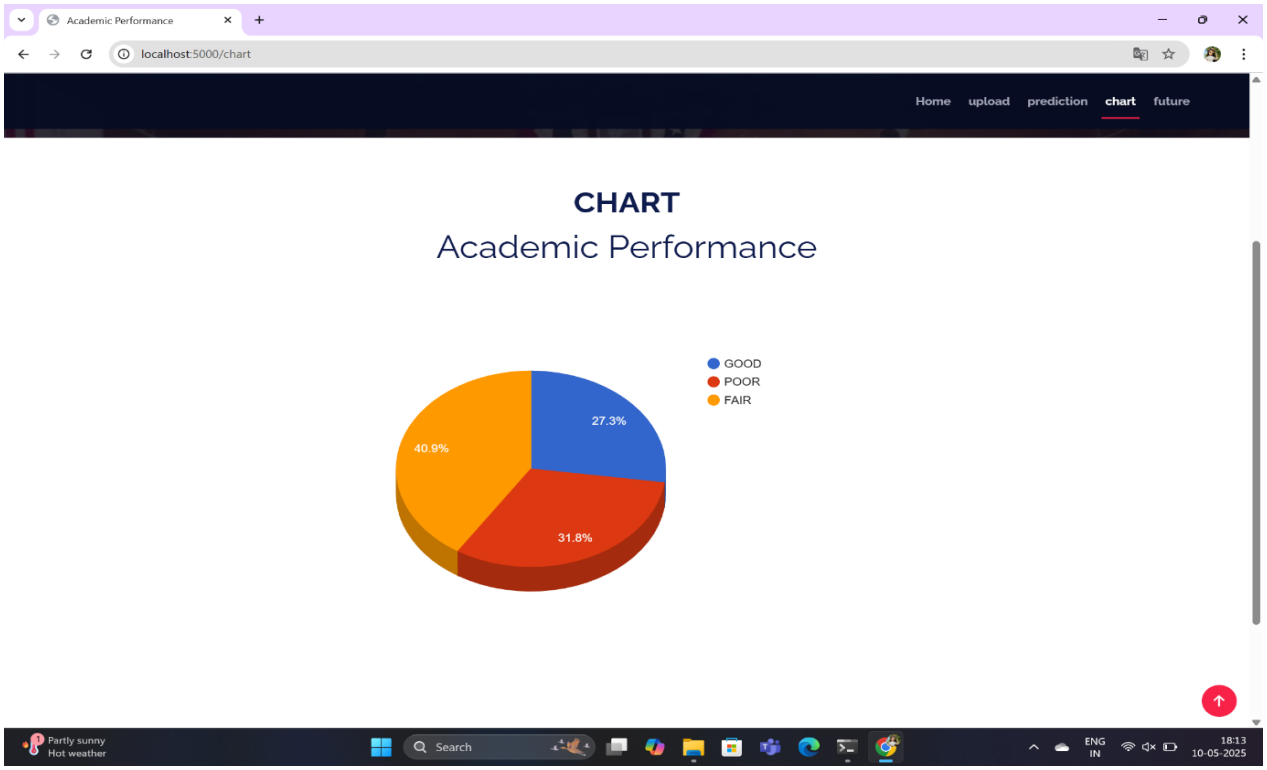


FIG 10.2.15: CHART

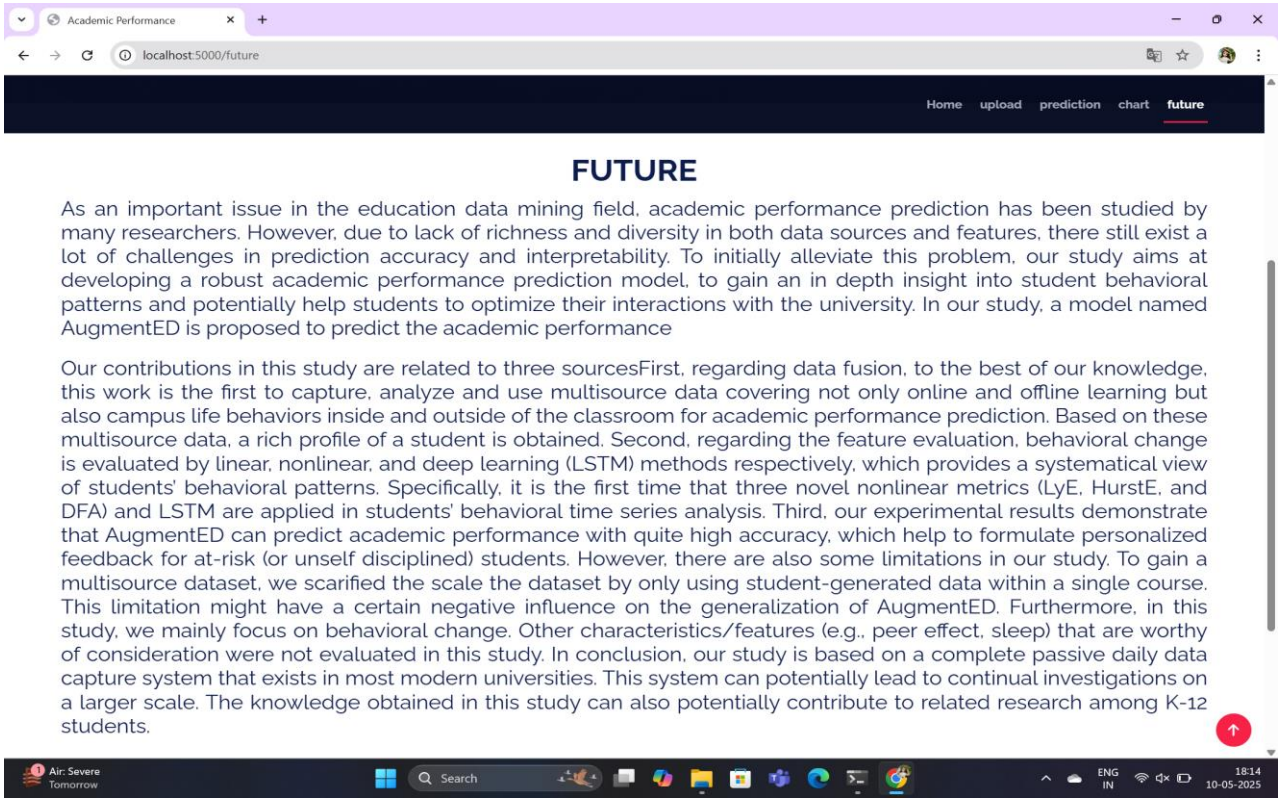


FIG 10.2.16: FUTURE

11. REFERENCES

[1] A. Furnham, and J. Monsen, “Personality traits and intelligence predict academic school grades," Learning and Individual Differences, vol. 19, no. 1, pp. 0-33, 2009.

[2] M. A. Conard, “Aptitude is not enough: How personality and behavior predict academic performance,” Journal of Research in Personality, vol. 40, no. 3, pp. 339-346, 2006.

[3] T. Chamorropremuzic, and A. Furnham, “Personality predicts academic performance: Evidence from two longitudinal university samples,” Journal of Research in Personality, vol. 37, no. 4, pp. 319- 338, 2003.

[4] R. Langford, C. P. Bonell, H. E. Jones, T. Pouliou, S. M. Murphy, and E. Waters, “The WHO health promoting school framework for improving the health and well-being of students and their academic achievement,” Cochrane Database of Systematic Reviews, vol. 4, no. 4, pp. CD008958, 2014.

[5] A. Jones, and K. Issroff, “Learning technologies: Affective and social issues in computer-supported collaborative learning,” Computers & Education, vol. 44, no. 4, pp. 395-408, 2005.

[6] D. N. A. G. Van, E. Hartman, J. Smith, and C. Visscher, “Modeling relationships between physical fitness, executive functioning, and academic achievement in primary school children,” Psychology of Sport & Exercise, vol. 15, no. 4, pp. 319-325, 2014.

[7] R. Wang, F. Chen, Z. Chen, T. Li, and A. T. Campbell, “StudentLife: Assessing mental health, academic performance and behavioral trends of college students using smartphones,” In Proc. of the ACM International Joint Conference on Pervasive & Ubiquitous Computing, Seattle, WA, USA, 2014.

[8] R.Wang, G. Harari, P. Hao, X. Zhou, and A. T. Campbell, “SmartGPA: How smartphones can assess and predict academic performance of college students,” In Proc. of the ACM International Joint Conference on Pervasive & Ubiquitous Computing, Osaka, Japan, 2015.

[9] M. T. Trockel, M. D. Barnes, and D. L. Egget, “Health-related variables and academic performance among first-year college students: Implications for sleep and other behaviors,” Journal of American College Health, vol. 49, no. 3, pp. 125-131, 2000.

[10] D. M. Hansen, S. D. Herrmann, K. Lambourne, J. Lee, and J. E. Donnelly, “Linear/nonlinear relations of activity and fitness with children’s academic achievement,” Med Sci Sports Exerc. vol. 46, no. 12, pp. 2279-2285, 2014.

[11] A. K. Porter, K. J.Matthews, D. Salvo, and H. W. Kohl, “Associations of physical activity, sedentary time, and screen time with cardiovascular fitness in United States adolescents: Results from the NHANES national youth fitness survey (NNYFS),” Journal of Physical Activity and Health, pp. 1-21, 2017.

[12] K. N. Aadland, O. Yngvar, A. Eivind, K. S. Bronnick, L. Arne, and G. K. Resaland, “Executive functions do not mediate prospective relations between indices of physical activity and academic performance: the active smarter kids (ask) study,” Frontiers in Psychology, vol. 8, pp. 1088, 2017.

[13] M. Credé, S. G. Roch, and U. M. Kieszczynska, "Class attendance in college: A meta-analytic review of the relationship of class attendance with grades and student characteristics," Review of Educational Research, vol. 80, no. 2, pp. 272-295, 2010.

[14] S. P. Gilbert, and C. C. Weaver, “Sleep quality and academic performance in university students: A wake-up call for college psychologists,” Journal of College Student Psychotherapy, vol. 24, no. 4, pp. 295-306, 2010.

[15] A. Wald, P. A. Muennig, K. A. O"Connell, and C. E. Garber, “Associations between healthy lifestyle behaviors and academic performance in U.S. undergraduates: A secondary analysis of the American college health association\"s National College Health Assessment II,” American Journal of Health Promotion, vol. 28, no. 5, pp. 298-305, 2014.

[16] P. Scanlon, and A. Smeaton, “Identifying the impact of friends on their peers academic performance,” In Proc. of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), San Francisco, CA, USA, 2016.