EDUINSIGHT STUDENT PERFORMANCE ANALYSIS & LEARNING STRATEGIES RECOMMENDATOIN SYSTEM

MINI PROJECT
ADD334

Mini Project Report submitted in partial fulfillment for the degree
of
B.Tech Artificial Intelligence & Data Science

Submitted by

AYSHA SANAM UMMER (Reg No: MUT21AD016)
BETTINA MARIAM BIJU (Reg No: MUT21AD018)
FATHIMA ZANHA (Reg No: MUT21AD024)
SAFUAN P ANVAR (Reg No: MUT21AD049)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE
(Affiliated to APJ Abdul Kalam Technological University)

May 2024

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

AYSHA SANAM UMMER (MUT21AD016)

BETTINA MARIAM BIJU (MUT21AD018)

FATHIMA ZANHA (MUT21AD024)

SAFUAN P ANVAR (MUT21AD049)

Place:

Date:

Muthoot
Institute of Technology & Science

Varikoli P.O., Puthencruz-682308

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CERTIFICATE

This is to certify that the Mini Project report entitled **"EDUINSIGHT STUDENT PERFORMANCE ANALYSIS & LEARNING STRATEGIES RECOMMENDATOIN SYSTEM"** submitted by **AYSHA SANAM UMMER (Reg No: MUT21AD016), BETTINA MARIAM BIJU (Reg No: MUT21AD018), FATHIMA ZANHA (Reg No: MUT21AD024), SAFUAN P ANVAR (Reg No: MUT21AD049)** of Semester VI is a bonafide account of the work done by them under our supervision during the academic year 2023-24.

Project Guide	**Project Coordinator**	**Head of the Department**
Dr.Cerene Mariam Abraham	Ms. Jayalekshmi J	Dr. Sumam Mary Idicula
Asst. Professor	Asst. Professor	Professor
Dept. of AI&DS	Dept. of AI&DS	Dept. of AI&DS

ACKNOWLEDGEMENT

We are grateful to Almighty who has blessed us with good health, committed and continuous interest throughout the project work.

We express our sincere thanks to our project guide, **Dr.Cerene Mariam Abraham**, Assistant Professor, Department of Artificial Intelligence and Data Science, Muthoot Institute of Technology and Science for her guidance and support which were instrumental in all the stages of the project work and without whom the project could not have been accomplished.

We are grateful to our project coordinator **Ms.Linu Paulose and Ms.Jayalekshmi J**, Assistant Professor, Department of Artificial Intelligence and Data Science, Muthoot Institute of Technology and Science, for her guidance and support.

We also wish to express our sincere appreciation to **Dr.Sumam Mary Idicula**, Professor & Head of the Department, Muthoot Institute of Technology and Science, who was willing to spend her precious time to give some ideas and suggestion towards this project.
We would like to thank **Dr. Neelakantan P.C.**, Principal, Muthoot Institute of Technology and Science, Varikoli for providing us all the necessary facilities.

We would like to express our sincere gratitude to all the teaching and non-teaching staff of Artificial Intelligence and Data Science Department for providing a good environment.

In our daily work we have been blessed with a friendly and cheerful group of fellow students. We are very much thankful to all the members of S6 AI-DS 2021-2025 batch.

Besides this, several people have knowingly and unknowingly helped us in the successful completion of this project. We express our sincere gratitude to all of them.

Abstract

EduInsight represents a pioneering effort in developing a machine learning-powered website to meticulously analyze and enhance students' academic performance. Leveraging advanced data analytics, the software processes a diverse range of datasets, including academic grades, attendance records, and participation levels, to construct comprehensive profiles for each student. Machine learning algorithms play a pivotal role in identifying patterns, correlations, and trends within the data, providing profound insights into individual learning behaviors. The recommendation engine takes center stage by translating this analysis into actionable steps. The objective of the project is to tailor suggestions for study techniques and time management strategies. Importantly, the system continually adapts and evolves, refining recommendations based on ongoing student performance and feedback.The project further extends its capabilities with the incorporation of logistic regression for prediction, an API created using Flask, and a dynamic web interface using Django, HTML, CSS, and JavaScript. This project aims to revolutionize the educational landscape by empowering students with personalized insights and recommendations, fostering a proactive approach to learning and academic success. The implementation of this intelligent educational analytics system has the potential to contribute significantly to the optimization of learning environments and the overall improvement of student performance.

Contents

List of Figures

List of Abbrevations

ANN	Artificial Neural Network
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets

Chapter 1

INTRODUCTION

1.1 Purpose

The purpose of our software is to revolutionize the way student performance is analyzed and improved. By leveraging advanced data analytics and machine learning techniques, our platform aims to provide personalized recommendations tailored to each student's unique learning style and needs.

Through comprehensive performance analysis, our software will identify patterns and trends in students' academic progress. This includes factors such as strengths and weaknesses in different subjects, preferred learning modalities, and optimal study habits. With this information, our platform will generate targeted recommendations for optimizing study schedules, and employing effective study techniques.

By empowering students with actionable insights and personalized guidance, our software aims to enhance academic performance, increase motivation, and foster a culture of continuous improvement in learning outcomes. Ultimately, our goal is to support students in their mission to success and achievement.

1.2 Intended Audience

The intended audience for our software primarily consists of students across various educational levels, from primary school to higher education institutions. Specifically, the software is designed to cater to:

Students:

The software serves as a valuable tool for students seeking to enhance their academic performance. By providing personalized recommendations on study techniques and time management strategies, students can optimize their learning experience and achieve better results in their studies.

Overall, the application has the potential to benefit a wide range of students by improving their performance

1.3 Product Scope

The product scope for our software, designed to enhance student performance through personalized recommendations and analysis, encompasses the following features and functionalities:

- **Performance Tracking and Analysis:**

  The software will enable students to track their academic performance across various subjects and assessments. It will analyze performance data to identify trends, strengths, and areas for improvement.

- **Personalized Recommendations:**

  Based on the analysis of performance data, the software will generate personalized recommendations for students. These recommendations may include suggested study techniques and time management strategies tailored to each student's learning style and needs.

- **Machine Learning Algorithms:**

  The core functionality of the software will leverage machine learning algorithms to analyze the collected performance data. These algorithms will identify patterns, correlations, and trends in students' academic performance.

- **Predictive Analytics:**

  Using machine learning models, the software will provide predictive analytics to forecast future academic performance based on historical data. This will enable early intervention and personalized support for students at risk of falling behind

- **User-Friendly Interface:**

  The software will feature a user-friendly interface designed for easy navigation and accessibility. Students can easily input data, view recommendations, and track their progress without requiring extensive technical knowledge.

Overall, the product scope aims to empower students with personalized recommendations and insights to optimize their learning experience, improve academic performance, and achieve their educational goals.

Chapter 2

PROPOSED WORK

2.1 Objectives

- **To analyze student performance:**
  Analyzing student performance involves examining assessment data, attendance, participation, and behavioral observations to understand students' academic progress, strengths, and areas needing improvement.

- **To provide personlized feedback:**
  Providing personalized feedback involves tailoring responses, suggestions, or evaluations to individual needs, preferences, and performance. This often includes specific comments, recommendations, or guidance based on an individual's unique characteristics, progress, and goals.

- **To improve performance:**
  Improving performance entails implementing strategies, interventions, or initiatives aimed at enhancing individuals' skills, knowledge, and outcomes. This may involve targeted actions, support, or resources tailored to address specific areas of weakness, capitalize on strengths, and optimize overall performance.

2.2 Problem statement

Design and Development of a system that analyses student performance and provides personalised learning and time management techniques using machine learning techniques to improve student's academic performance.
The project aims to create a comprehensive system leveraging machine learning techniques to analyze student performance data. This system will provide personalized learning recommendations

and time management strategies tailored to individual students, with the overarching goal of enhancing their academic performance. By utilizing advanced algorithms, it seeks to optimize the learning process and empower students to maximize their potential through targeted interventions and support.

2.3 Existing System and Proposed Solution

- **Mining Educational Data to Predict Student's academic Performance using Ensemble Methods:**

  Overview: The project aims to develop a cutting-edge model for predicting student performance by incorporating behavioral features alongside traditional academic metrics. Leveraging educational data mining techniques, the model emphasizes the importance of behavioral aspects in understanding and predicting student outcomes. Through the use of various classifiers and ensemble methods, the project seeks to achieve substantial improvements in accuracy compared to previous models.

  The project represents a significant advancement in the field of educational analytics, offering valuable insights into student performance and behavior. By leveraging advanced machine learning techniques and emphasizing the integration of behavioral features, it holds promise for informing more personalized and effective educational interventions, ultimately contributing to improved student outcomes and learning experiences.

  Advantages:
  - Comprehensive Analysis: The inclusion of behavioral features allows for a more holistic understanding of student performance, capturing factors beyond just academic achievements.

  - Enhanced Accuracy: Evaluation results demonstrate significant improvements in accuracy, with up to a 22.1% enhancement achieved through the incorporation of behavioral features.

  - Diverse Techniques: By utilizing classifiers such as Artificial Neural Networks (ANN), Naïve Bayesian, and Decision Trees, alongside ensemble methods like Bagging, Boosting, and Random Forest, the model benefits from a diverse set of techniques to refine predictions and mitigate biases.

  - Informed Interventions: The insights generated by the model can inform personalized interventions and support strategies, enabling educators to tailor their approaches to individual student needs.

  Techniques Used:
  - Educational Data Mining: The project utilizes techniques from educational data mining to extract valuable insights from student data, with a specific emphasis on behavioral features.

- Classifiers: Various classifiers including ANN, Naïve Bayesian, and Decision Trees are employed to predict student performance based on the collected data.

- Ensemble Methods: Ensemble methods such as Bagging, Boosting, and Random Forest are applied to combine multiple models to improve prediction accuracy.

Disadvantages:

- Risk of Overfitting: There is a risk of overfitting, particularly when incorporating a large number of features or when the model is excessively complex. Careful regularization and validation techniques must be employed to mitigate this risk.

- Data Quality: The accuracy and reliability of predictions are heavily dependent on the quality of the input data. Inaccurate or incomplete data could lead to biased or erroneous predictions, highlighting the importance of data cleaning and preprocessing steps.

- **Learning Styles: Concept and Evidence**

Overview: The project aims to understand and accommodate individual strengths and preferences across various elements to enhance long-term memory and retention effectively. Rather than assigning graded scores on different dimensions, the project seeks to classify individuals into distinct groups based on their unique characteristics. By emphasizing the uniqueness of each individual, educators can tailor their approaches to better meet the diverse needs of their students. However, achieving this goal requires large and robust datasets to capture the complexity and variability of individual learning profiles.

The project aims to enhance long-term memory and retention by understanding and accommodating individual strengths and preferences. By treating individuals as unique entities and employing advanced machine learning techniques, educators can personalize learning experiences to better meet the diverse needs of their students, ultimately leading to improved educational outcomes. However, challenges such as data quality, ethical considerations, and the complexity of classification must be carefully addressed to ensure the success and fairness of the project.

Advantages:

- Personalized Learning: By classifying individuals into distinct groups based on their unique strengths and preferences, the project enables educators to personalize learning experiences to better match the needs of each student.

- Improved Long-Term Memory and Retention: Understanding individual learning profiles allows for the development of targeted strategies to enhance long-term memory and retention, leading to more effective learning outcomes.

- Enhanced Engagement: Tailoring educational experiences to individual preferences can

increase student engagement and motivation, fostering a more positive and effective learning environment.

- Better Educational Outcomes: By treating individuals as unique entities and accommodating their individual strengths and preferences, educators can potentially improve overall educational outcomes and student success rates.

Techniques Used:

- Clustering Algorithms: Clustering algorithms such as K-means or hierarchical clustering may be employed to group individuals with similar characteristics into distinct clusters.

- Dimensionality Reduction:** Techniques like Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbor Embedding (t-SNE) may be used to reduce the dimensionality of the data while preserving important information.

- Feature Engineering: The project may involve the creation of new features or variables based on individual strengths and preferences to better capture the diversity of learning profiles.

- Predictive Modeling: Machine learning algorithms such as Decision Trees, Random Forests, or Neural Networks may be employed to predict individuals' learning preferences and behaviors based on the collected data.

Disadvantages:

- Complexity of Classification: Classifying individuals into distinct groups based on their unique characteristics can be challenging and may oversimplify the complexity of human learning behaviors.

- Data Quality and Size: Large and robust datasets are essential to accurately capture the variability of individual learning profiles. However, collecting and maintaining such datasets can be resource-intensive and may pose challenges in terms of data quality and privacy concerns.

- Ethical Considerations: Classifying individuals into distinct groups based on their characteristics raises ethical considerations regarding stereotyping, bias, and fairness. Careful consideration and mitigation of these issues are necessary to ensure equitable treatment for all individuals.

- **Student Performance Analysis using Machine Learning:**

Overview: The project aims to predict student performance to aid teachers in identifying strengths, weaknesses, and potential failures, thus fostering student growth. Employing Data Mining techniques, specifically the Support Vector Machine (SVM) algorithm, the project analyzes diverse datasets to generate predictive insights. Techniques such as Decision Trees,

K-Nearest Neighbors (KNN), Random Forest Classifier, and AdaBoost are utilized alongside data set collection, feature selection, and data preprocessing methods.

The project holds promise for enhancing student support and improving teaching strategies through predictive modeling techniques. However, careful attention must be paid to address privacy concerns, ensure data quality, navigate ethical considerations, overcome integration challenges, and provide adequate training to stakeholders to maximize the project's effectiveness and impact.

Advantages:

- Improved Student Support: Predictive insights enable proactive identification of struggling students, allowing for targeted support and intervention.

- Enhanced Teaching Strategies: Tailoring teaching strategies to individual student needs promotes more effective instruction and student success.

- Efficient Resource Allocation: Anticipating student needs optimizes resource allocation, directing support services where they are most needed.

- Data-Driven Decision Making: Evidence-based teaching practices are facilitated by leveraging insights derived from large datasets.

Techniques Used:

- Support Vector Machine (SVM) Algorithm: Utilized as the primary predictive modeling technique to classify students into performance categories based on input features.

- Decision Trees, K-Nearest Neighbors (KNN), Random Forest Classifier, and AdaBoost: Employed as additional classifiers to further enhance predictive accuracy.

- Data Set Collection: Involves gathering diverse datasets related to student demographics, academic records, and behavioral attributes.

- Feature Selection: Relevant features are selected from the collected datasets to improve the efficiency and accuracy of predictive models.

- Data Preprocessing: Techniques such as normalization, outlier detection, and missing value imputation are applied to ensure the quality and consistency of input data.

Disadvantages:

- Privacy Concerns: Collection and analysis of student data raise privacy issues, necessitating safeguards to protect sensitive information.

- Data Quality: Accuracy and reliability of predictive models depend on high-quality data, necessitating careful data cleaning and validation.

- Ethical Considerations: Fairness, bias, and transparency must be addressed when using predictive modeling outcomes in decision-making.

- Integration Challenges: Integrating predictive techniques into existing systems may pose technical and organizational hurdles.

- Training Needs: Ensuring educators understand and effectively utilize predictive insights is crucial for successful implementation.

- **Predicting time-management skills from learning analytics:**

  Overview:
  The project aims to enhance student time-management skills through the integration of Learning Management Systems (LMS) and analytics, enabling the prediction of skills for preemptive academic interventions. By applying analytics for time-management prediction, the project targets less-explored time-management skills and facilitates tailored interventions to support student success.

  the project represents a valuable initiative in leveraging LMS and analytics to enhance student time-management skills and support academic success. While offering tailored interventions and improving resource allocation, challenges such as limited model portability and variable model composition must be addressed to maximize the project's impact and widespread adoption.

  Advantages:
  - Improved Time-Management: The project enables the identification and prediction of time-management skills, allowing for targeted interventions to enhance students' ability to manage their time effectively.

  - Tailored Interventions: By focusing on less-explored time-management skills, the project offers personalized interventions that address specific areas of need, thereby increasing the effectiveness of support services.

  - Enhanced Academic Performance: Through preemptive academic interventions, students can receive timely support to overcome challenges and improve their academic performance.

  - Efficient Resource Allocation: Targeting interventions based on predictive analytics optimizes resource allocation, directing support to where it is most needed.

  Techniques Used:
  - Learning Management Systems (LMS): Leveraging LMS data allows for the collection of rich information on student behavior and engagement, which can be analyzed to identify patterns and predict time-management skills.

  - Analytics: Advanced analytics techniques, such as machine learning algorithms or statistical modeling, are applied to analyze LMS data and predict time-management skills.

- Tailored Interventions: Interventions are designed based on the insights gained from analytics, with a focus on addressing specific time-management skills identified as areas of need.

Disadvantages:

- Limited Model Portability: Challenges in transferring the model to different educational contexts or institutions may impede widespread use, as the effectiveness of the model could be influenced by course-specific considerations.

- Challenges in Transferability: The variable composition of the model may pose challenges in terms of transferability, as different courses may require different approaches to time-management interventions.

- Variable Model Composition: The effectiveness of the model may vary depending on the composition of the courses and the specific time-management skills targeted, which could impact the generalizability of the findings.

Chapter 3

PROJECT DESIGN

3.1 System Architecture

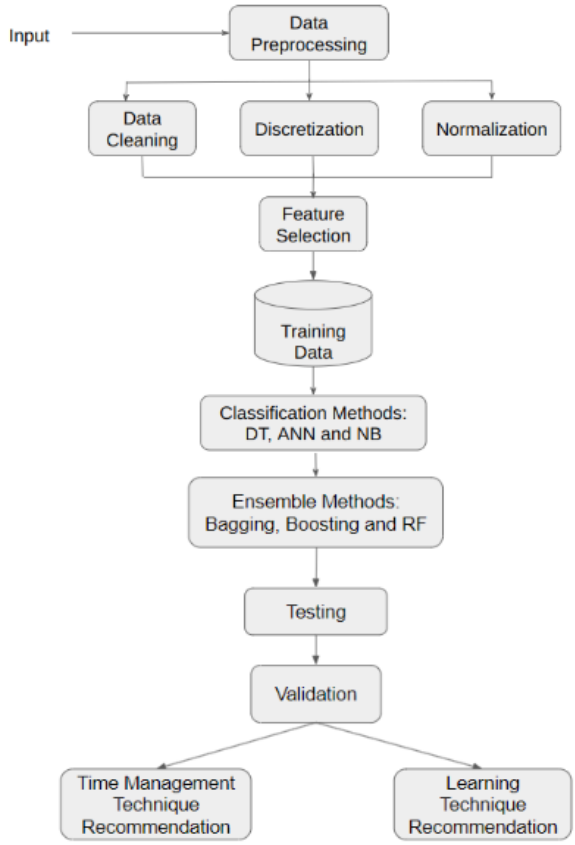

Figure 3.1: System Architecture

3.2 Modules

- **Data Preprocessing:**
  This module involves cleaning, transforming, and preparing the raw data for analysis. It includes tasks such as handling missing values, removing outliers, and standardizing or normalizing the data to ensure its suitability for machine learning algorithms.

- **Feature Selection:**
  Feature selection is the process of choosing the most relevant features from the dataset that contribute the most to the predictive model's performance. This module selects the most informative features to improve the efficiency and accuracy of the machine learning algorithms.

- **Model Selection:**
  In this module, various machine learning algorithms are evaluated and compared to identify the best-performing model for the given dataset and problem. Different algorithms such as decision trees, support vector machines, and neural networks are tested to determine which one produces the most accurate predictions.

- **Feedback Generation:**
  Feedback generation involves providing insights and recommendations based on the analysis of student data. This module interprets the results from the selected model and generates personalized feedback for students, including study techniques and time management strategies tailored to their individual learning behaviors and performance patterns.

3.3 Methodology

Preprocessing:

- Discretization

```
#Discretization

#choose column to discretize
column_to_discretize="raisedhands"
#Define the number of bins or custom bin edges
num_bins=5
#Perform discretization using pandas cut function
data['discretized_raisedhands']=pd.cut(data[column_to_discretize],bins=num_bins, labels=False)
#Display
data.head()
```

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16	2	20
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20	3	25
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7	0	30
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25	5	35
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50	12	50

Figure 3.2: Discretization

- Label Encoding

```
#Label Encoding
# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Iterate through each column in the dataset
for column in data.columns:
    # Check if the column data type is object (string)
    if data[column].dtype == 'object':
        # Use LabelEncoder to convert strings to numerical values
        data[column] = label_encoder.fit_transform(data[column])
data.head()
```

```
<IPython.core.display.Javascript object>
```

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	AnnouncementsView	Discussion
0	1	4	4	2	1	0	7	0	0	15	16	2	20
1	1	4	4	2	1	0	7	0	0	20	20	3	25
2	1	4	4	2	1	0	7	0	0	10	7	0	30
3	1	4	4	2	1	0	7	0	0	30	25	5	35
4	1	4	4	2	1	0	7	0	0	40	50	12	50

Figure 3.3: Label Encoding

- Normalization

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
x=df.drop("Class",axis=1)
y=df['Class']
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=42)
xtrain_scaled=scaler.fit_transform(xtrain)
xtest_scaled=scaler.fit_transform(xtest)
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
a=model.fit(xtrain_scaled,ytrain)
p=a.predict(xtest_scaled)
from sklearn.metrics  import accuracy_score
acc=accuracy_score(p,ytest)*100
acc
```

```
72.22222222222221
```

Figure 3.4: Normalization

Feature Selection:

- chi square test

```
]:
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
chi2_features = SelectKBest(chi2,k=3)
x_kbest_features = chi2_features.fit_transform(xtrain_scaled,ytrain)
print("Original feature number:" , xtrain_scaled.shape[1])
print('Reduced feature number:',x_kbest_features.shape[1])
```

```
Original feature number: 13
Reduced feature number: 3
```

Figure 3.5: chi square test

- Select kbest

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
chi2_features = SelectKBest(chi2,k=3)
x_kbest_features = chi2_features.fit_transform(xtrain_scaled,ytrain)
print("Original feature number:" , xtrain_scaled.shape[1])
print('Reduced feature number:',x_kbest_features.shape[1])
```

```
Original feature number: 13
Reduced feature number: 3
```

Figure 3.6: Select kbest

- Recursive Feature Elimination

```
# Train classifier on selected features
clf.fit(X_train_selected, y_train)

# Evaluate classifier on testing data
accuracy = clf.score(X_test_selected, y_test)
print("Accuracy on Test Set:", accuracy)
```

```
Selected Features: Index(['raisedhands', 'VisITedResources', 'AnnouncementsView', 'Discussion',
       'StudentAbsenceDays'],
      dtype='object')
Accuracy on Test Set: 0.7916666666666666
```

Figure 3.7: Recursive Feature Elimination

Model Selection:

ARTIFICIAL NEURON NETWORK

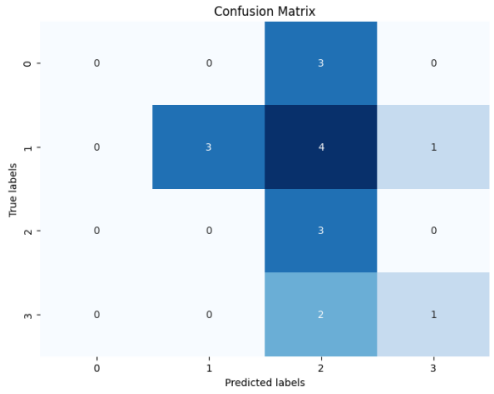

Figure 3.8: ANN Confusion Matrix

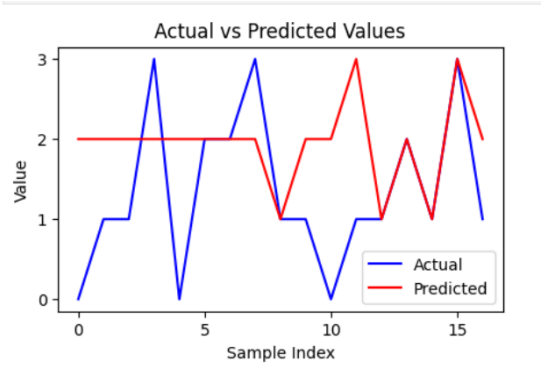

Figure 3.9: ANN Graph

DECISION TREE

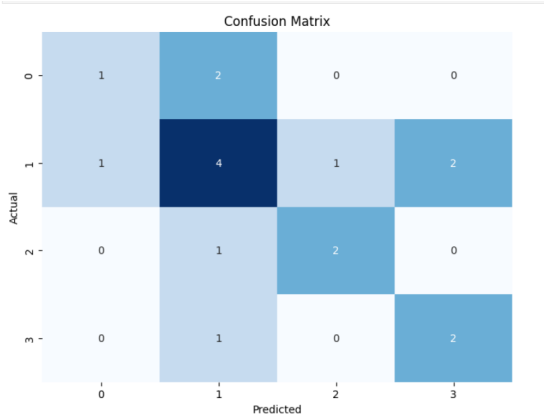

Figure 3.10: Decision Tree Confusion Matrix

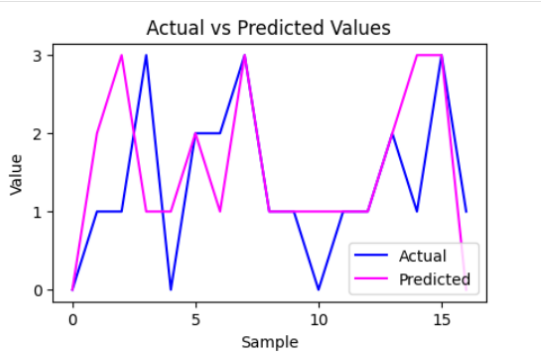

Figure 3.11: Decision Tree Graph

LINEAR REGRESSION

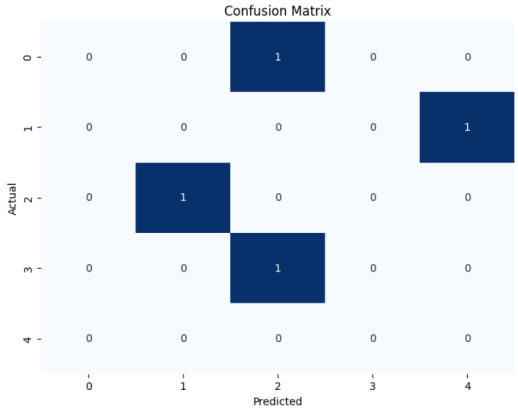

Figure 3.12: Linear Regression Confuson Matrix

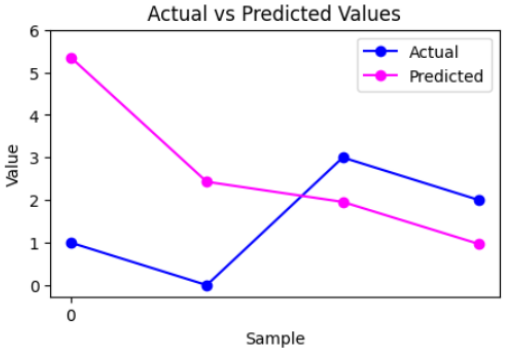

Figure 3.13: Linear Regression Grpah

NAIVE BAYES

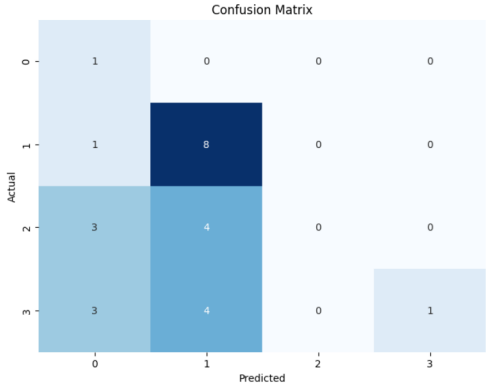

Figure 3.14: Naive Bayes Confusion Matrix

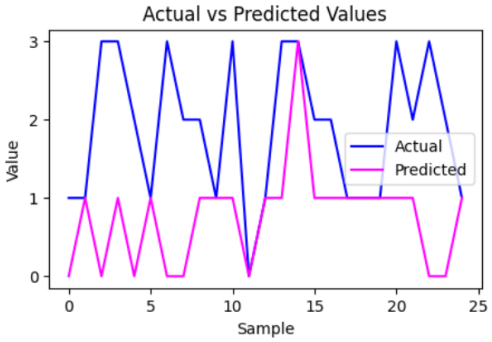

Figure 3.15: Naive Bayes Graph

3.4 Data Flow Diagram

3.4.1 DFD level 0

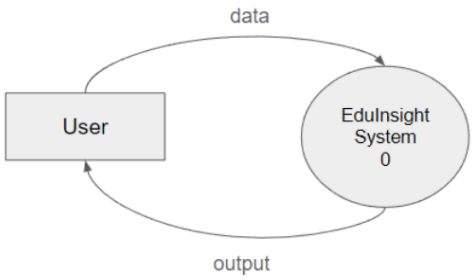

Figure 3.16: DFD level 0

3.4.2 DFD level 1

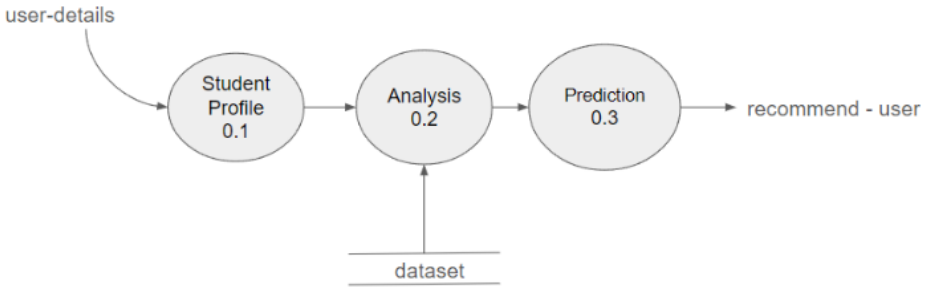

Figure 3.17: DFD level 1

3.5 Database Table Design

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
gender	backlogs	sleep	concentration	age	academic_level	satisfied	absent_days	time_techniques	info_delivery	learning_strategy	mental_activities	raise_hands	study_time	grades
Female	Zero	Less than 7hr	15min - 30min	20-25	UNDER GRADUATE	Neutral	Less than 10	pomodoro	Read/Write	feyman	avors like writing,	3	Less than 1 hour	70-90%
Female	Zero	7hr	15min - 30min	20-25	UNDER GRADUATE	Neutral	Less than 10	pomodoro	Visual	flowtime	eep and thought-p	1	Less than 1 hour	70-90%
Female	Zero	More than 7hr	15min - 30min	20-25	UNDER GRADUATE	Slightly	Less than 30	pomodoro	Auditory	flowtime	avors like writing,	1	Less than 1 hour	Greater than 90%
Female	Zero	Less than 7hr	15min - 30min	15-20	UNDER GRADUATE	Slightly	Less than 30	time_blocking	Read/Write	feyman	puzzles and brain	1	Less than 1 hour	50-70%
Male	Zero	Less than 7hr	Less than 15min	20-25	UNDER GRADUATE	vey much	Less than 30	time_blocking	Visual	feyman	ical thinking and p	4	Less than 1 hour	70-90%
Female	Zero	7hr	30min - 45min	15-20	UNDER GRADUATE	Neutral	Less than 10	pomodoro	Read/Write	smart_goals	d discussions that	2	Less than 1 hour	70-90%
Male	Zero	Less than 7hr	15min - 30min	15-20	UNDER GRADUATE	Neutral	Less than 10	pomodoro	Read/Write	flowtime	and connecting th	4	Less than 1 hour	70-90%
Female	Zero	Less than 7hr	15min - 30min	15-20	UNDER GRADUATE	Not at all	Less than 10	pomodoro	Visual	flowtime	ion through visual	0	Less than 1 hour	70-90%
Female	Zero	Less than 7hr	30min - 45min	20-25	UNDER GRADUATE	Not at all	Less than 10	pomodoro	Read/Write	flowtime	d discussions that	0	Less than 1 hour	50-70%

Figure 3.18: Database Table Design

3.6 Technology Stack

- **Programming Language: Python**

  Python is used for implementing the entire system due to its simplicity, readability, and extensive libraries for data analysis, machine learning, and web development.

- **Data Analysis and Machine Learning Libraries:**

  Pandas:

  For data manipulation and analysis, such as reading and processing student performance data.

  Scikit-learn:

  For implementing the Decision Tree algorithm and other machine learning models for predicting learning strategies and time management strategies.

  Matplotlib and Seaborn:

  For data visualization to gain insights into student performance and learning strategies.

  Jupyter Notebook:

  For interactive data analysis, model training, and experimentation.

- **Web Development Framework:**

  Flask:

  A lightweight Python web framework for building web applications. Flask is used to create the backend of the website and provide RESTful APIs for interacting with the machine learning models.

  Flask-RESTful:

  An extension for Flask that simplifies the creation of RESTful APIs. It's used to define endpoints for predicting learning strategies and time management strategies.

- **Frontend Technologies:**

  HTML, CSS, JavaScript:

  For building the user interface (UI) of the website. HTML is used for structuring the content, CSS for styling, and JavaScript for interactivity.

3.7 System Requirements

Software:

- Backend - Python
- Frontend: HTML, CSS, JavaScript
- Development Framework: Django, Flask
- Windows,Mac,Linux

Hardware:

- Processor : Ryzen 5
- Memory (RAM) : 8 GB
- Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)
- Hard Drive: Minimum 32 GB; Recommended 64 GB or more

Chapter 4

IMPLEMENTATION

Front End

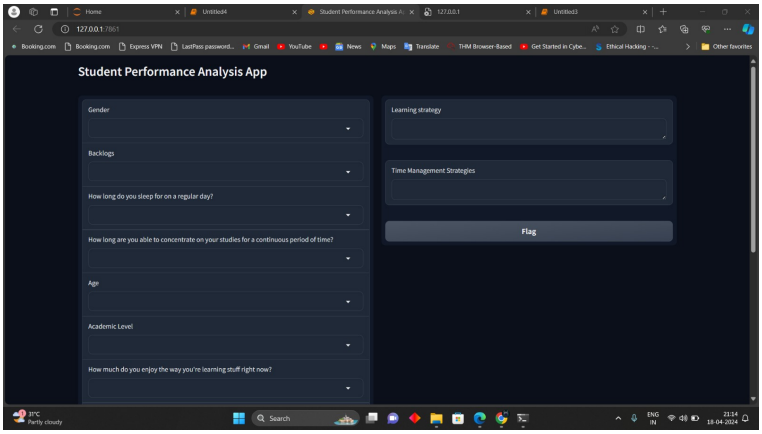

Figure 4.1: Front End 1

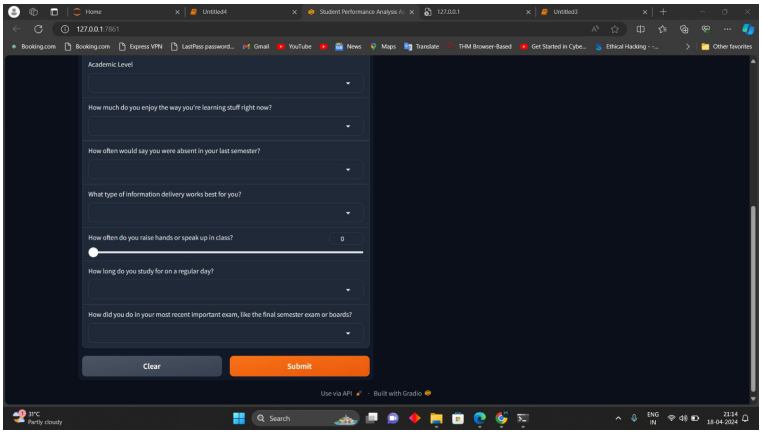

Figure 4.2: Front End 2

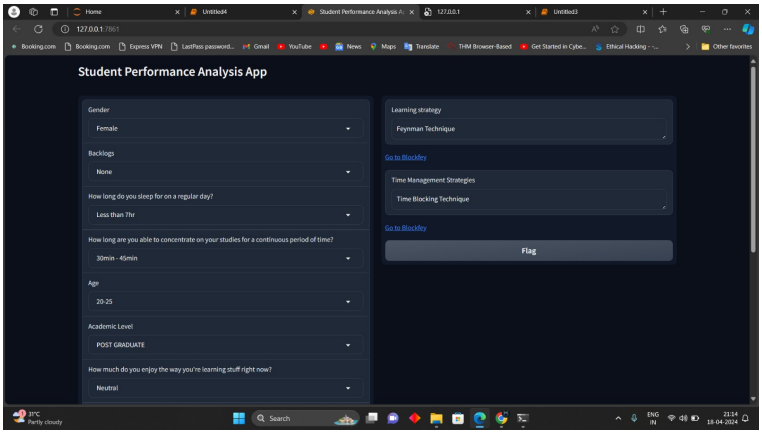

Figure 4.3: Front End 3

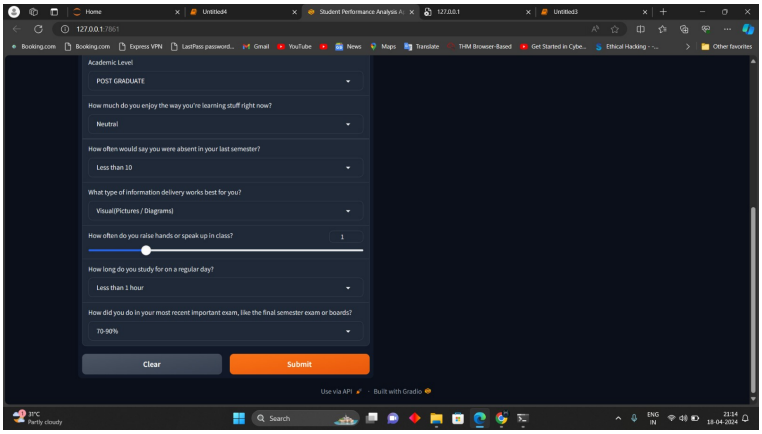

Figure 4.4: Front End 4

code

```
import pandas as pd
import joblib
from sklearn.preprocessing import LabelEncoder
# Load the input data
input_file = 'data.csv'
df = pd.read_csv(input_file)
# Create a dictionary to store the label encoders for each categorical column
label_encoders =
# Iterate over each column in the DataFrame
for column in df.columns:
# Check if the column is of type 'object' (categorical)
if df[column].dtype == 'object':
# Create a LabelEncoder for the column
label_encoders[column] = LabelEncoder()
# Fit and transform the column using the LabelEncoder
df[column] = label_encoders[column].fit_transform(df[column])
# Save the label encoders for future use (optional)
joblib.dump(label_encoders, 'label_encoders.joblib')
# Save the encoded DataFrame to a new CSV file
output_file = 'first.csv'
df.to_csv(output˙file, index=False)
print(f"Encoded data saved as {output_file}")

import pandas as pd
from sklearn.model˙selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
def training˙model(file_path='first.csv'):
# Load the data from 'first.csv'
data = pd.read˙csv(file_path)
# Separate features (X) and target variable (y)
# Assuming the target variable is named 'currently_satisfied'
X = data.drop(columns=['learning_strategy']) # Update with your
target column name
y = data['learning_strategy'] # Update with your target column name
# Convert target variable to integer if it is categorical
if y.dtype == 'object':
```

```
y = y.astype(int)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
# Create a DecisionTreeClassifier model
model = DecisionTreeClassifier(random_state=1)
# Train the model
trained_model = model.fit(X_train, y_train)
# Return the trained model
return trained_model
# Call the function to train the model
trained_model = training_model()

from sklearn.metrics import accuracy_score
def final˙prediction(trained_model, file_path='first.csv'):
# Load the data from 'first.csv'
data = pd.read_csv(file_path)
# Separate features (X) and target variable (y)
X = data.drop(columns=['learning_strategy']) # Update with your target column name
y = data['learning_strategy'] # Update with your target column name
# Convert target variable to integer if it is categorical
if y.dtype == 'object':
y = y.astype(int)
# Split the data into training and testing sets
_, X˙test, _, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
# Make predictions using the trained model
y_pred = trained_model.predict(X_test)
# Calculate the accuracy of the model on the test set
test_accuracy = accuracy_score(y_test, y_pred)
# Return the predictions and test accuracy
return y_pred, test_accuracy
# Call the final˙prediction function to obtain predictions and test accuracy
final_predictions, test_accuracy = final_prediction(trained_model)
# Display the final predictions and test accuracy
print(f"Final Predictions: final_predictions")
print(f"Test Accuracy: test_accuracy * 100:.2f%")
```

```
joblib.dump(trained_model, 'trained_model.pkl')

import pandas as pd
import joblib
from sklearn.preprocessing import LabelEncoder
# Load the input data
input_file = 'l.csv'
df = pd.read_csv(input_file)
# Create a dictionary to store the label encoders for each categorical column
label_encoders =
# Iterate over each column in the DataFrame
for column in df.columns:
# Check if the column is of type 'object' (categorical)
if df[column].dtype == 'object':
# Create a LabelEncoder for the column
label_encoders[column] = LabelEncoder()
# Fit and transform the column using the LabelEncoder
df[column] = label_encoders[column].fit_transform(df[column])
# Save the label encoders for future use (optional)
joblib.dump(label_encoders, 'label_encoders.joblib')
# Save the encoded DataFrame to a new CSV file
output_file = 'ls.csv'
df.to_csv(output_file, index=False)
print(f"Encoded data saved as output_file")
import pandas as pd
import joblib
from sklearn.preprocessing import LabelEncoder
# Load the input data
input_file = 'l.csv'
df = pd.read_csv(input_file)
# Create a dictionary to store the label encoders for each categorical column
label_encoders =
# Iterate over each column in the DataFrame
for column in df.columns:
# Check if the column is of type 'object' (categorical)
if df[column].dtype == 'object':
# Create a LabelEncoder for the column
```

```
label_encoders[column] = LabelEncoder()
# Fit and transform the column using the LabelEncoder
df[column] = label_encoders[column].fit_transform(df[column])
# Save the label encoders for future use (optional)
joblib.dump(label_encoders, 'label_encoders.joblib')
# Save the encoded DataFrame to a new CSV file
output_file = 'ls.csv'
df.to_csv(output_file, index=False)
print(f"Encoded data saved as {output_file}")

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
def training_model(file_path='ls.csv'):
# Load the data from 'first.csv'
data = pd.read_csv(file_path)
# Separate features (X) and target variable (y)
# Assuming the target variable is named 'currently_satisfied'
X = data.drop(columns=['time-management']) # Update with your target column name
y = data['time-management'] # Update with your target column name
# Convert target variable to integer if it is categorical
if y.dtype == 'object':
y = y.astype(int)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
# Create a DecisionTreeClassifier model
model = DecisionTreeClassifier(random_state=1)
# Train the model
trained_model = model.fit(X_train, y_train)
# Return the trained model
return trained_model
# Call the function to train the model
trained_model = training_model()

from sklearn.metrics import accuracy_score
def final_prediction(trained_model, file_path='ls.csv'):
# Load the data from 'first.csv'
```

```
data = pd.read_csv(file_path)
# Separate features (X) and target variable (y)
X = data.drop(columns=['time-management']) # Update with your target column name
y = data['time-management'] # Update with your target column name
# Convert target variable to integer if it is categorical
if y.dtype == 'object':
y = y.astype(int)
# Split the data into training and testing sets _, X_test, _, y_test = train_test_split(X, y,
test_size=0.3, random_state=1)
# Make predictions using the trained model
y_pred = trained_model.predict(X_test)
# Calculate the accuracy of the model on the test set test_accuracy = accuracy_score(y_test,
y_pred)
# Return the predictions and test accuracy
return y_pred, test_accuracy
# Call the final_prediction function to obtain predictions and test accuracy
final_predictions, test_accuracy = final_prediction(trained_model)
# Display the final predictions and test accuracy
print(f"Final Predictions: final˙predictions")
print(f"Test Accuracy: test_accuracy * 100:.2f%")

joblib.dump(trained_model, 'model7.pkl')

import gradio as gr
import numpy as np
from joblib import load
# Load the models from files
trained_model = load('trained_model.pkl')
model4 = load('model4.pkl')
# Define dictionaries to map each dropdown option to a specific number
gender_mapping = "Male": 0, "Female": 1
backlogs_mapping = "None": 0, "Less than 5": 1, "Greater than 5": 2
sleep_mapping = "Less than 7hr": 0, "7hr": 1, "More than 7hr": 2
concentration_mapping = "15min - 30min": 0, "30min - 45min": 1, "More than 45min": 2
age_mapping = "15-20": 0, "20-25": 1, "25-30": 2
academic_level_mapping = "HIGHER SECONDARY": 0, "UNDER GRADUATE": 1, "POST
GRADUATE": 2
```

```
satisfaction_mapping = "Not at all": 0, "Neutral": 1, "Slightly": 2, "Extremely": 3
absent_days_mapping = "Less than 10": 0, "Less than 30": 1, "More than 30": 2
info_delivery_mapping =
"Read/Write": 0,
"Visual(Pictures / Diagrams)": 1,
"Auditory": 2,
"Kinaesthetic (learning by physical activity and hands-on experiences.)": 3
study_time_mapping = "Less than 1 hour": 0, "1-3 hours": 1, "more than 3 hours": 2
grades_mapping = "Less than 50%": 0, "50-70%": 1, "70-90%": 2, "Greater than 90%": 3
# Define dictionaries to map predictions from models to specific text
trained_model_text_mapping =
0: "Eisenhower Matrix ",
1: "Feynman Technique",
2:"Flowtime Technique",
3:"SMART Goals Technique"
model4_text_mapping =
0: "Pomodoro Technique",
1: "Time Blocking Technique"
# Define the predict function
def predict(gender, backlogs, sleep, concentration, age, academic_level, currently_satisfied,
absent_days, info_delivery, raise_hands, study_time, grades):
# Convert each input using the mapping dictionaries
gender_num = gender_mapping[gender]
backlogs_num = backlogs_mapping[backlogs]
sleep_num = sleep_mapping[sleep]
concentration_num = concentration_mapping[concentration]
age_num = age_mapping[age]
academic_level_num = academic_level_mapping[academic_level]
satisfaction_num = satisfaction_mapping[currently_satisfied]
absent_days_num = absent_days_mapping[absent_days]
info_delivery_num = info_delivery_mapping[info_delivery]
study_time_num = study_time_mapping[study_time]
grades_num = grades_mapping[grades]
# Convert raise˙hands directly as it's already a number
raise_hands_num = raise_hands
# Combine the input data into an array
input_data = [
```

```
gender_num,
backlogs_num,
sleep_num,
concentration_num,
age_num,
academic_level_num,
satisfaction_num,
absent_days_num,
info_delivery_num,
raise_hands_num,
study_time_num,
grades_num
]
# Convert the list to a numpy array and reshape it for the models
input_data_reshaped = np.array(input_data).reshape(1, -1)
# Predict using the models
y_pred_trained_model = trained_model.predict(input_data_reshaped)
y_pred_model4 = model4.predict(input_data_reshaped)
# Retrieve text corresponding to the predictions from the dictionaries
trained_model˙text = trained_model_text_mapping[y_pred_trained_model[0]]
model4_text = model4_text_mapping[y_pred_model4[0]]
# Generate redirection links based on predictions
redirect_link_trained_model = '<a href="blockfey.html">Go to Blockfey</a>' if y_pred_trained_model[0]
== 1 else '<a href="blockeisen.html">Go to Blockeisen< /a >' redirect_link_model4 = '<a
href="blockfey.html">Go to Blockfey</a>' if y_pred_model4[0] == 1 else
'<a href="blockeisen.html">Go to Blockeisen</a>'
# Return the predictions and redirection links
return trained_model_text, redirect_link_trained_model, model4_text, redirect_link_model4
# Create the Gradio interface
iface = gr.Interface(
fn=predict,
inputs=[
gr.Dropdown(["Male", "Female"], label="Gender"),
gr.Dropdown(["None", "Less than 5", "Greater than 5"], label="Backlogs"),
gr.Dropdown(["Less than 7hr", "7hr", "More than 7hr"], label="How long do you sleep for on
a regular day?"),
gr.Dropdown(["15min - 30min", "30min - 45min", "More than 45min"], label="How long are
```

```
you able to concentrate on your studies for a continuous period of time?"),
gr.Dropdown(["15-20", "20-25", "25-30"], label="Age"),
gr.Dropdown(["HIGHER SECONDARY", "UNDER GRADUATE", "POST GRADUATE"],
label="Academic Level"),
gr.Dropdown(["Not at all", "Neutral", "Slightly", "Extremely"], label="How much do you
enjoy the way you're learning stuff right now?"),
gr.Dropdown(["Less than 10", "Less than 30", "More than 30"], label="How often would say
you were absent in your last semester?"),
gr.Dropdown([
"Read/Write",
"Visual(Pictures / Diagrams)",
"Auditory",
"Kinaesthetic (learning by physical activity and hands-on experiences.)"
], label="What type of information delivery works best for you?"),
gr.Slider(0, 5, step=1, label="How often do you raise hands or speak up in class?"),
gr.Dropdown(["Less than 1 hour", "1-3 hours", "more than 3 hours"], label="How long do you
study for on a regular day?"),
gr.Dropdown(["Less than 50%", "50-70%", "70-90%", "Greater than 90%"], label="How did
you do in your most recent important exam, like the final semester exam or boards?")
],
outputs=[
gr.Textbox(label="Learning strategy"),
gr.HTML(label="Redirection link from trained_model"),
gr.Textbox(label="Time Management Strategies"),
gr.HTML(label="Redirection link from model4")
],
title="Student Performance Analysis App"
)
# Launch the Gradio app
iface.launch()
```

Chapter 5

CONCLUSION

EduInsight's innovative approach not only enhances academic performance but also fosters a culture of personalized learning. By delving deep into student data and leveraging machine learning, it offers tailored interventions that address individual learning needs. As EduInsight continues to evolve and refine its methodologies, it holds the potential to not only improve learning outcomes but also revolutionize the way education is approached and delivered. Its user-friendly interface makes it accessible to students and educators alike, paving the way for a more collaborative and effective learning environment. With its promise of empowering students and educators with actionable insights, EduInsight stands at the forefront of educational innovation, driving towards a future where every student can thrive academically.

5.1 References

1. Amrieh, E. A., Hamtini, T., & Aljarah, I. (2016). Mining educational data to predict student's academic performance using ensemble methods. International journal of database theory and application, 9(8).
2. Pashler, H., McDaniel, M., Rohrer, D., & Bjork, R. (2008). Learning styles: Concepts and evidence. Psychological science in the public interest.
3. Van Sluijs, M., & Matzat, U. (2023). Predicting time-management skills from learning analytics. Journal of Computer Assisted Learning.
4. Gobbo, Federico, and Matteo Vaccari. "The pomodoro technique for sustainable pace in extreme programming teams." In International conference on agile processes and extreme programming in software engineering, pp. 180-184. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
5. Mfondoum, AH Ngandam, Mesmin Tchindjang, Jean Valery, M. Mfondoum, and I. Makouet. "Eisenhower matrix* Saaty AHP= Strong actions prioritization? Theoretical literature and

lessons drawn from empirical evidences." Iaetsd Journal For Advanced Research In Applied Sciences. Retrieved from https://www. iaetsdjaras. org/gallery/3-february-880. pdf (2019).

6. Mushtaq, Irfan, and Shabana Nawaz Khan. "Factors affecting students' academic performance." Global journal of management and business research 12, no. 9 (2012): 17-22.
7. Yegnanarayana, Bayya. Artificial neural networks. PHI Learning Pvt. Ltd., 2009.
8. Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining. Introduction to linear regression analysis. John Wiley & Sons, 2021.
9. Dizon, R. J., H. D. Ermitanio, D. M. Estevez, J. Ferrer, S. J. Flores, K. M. Fontanilla, A. Frias, E. Galang, N. F. Guei, and Dr J. Sugay. "The effects of pomodoro technique on academic-related tasks, procrastination behavior, and academic motivation among college students in a mixed online learning environment." Globus Journal of Progressive Education 11 (2021): 58-63.
10. Rish, Irina. "An empirical study of the naive Bayes classifier." In IJCAI 2001 workshop on empirical methods in artificial intelligence, vol. 3, no. 22, pp. 41-46. 2001.
11. Priyanka, and Dharmender Kumar. "Decision tree classifier: a detailed survey." International Journal of Information and Decision Sciences 12, no. 3 (2020): 246-269.